UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

CONSTRUÇÃO DE UM PROTÓTIPO (HARDWARE E SOFTWARE) PARA SEGURANÇA PREDIAL ATRAVÉS DE MONITORAÇÃO VIA CÂMERA DIGITAL E DISPLAY GRÁFICO

JORGE DE ASSIS MEREGE NETO

BLUMENAU
2004

2004/1-17

JORGE DE ASSIS MEREGE NETO

CONSTRUÇÃO DE UM PROTÓTIPO (HARDWARE E SOFTWARE) PARA SEGURANÇA PREDIAL ATRAVÉS DE MONITORAÇÃO VIA CÂMERA DIGITAL E DISPLAY GRÁFICO

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Ciências da Computação — Bacharelado.

Prof. Antonio Carlos Tavares - Orientador

BLUMENAU
2004

2004/1-17

CONSTRUÇÃO DE UM PROTÓTIPO (HARDWARE E SOFTWARE) PARA SEGURANÇA PREDIAL ATRAVÉS DE MONITORAÇÃO VIA CÂMERA DIGITAL E DISPLAY GRÁFICO

Por

JORGE DE ASSIS MEREGE NETO

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Antonio Carlos Tavares, – Orientador, FURB
Membro:	Prof. Jomi Fred Hubner, FURB
Membro:	Prof. Miguel A. Wisintainer, FURB

Blumenau, 18 de junho de 2004

Dedico este trabalho aos meus pais, a quem espero que o aceitem na forma de um sincero e humilde pedido de desculpas.

O que não me destrói, me fortalece.

Turuts

AGRADECIMENTOS

À minha esposa, por ter me ensinado o verdadeiro significado de algumas palavras como amor, companheirismo e respeito e por me ter dado o meu filho Pedro, que sem nenhuma dúvida, é o meu bem mais precioso.

À AUCCON DO BRASIL LTDA, em especial a Alberto Chuffi Júnior a quem, antes de mais nada, considero um grande amigo.

Ao meu orientador, Antonio Carlos Tavares, por ter acreditado na conclusão deste trabalho e por não ter poupado esforços para me ajudar.

Ao professor Miguel Wisintainer, por sua incansável disposição para ajudar.

Ao Ariberto, monitor da FURB.

RESUMO

Este trabalho tem por objetivo principal dar continuidade à pesquisa desenvolvida por Santos (2002) que foi de especificar e implementar um protótipo de hardware e software para compartilhamento de imagens via interface serial diferencial balanceada, visando à implementação de um sistema de monitoramento para segurança residencial. A modificação principal está em adicionar um banco de memória para melhorar a velocidade de transmissão da imagem.

Palavras chaves: Banco de memória; Velocidade de transmissão.

ABSTRACT

This work has as a main objective to continue the research did by Santos(2002) that was to specify and to implement a hardware and software prototype that shares images by differential balanced serial interface, intending a home automation systems application. The main modification is to add a memory bank to improve the speed of image transmission.

Key-Words: Memory bank; Transmission speed

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

CI	-	Circuito Integrado
CISC	-	*Complex Instruction Set Computer*
CMOS	-	Complementary Metal Oxide Semiconductor
EEPROM	-	*Electrically Erasable Programable Read Only Memory*
EPROM	-	*Erasable Programable Read Only Memory*
FPS	-	*frames per second*
GBPS	-	Gigabits por segundo
I²C	-	*Inter Integrated Circuit*
LCD	-	*Liquid Cristal Display*
Mbps	-	Megabits por segundo
MSSP	-	*Master Synchronous Serial Port*
RAM	-	*Random Access Memory*
RISC	-	*Reduced Instruction Set Computer*
ROM	-	*Read Only Memory*
RS	-	*Recommended Standard*
SFR	-	*Special Function Register*
SPI	-	*Serial Port Interface*
USART	-	*Universal Synchronous Asynchronous Receiver/Transmiter*
USB	-	*Universal Serial Bus*

SUMÁRIO

1 INTRODUÇÃO

Os altos índices de violência, que estão cada vez mais presentes no nosso dia a dia, aumentaram a necessidade e a procura por soluções para reforçar a segurança de construções prediais como casas ou escritórios. Uma solução que tem se destacado, entre as várias centenas existentes no mercado, é o sistema de vigilância eletrônica que tem se mostrado bastante eficiente e, devido à evolução e surgimento de novas tecnologias, apresenta um custo viável e cada vez mais acessível à população em geral.

É possível encontrar diversas opções de vigilância eletrônica tanto no que tange preço e/ou complexidade que vai desde a utilização de um sensor de movimento ligado a uma sirene até complexos sistemas de monitoramento através de câmeras inteligentes que são capazes de reconhecer rostos ou movimentos específicos como, por exemplo, focar em uma pessoa que está com os braços erguidos, posição esta que é muito comum em assaltos à mão armada. Estas soluções podem ser encontradas em lojas especializadas do comércio e, principalmente, em feiras e congressos do setor que tornam estas tecnologias cada vez mais constantes e mais acessíveis ao público em geral (AURESIDE, 2004).

É neste cenário que se enquadrou o trabalho realizado por Santos (2002) que é de um protótipo de sistema de vigilância para auxiliar no controle de acesso de pessoas em áreas específicas através de monitoração constante de pontos pré-definidos.

Segundo Santos (2002), tanto *hardware* (parte física do sistema) quanto *software* (maneira como o hardware irá se comportar) são dedicados. Isto é, foram desenvolvidos especificamente para a função a que se destina, para que o custo final fique reduzido. Também, segundo Santos (2002), a interface de comunicação serial com tecnologia diferencial balanceada também é de baixo custo de implementação.

No trabalho de Santos (2002) foram identificados dois pontos principais que serão desenvolvidos neste trabalhado. Um destes pontos reside em um problema de captura e transmissão da imagem, causado pela baixa velocidade de transmissão de dados. Outro ponto que ficou claro que deveria ser desenvolvido é referente à qualidade da visualização da imagem capturada, uma vez que no trabalho desenvolvido por Santos (2002) não era possível nenhuma visualização de imagem.

1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é aprimorar o trabalho elaborado por Santos (2002) que visa a construção de uma rede digital (hardware e software) para monitoramento de ambientes via câmeras digitais.

Os objetivos específicos do trabalho são:

- reestruturar o hardware do módulo de captura de imagens e respectivo software para armazenar a imagem a ser tratada e transmitida.
- visualizar as imagens do módulo de monitoramento (LCD)

1.2 ESTRUTURA DO TRABALHO

O texto está organizado de forma a possibilitar uma visão geral do trabalho realizado por Santos (2002) com uma explanação das teorias e tecnologias utilizadas por Santos (2002) em conjunto com as teorias e tecnologias que foram adicionadas neste trabalho, buscando aprofundamento apenas em questões que fogem do censo comum na área de ciências da computação.

No capítulo 2 encontra-se uma curta revisão do trabalho realizado por Santos (2002), disposta por assunto, das diversas tecnologias envolvidas. É feita uma introdução a cada componente utilizado por Santos (2002), em alguns casos com comentários adicionais, e também dos componentes que foram adicionados durante o desenvolvimento deste trabalho como, por exemplo, a memória SRAM e o microcontrolador 8051 da ATMEL. Neste trabalho não foram feitos comentários a respeito de componentes conhecidos do grande público e de estrutura mais simples, como é o caso dos resistores e capacitores. Sobre estes dois últimos e demais componentes mais simples, poderão ser encontrados maiores detalhes em Eletrônica-Senai (2002).

O capítulo 3 apresenta as ferramentas utilizadas para o desenvolvimento deste trabalho, as funções e modo de funcionamento a fim de permitir uma melhor avaliação pela separação dos conteúdos, além da especificação, implementação e testes. A disposição das informações visa salientar a maneira como foi desenvolvido o trabalho. Dispostas em ordem cronológica de implementação estão a visualização e armazenamento da imagem, que foram assim incorporados ao protótipo de Santos (2002). As formas como Santos (2002) trabalhou quanto a captura e tratamento da imagem além da rede de comunicação entre o módulo de

captura (câmera) e o módulo de visualização (monitor), foram quase que abstraídas sendo citado somente o que está diretamente relacionado com as modificações realizadas neste trabalho.

O capítulo 4 destaca, então, as conclusões alcançadas. Traz também sugestões para o desenvolvimento de trabalhos correlatos, a quem possa eventualmente interessar-se pelo mundo do hardware.

2 FUNDAMENTAÇÃO TEÓRICA

No trabalho de Santos (2002) e, conseqüentemente neste trabalho, foram aplicadas técnicas e ferramentas que visam a integração entre hardware e software, obrigando ao pesquisador ter conhecimento básico em eletrônica. Isto faz aflorar uma característica quase que obrigatória nos profissionais da informática, em especial os desenvolvedores, que é conhecer o ambiente onde o software será executado, não importando o nível da linguagem utilizada ou a arquitetura do hardware em questão.

Estando bem claro que este trabalho exige um conhecimento básico em eletrônica, passa-se a tratar do que cabe á área de ciências da computação como arquitetura de microcontroladores, ambientes de programação (incluindo linguagem de máquina), comunicação e armazenamento de dados são técnicas utilizadas na implementação e que caracterizam aspectos relevantes deste trabalho.

O fato de este trabalho ser uma continuação do trabalho desenvolvido por Santos (2002), faz com que o mesmo seja citado em quase todos os parágrafos a seguir. Santos (2002) propõe a captura de imagens e sua transmissão via interface diferencial balanceada, buscando uma visualização em um *display* de cristal líquido. A diferença principal do trabalho a seguir é a utilização de um banco de memória com a função de armazenar os dados coletados pela câmera antes de envia-los de uma única vez ao LCD, para que a velocidade de transmissão seja suficiente para que a imagem capturada pela câmera possa ser visualizada no LCD, o que não ocorreu com o trabalho desenvolvido por Santos (2002).

2.1 DIFERENÇA ENTRE MICROCONTROLADORES E MICROPROCESSADORES

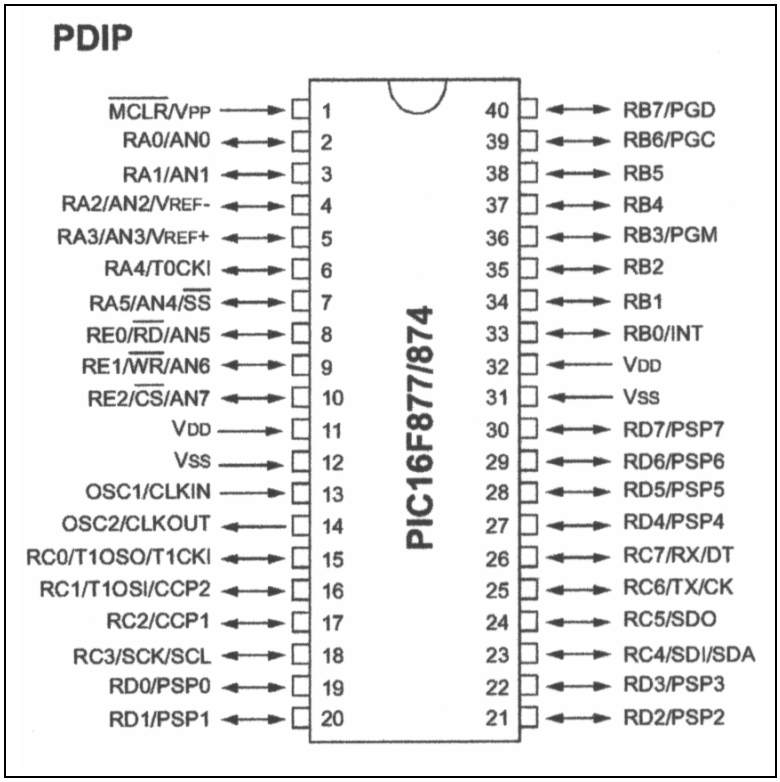
Segundo Nicolosi (2000), o *hardware* interno do *chip* do microcontrolador possui muito mais funções do que um *chip* de microprocessador e que, na maioria das aplicações utilizando microprocessadores necessita-se, além do *chip* do microprocessador, de outros *chips* como o da ROM, do *Latch* e da RAM, e de outros *chips* auxiliares como *timers* e *chips* para fazer a interface como barramento do microprocessador para gerar uma linha serial de comunicação. O microcontrolador, normalmente já possui estes *chips* dentro a mesma pastilha do microprocessador. Segundo Nicolosi (2000), o microcontrolador corresponde a um microprocessador e seus periféricos típicos, todos juntos num só *chip.*

2.2 MICROCONTROLADOR PIC16F877

De acordo com Santos (2002), este é um microcontrolador que possui praticamente todas as opções disponíveis em termos de periféricos. Este dispositivo possui um tempo de execução de cada instrução de 200 nanosegundos (com um *clock* de 4 Mhz) com uma arquitetura de 40 ou 44 pinos (MICROCHIP,2004). Neste trabalho foram utilizados estes dispositivos com 40 pinos que estão detalhados na fig. 1. Abaixo segue um resumo das principais informações obtidas em Microchip (2004):

a) CPU de alto desempenho RISC (35 instruções), com *clock* máximo de 20MHz;
b) memória FLASH de programa com 8K palavras de 14 bits;
c) memória de dados: RAM (368 bytes) e EEPROM (256 bytes);
d) 14 pinos de interrupção configuráveis;
e) pilha com 8 níveis;
f) modos de endereçamento direto, indireto e relativo;
g) circuitos e temporizadores de inicialização do sistema para garantir estabilização de clock e integridade de dados em caso de inicialização do sistema;
h) temporizador *watchdog* com oscilador próprio;
i) proteção de código programável;
j) opções de freqüência de oscilador selecionáveis;
k) programação serial e depuração de erros *in-circuit* usando dois pinos;
l) faixa de alimentação de 2 a 5,5Vcc;
m) capacidade de corrente de 25mA;
n) baixo consumo – em modo *SLEEP* ainda mais econômico;
o) dois temporizadores/contadores de 8 bits e um de 16 bits, com *prescaler*;
p) dois módulos PWM com resolução de 10 bits;
q) conversor analógico/digital de 8 ou 10 bits;
r) porta serial síncrona com SPI e I²C e USART com detecção de endereço;

s) porta paralela de 8 bits.



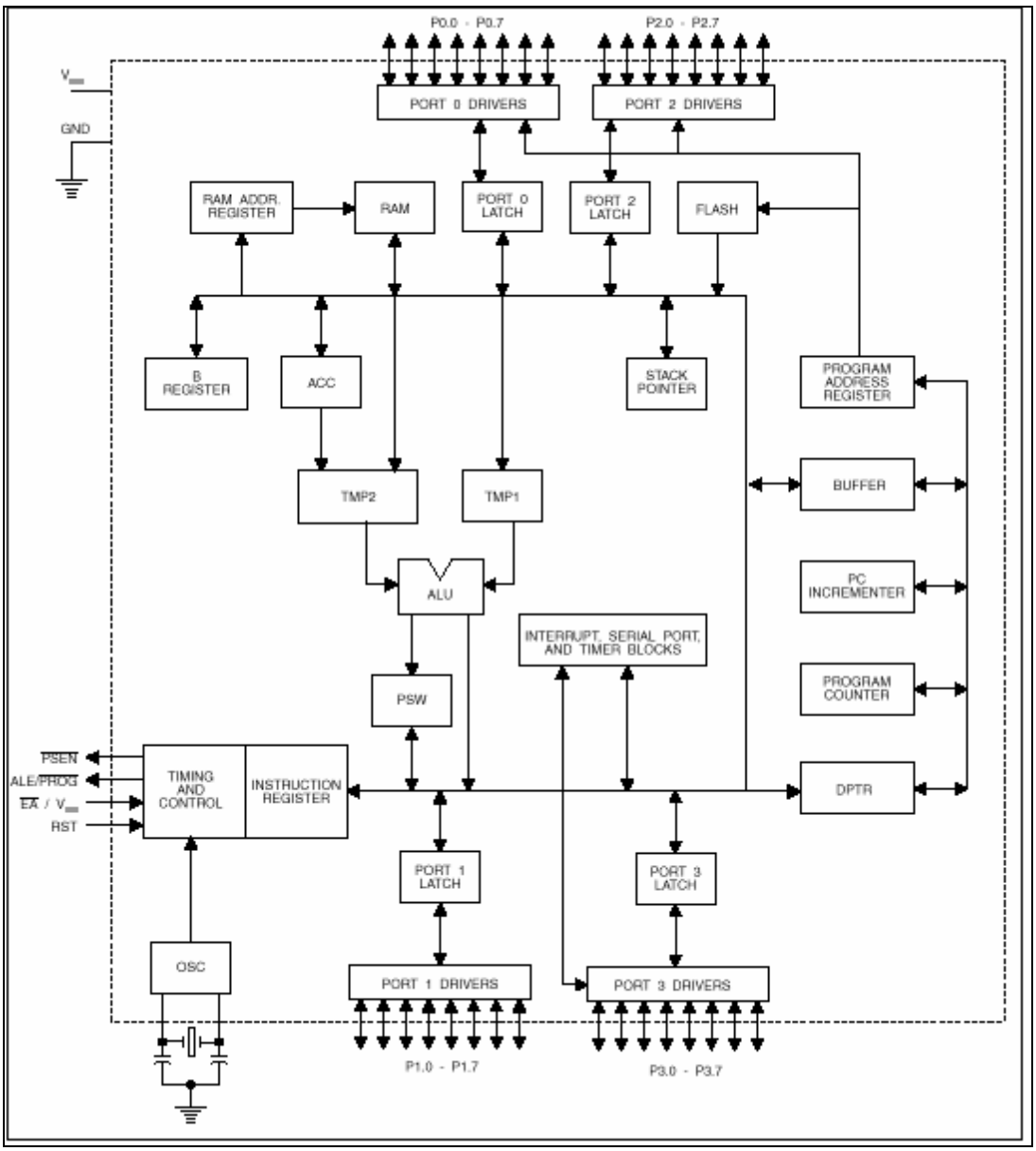
Fonte : Adaptado de Microchip (2002)

FIGURA 1 – pinos do PIC 16F877 com encapsulamento DIP

O microcontrolador PIC é aplicado no módulo denominado por Santos (2002) como MONITOR, módulo este que recebeu modificações no *software,* sendo que permaneceu com as mesmas características originais no *hardware*, desenvolvido por Santos (2002).

2.3 MICROCONTROLADORES 8051

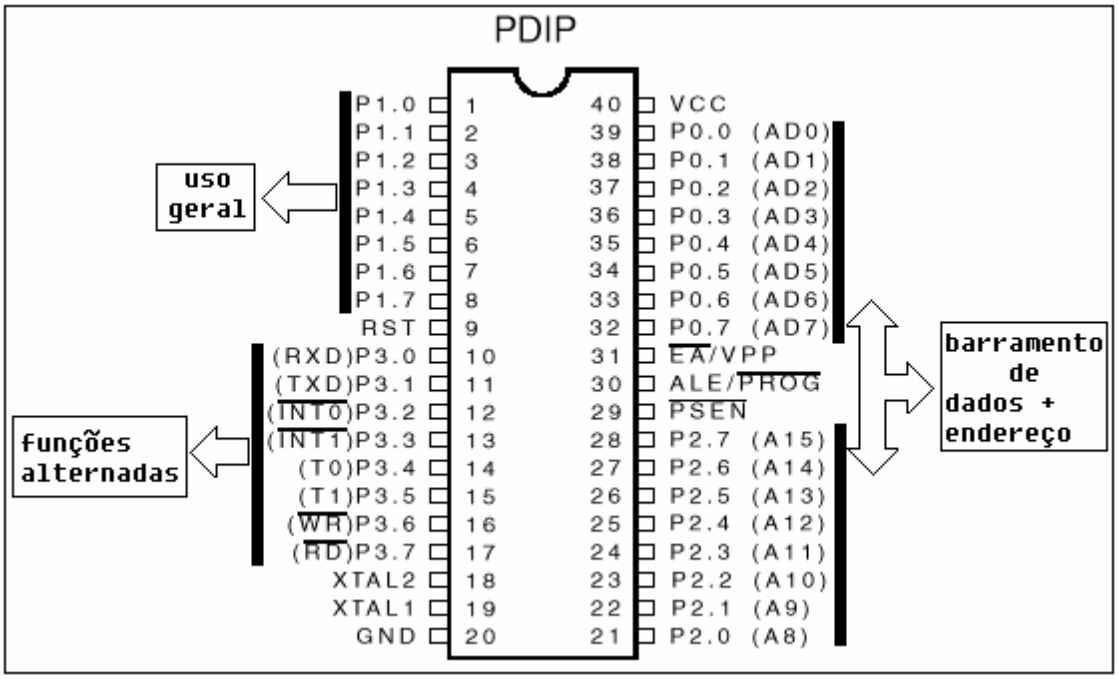
O 8051 foi criado pela empresa Intel no início da década de 80. Segundo Klitzke (1999) o dispositivo em si é um microcontrolador relativamente simples, mas com ampla aplicação. Também segundo Klitzke (1999) o mais importante é que não existe somente o CI 8051, mais uma família de microcontroladores (MCS51) baseada no mesmo, o que significa que existem vários modelos de microcontroladores que possuem os mesmos elementos básicos com um mesmo conjunto básico de instruções. As empresas que fabricam microcontroladores compatíveis com o 8051 da Intel são: Amd, Atmel, Dallas, Matra ,OKI, Philips, Siemens, SMC e SSI (KLITZKE, 1999). Na figura 2, é possível ver a arquitetura básica do 8051.



Fonte : Klitzke (2004, p.10)

FIGURA 2 – arquitetura básica do 8051

Neste trabalho, foi utilizado o chip da família MCS51 modelo AT89C55WD fabricado pela empresa ATMEL que é um modelo de alto desempenho com 20k de memória de leitura (*flash*) programável e 256 bytes de memória RAM. Possui 40 pinos sendo que 32 pinos são de I/O e a freqüência máxima deste microcontrolador é de 33 Mhz (NICOLOSI, 2000). Na figura 3, está exposta a numeração de todos os pinos de um chip 8051 e respectivas funções.



Fonte : Klitzke (2004, p.11)

FIGURA 3 – Pinos 8051

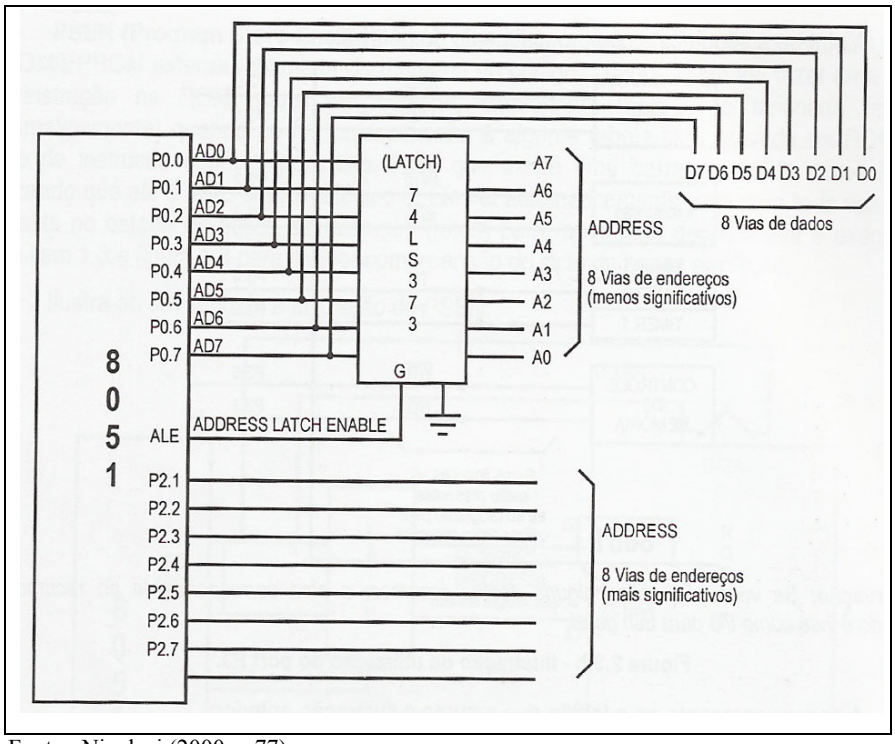
Na tabela 1 existe uma descrição de cada uma das portas existentes neste *chip*

Tabela 1 – Descrição das portas do *chip* 8051

Porta	**Descrição**
P0	Port de propósito geral. Quando for um usado uma memória externa é possível utilizar este port como uma via multiplexada para enviar hora dado, hora endereço
P1	Port de propósito geral como I/O
P2	Port de propósito geral. Quando for um usado uma memória externa é possível utilizar este port como endereçamento desta memória.
P3	Port de propósito geral de I/O. Também pode ser utilizado por algum periférico externo como porta serial, timers e memória.

Um dos motivos pelo qual foi escolhida a utilização do 8051, que neste trabalho foi aplicado no módulo onde existe a câmera, foi a facilidade de trabalho e a economia de pinos que, segundo Nicolosi (2000), é gerada por sua estrutura quando da utilização de uma memória externa. Através da utilização do pino 30 (ALE), é possível ligar um *latch* (descrito no capítulo 2.5) para demultiplexar externamente os dados e endereços no tempo, separando assim as informações. Isto é transparente ao programador, pois não é necessária a preocupação com o comando desse pino ALE por este ser automaticamente gerenciado pelo microcontrolador, basta conectar o *latch*neste pino e o controle é feito internamente pelo microcontrolador. Na figura 4, é possível ver um exemplo de esquema que mostra a ligação

de um *latch* e como os pinos trabalham com o envio de endereço/dados para uma memória externa.



Fonte : Nicolosi (2000, p.77)

FIGURA 4 – Esquema de ligação do 8051 com um *latch*

Através do pino 29 (PSEN), é possível saber se a operação em andamento é uma leitura de instrução (acesso à memória de programa) ou um acesso à memória de dados. Este pino permite que o microcontrolador tenha duas regiões distintas de memória externa, uma para armazenar código e outra para dados.

O pino 31 (EA) é um sinal de entrada, através do qual o usuário escolhe se será utilizada a memória ROM interna ou se o programa será armazenado externamente.

2.4 COMUNICAÇÃO DE DADOS

Segundo Santos (2002), para que a imagem coletada pelas câmeras possa chegar ao monitor, uma rede de comunicação de dados deve ser implementada.

Quanto ao método de transferência de dados, é notório que a transferência paralela leva vantagem sobre a serial quando as distâncias não ultrapassem poucos metros. Porém, no caso da proposta, as distâncias podem envolver dezenas de metros, o que viabiliza a transferência serial em detrimento à paralela (SANTOS, 2002).

Segundo Santos (2002), existem especificações de padrões seriais criados há vários anos, mas que continuam sendo largamente utilizados. É o caso dos padrões *Recommended Standard* (RS), que são normas criadas para a especificação de interfaces de comunicação de dados. Com base nas informações da tabela 2, pode-se verificar que o padrão RS-485 é uma boa opção para a criação de uma rede multiponto (até 32 pontos) que necessite de distâncias de até 4000 pés (cerca de 1200m) e que necessite de boa taxa de transferência de dados. Comparação entre outros padrões de interface pode ser verificada em Klitzke (1999), onde fica latente a opção da RS-485 para longas distâncias.

Tabela 2 – Especificações de Padrões RS

SPECIFICATIONS		RS232	RS423	RS422	RS485
Mode of Operation		SINGLE -ENDED	SINGLE –ENDED	DIFFER- ENTIAL	DIFFER- ENTIAL
Total Number of Drivers and Receivers on One Line		1 DRIVER 1 RECVR	1 DRIVER 10 RECVR	1 DRIVER 10 RECVR	1 DRIVER 32 RECVR
Maximum Cable Length		50 FT.	4000 FT.	4000 FT.	4000 FT.
Maximum Data Rate		20kb/s	100kb/s	10Mb/s	10Mb/s
Maximum Driver Output Voltage		+/-25V	+/-6V	-0.25V to +6V	-7V to +12V
Driver Output Signal Level (Loaded Min.)	**Loaded**	+/-5V to +/-15V	+/-3.6V	+/-2.0V	+/-1.5V
Driver Output Signal Level (Unloaded Max)	**Unloaded**	+/-25V	+/-6V	+/-6V	+/-6V
Driver Load Impedance (Ohms)		3k to 7k	>=450	100	54
Max. Driver Current in High Z State	**Power On**	N/A	N/A	N/A	+/-100uA
Max. Driver Current in High Z State	**Power Off**	+/-6mA @ +/-2v	+/-100Ua	+/-100uA	+/-100uA
Slew Rate (Max.)		30V/uS	Adjustable	N/A	N/A
Receiver Input Voltage Range		+/-15V	+/-12V	-10V to +10V	-7V to +12V
Receiver Input Sensitivity		+/-3V	+/-200mV	+/-200mV	+/-200mV
Receiver Input Resistance (Ohms)		3k to 7k	4k min.	4k min.	>=12k

Fonte: Santos (2002).

2.4.1 INTERFACE RS-485

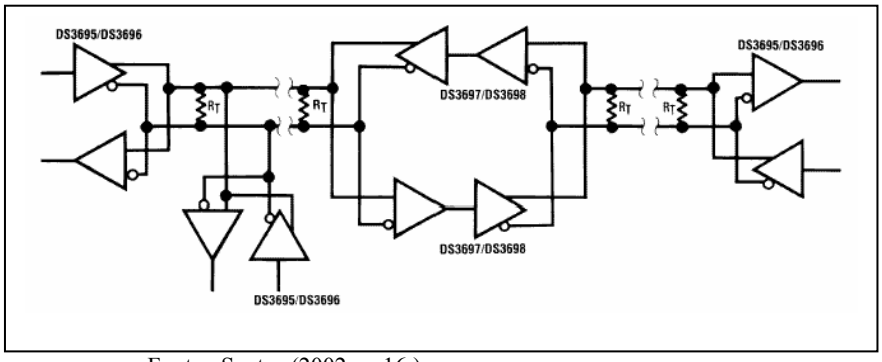
A RS-485, como padrão de comunicação, especifica as características elétricas de uma interface serial digital entre circuitos, baseada em tensão diferencial balanceada. O padrão especifica o modo de transmissão assíncrono, mas não determina, por exemplo, o tipo de condutor (normalmente par trançado) nem o protocolo (geralmente proprietário) (Franco, 2001).

Uma interface balanceada é mais vantajosa para altas taxas de transmissão (até 10 Mbps), longas distâncias e ambientes ruidosos. A lógica para este padrão é definida quando um terminal torna-se mais negativo ou mais positivo que o outro. Portando, a tensão diferencial entre os dois terminais permitirá o reconhecimento do bit que está sendo transmitido, se é "0" ou se "1". Convencionalmente, o valor lógico "1" é reconhecido quando o terminal A do transmissor torna-se mais negativo em relação ao terminal B. O valor lógico

"0" é identificado quando o terminal A torna-se positivo em relação ao terminal B, é por esse motivo que se tem maior imunidade a ruídos eletromagnéticos externos (FRANCO, 2001).

Segundo Santos (2002), existem dois cuidados a serem tomados com relação à instalação da rede em si. São cuidados que previnem problemas elétricos:

a) para garantir a não existência de ondas estacionárias que podem causar falha de comunicação, um resistor de terminação, ou terminador, de 120 ohms deve ser colocado nas extremidades da rede, conforme figura 5;
b) para maior segurança pessoal e do equipamento, o cabo deve possuir malha que deverá ser aterrada em apenas uma das extremidades (Atos, 2000).

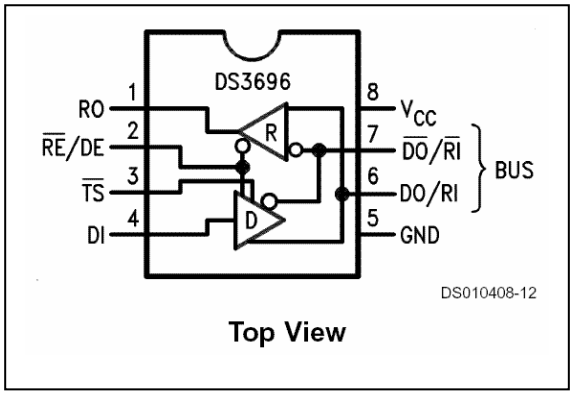


Fonte : Santos (2002, p. 16)

FIGURA 5 – Ligação do terminador na rede 485

2.4.2 CIRCUITO INTEGRADO INTERFACE RS-485

O DS3696 é um dispositivo desenvolvido em concordância com a norma EIA RS485 para transmissão multiponto de dados (NATIONAL, 1998). Ainda segundo este mesmo autor, este circuito é um transmissor diferencial de alta velocidade cuja descrição dos pinos pode ser vista na figura 6.



Fonte: Santos (2002, p. 17)
FIGURA 6 – Descrição dos pinos do DS3696

A alimentação do *chip* se dá através dos pinos 5 (GND) e 8 (Vcc). Os pinos RO e DI representam a interface lógica e são diretamente conectadas aos pinos RX e TX de microcontroladores com módulo de comunicação serial. O controle de transmissão/recepção (half-duplex) é feito através do pino $\overline{DE}/RE$, onde um sinal de nível alto (1) habilita o dispositivo a transmitir, enquanto o nível baixo (0) inibe a transmissão e põe o DS3696 em estado de recepção.

Os pinos 6 e 7 formam a interface diferencial do circuito e são conectados entre si. Ou seja, todos os pinos 6 de todos os DS3696 presentes na rede são interligados. Da mesma forma, todos os pinos 7 são também interligados.

Quando um DS3696 transmite, o sinal que recebe em sua interface lógica é convertido para tensão diferencial aplicada nos pinos 6 e 7, que serão recebidos pelos dispositivos do outro lado e reconvertidos em sinal digital (SANTOS, 2002).

Segundo Santos (2002), o cuidado que deve ser tomado pelo desenvolvedor é quanto ao controle feito no $\overline{DE}/RE$ e quanto ao dado lido/escrito nos pinos RX/TX, pois tudo o que acontece entre esses circuitos de interface pode ser desconsiderado em nível de implementação de software.

2.5 CIRCUITO 74LS373 – *LACTH*

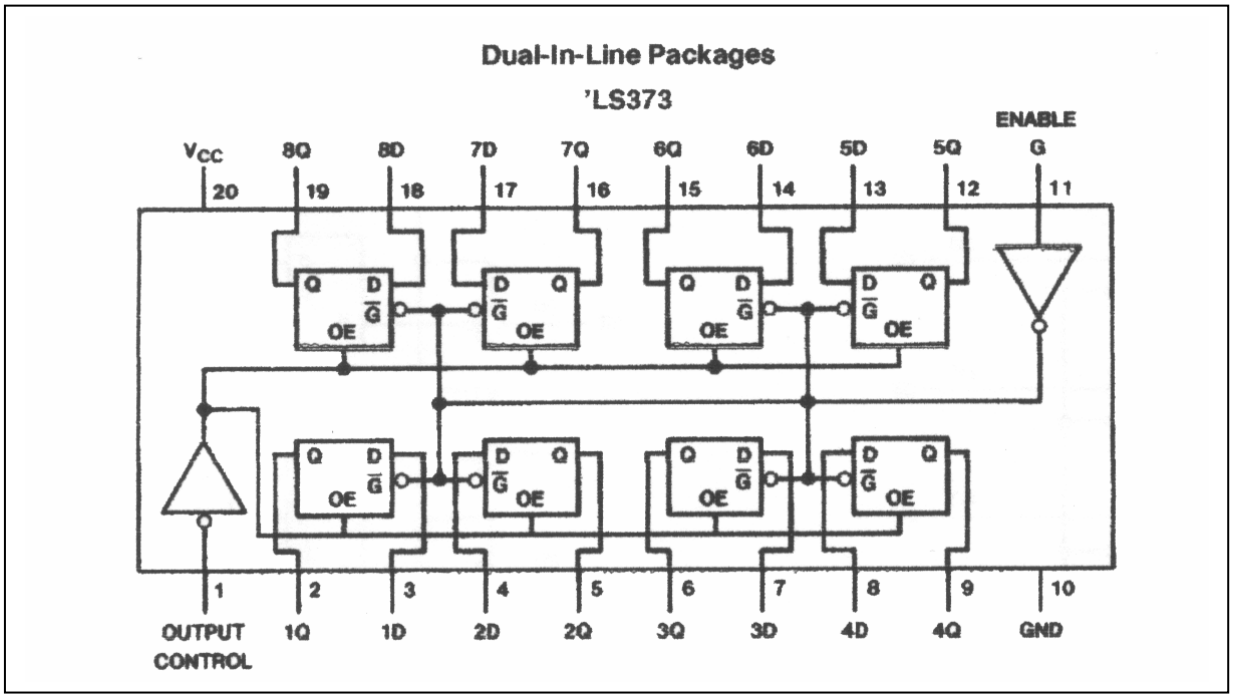
Segundo UEM (2003), os *lacthes* são circuitos lógicos muito importantes, que consistem basicamente do acoplamento de um par de inversores, de forma que, a saída do

primeiro inversor é ligada a entrada do segundo e que a saída do segundo é ligada à entrada do primeiro.

Ainda segundo UEM (2003), sem intervenção externa, o *latch* permanece indefinidamente em uma de duas situações possíveis chamadas estados. Por essa característica, o *latch*pode ser usado para estabelecer e manter um nível lógico sem qualquer intervenção externa. Essa independência do *latch* das entradas externas, possibilita-o a ser usado para armazenar, ou lembrar um bit lógico (valor 0 ou 1).

O circuito 74LS373 usado neste trabalho é muito interessante para aplicações como implementações de *buffers* e de portas de I/O (NATIONAL, 1998). Por possuir 8 *latches* em um único *chip*, consegue armazenar e transmitir dados de até 8 bits.

Na figura 7 está um esquema de um circuito 74LS373 com a localização dos pinos e suas respectivas funções. Os pinos com identificados com a letra D são designados para a entrada de dados e os pinos identificados com a letra Q são designados para a saída dos dados (NATIONAL, 1998).



Fonte : Adaptado de NATIONAL (1998)
Figura 7 – Pinos do 74LS373

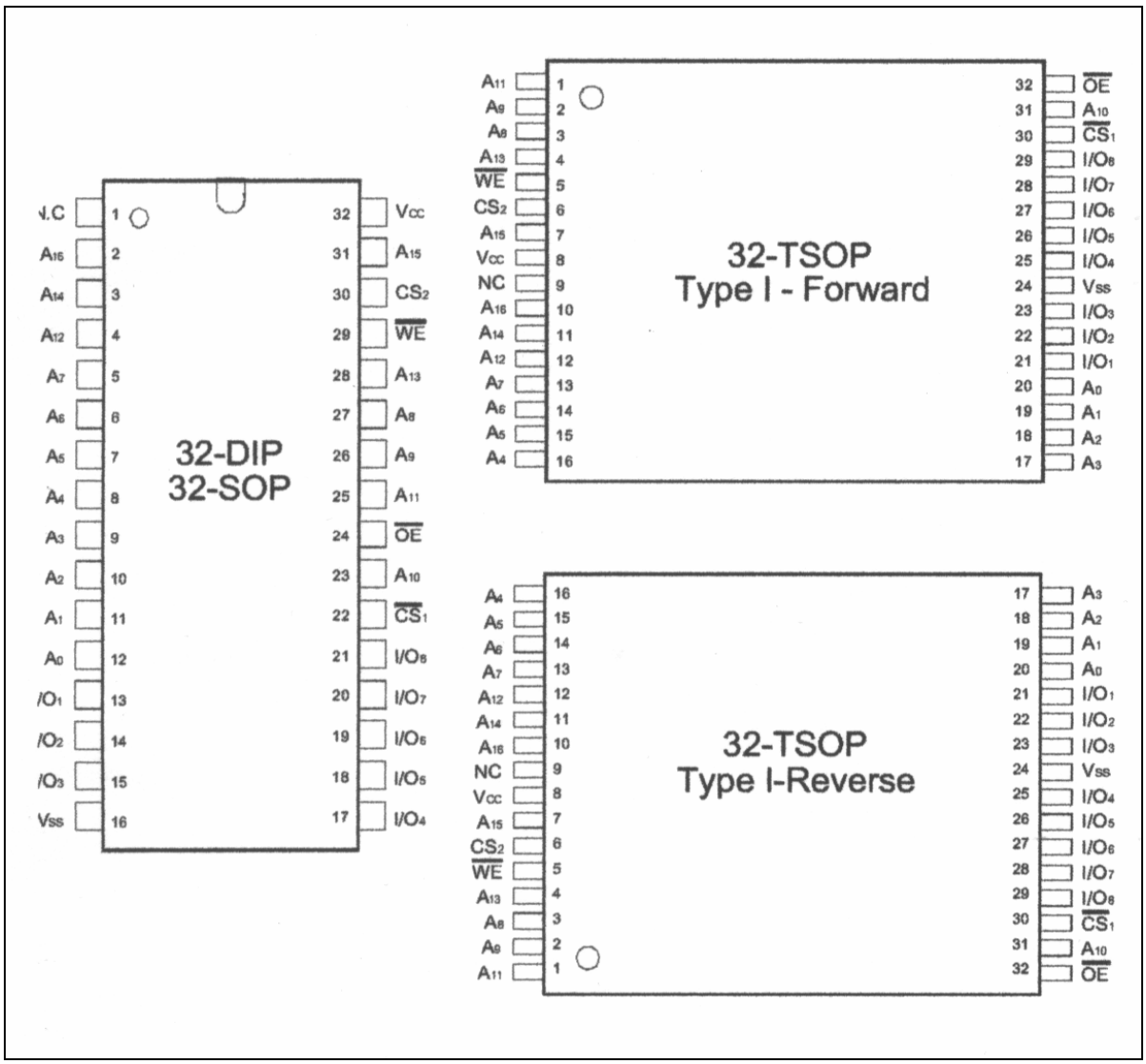
2.6 MEMÓRIAS SRAM

Este é um tipo de memória muito aplicada para formar a memória *cache* em computadores, *palmtops* ou de qualquer equipamento que necessite de armazenamento de dados, devido a sua alta velocidade. A sigla SRAM vem de S*tatic Random Access Memory* ou memória estática de acesso aleatório. A SRAM é um tipo de memória que somente mantém a informação armazenada enquanto a alimentação estiver aplicada ao *chip*. As células de memória aplicadas nas memórias SRAM são formadas por *flip-flops* que estarão em certo estado (1 ou 0), por tempo indeterminado e estão disponíveis nas tecnologias bipolar e *MOS* (CLUBE DAS REDES, 2003).

Segundo UFPB (2003), o que torna a SRAM muito mais rápida do que qualquer das demais modalidades de memória DRAM (*Dinamic Random Access Memory* ou memória dinâmica de acesso aleatório) é que ela não precisa do contínuo refrescamento de seu conteúdo para evitar a perda de dados. Essa característica, juntamente com outras particularidades técnicas, faz com que a memória SRAM seja muito rápida, chegando a alcançar tempos de acesso inferiores a 2 nanosegundos.

Em vez dos diminutos acumuladores característicos das memórias DRAM, as memórias SRAM contam com um grupo de seis transistores para o armazenamento de cada bit. Em conseqüência, ocorre uma drástica redução no tempo de acesso, pois se evitam os atrasos criados pelos processos de carga e descarga elétrica em cada acumulador. A presença de transístores melhora o rendimento das memórias estáticas, mas implica uma renúncia à alta densidade de armazenamento, típica das memórias DRAM. O resultado é um considerável aumento no tamanho físico nos módulos de memória SRAM e, também, em seu custo de produção, o que impossibilita usá-la como memória principal (UFPB, 2003).

Neste trabalho foi utilizado um *chip* de memória SRAM de 1 megabit (128k x 8 bits) com encapsulamento DIP para facilitar a sua inserção e conexão no *protoboard.* Na figura 8 é possível ver um *chip* de memória SRAM com os diferentes encapsulamentos em que é comercializada e na tabela 3 a descrição dos pinos no encapsulamento DIP.



Fonte: Adapatdo de NEC (2002).

FIGURA 8 – pinos do *chip* de uma memória SRAM

Tabela 3 – Descrição dos pinos de um *chip* de memória SRAM com encapsulamento DIP

Pino	**Descrição**
A0 – A16	Endereços de entradas
I/O1 – I/O8	Entrada / Saída de dados
CE1, CE2	*Chip Enable* 1,2
WE	*Write Enable*
OE	*Output Enable*
Vcc	Alimentação
GND	Terra
NC	Não conectado

Fonte: adaptado de NEC (2002)

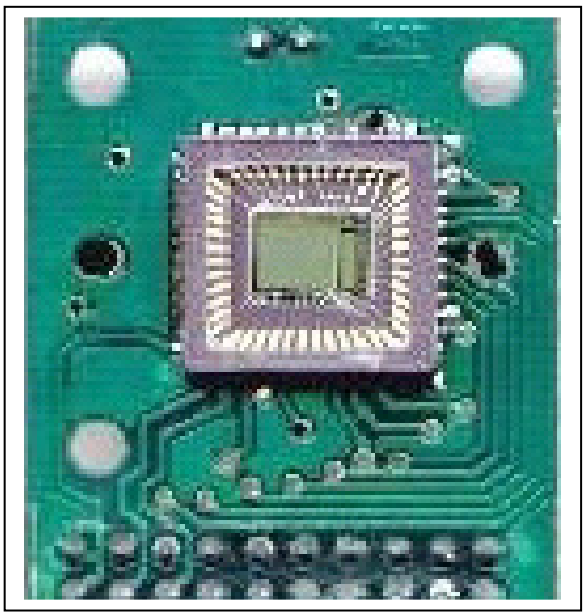
2.7 CAPTURA DE IMAGENS

Apesar de ainda longe da excelência da imagem analógica, já existem câmeras digitais com resolução de 1800 por 1200 *pixels* – muito superior do que resoluções de 640 por 800 *pixels* dos primeiros modelos de câmeras fotográficas comerciais de 1997. A qualidade de imagem depende de vários fatores, mas o principal deles é a resolução do sensor, dispositivo sensível à luz formado por diversas células capazes de armazenar informação (SANTOS, 2002). Klitzke (1999) fornece maiores detalhes sobre o funcionamento destes dispositivos.

2.7.1 Módulo de câmera digital M4088

Neste trabalho, como no de Santos (2002), foi ignorado o modo como as câmeras digitais capturam as imagens. Informações detalhadas sobre o módulo M4088 pode ser encontrada em Comedia (1999).

O módulo é mostrado na figura 9, sem a lente, para dar ênfase ao sensor, que utiliza tecnologia CMOS. Esta tecnologia em conjunto com uma interface digital de fácil uso, faz deste conjunto uma solução de baixo custo para aplicações de vídeo de boa qualidade.



Fonte: Santos (2002, p. 19)
Figura 9 – Módulo M4088

Segundo Santos (2002), O M4088 possui uma porta de vídeo digital que fornece fluxo contínuo de dados de 8 bits. As configurações da câmera incluem taxa variável de *frames* (quadros), ajuste de exposição e tamanho da imagem. Todas as configurações são baseadas

em registradores. A tabela 4 demonstra os pinos e sinais necessários para a operação do módulo em modo digital.

Tabela 4 – Pinos do módulo M4088

Pino	Mnemônico	Tipo	Função
1 a 8	D0 a D7	Entrada/saída digitais	Barramento bidirecional de dados
9	VCC	Alimentação	5Vcc
10	GND	Alimentação	0Vcc
11 a 14	A3 a A0	Entrada digital	Entrada de endereço para registradores internos
15	OEB	Entrada digital	Habilita saída de dados (8 bits)
16	WEB	Entrada digital	Habilita escrita (registradores internos)
17	CSB	Entrada digital	Habilita dispositivo (chip enable)
18	HREF	Saída digital	Saída de sincronismo horizontal
19	PCLK	Saída digital	Saída para sincronismo pixel a pixel
20	VSYNC	Saída digital	Saída de sincronismo vertical

Fonte : Santos (2002)

Segundo Omnivision Technologies (1998), módulo M4088, na verdade, é uma placa de circuito impresso composta pelo sensor OV5017, uma lente para concentrar a luz no sensor, alguns componentes discretos externos e dois conectores de saída: um analógico (vídeo composto) e um digital. E ainda, o OV5017 é um *chip* completo para câmera de vídeo analógica. Com taxa de exibição de *frames* ou de *pixels* programável para adaptar-se às necessidades dos sistemas externos, tem resolução de 8 bits e fornece *frames* de até 384 x 288 *pixels*. O conversor interno A/D pode converter o sinal de vídeo a 50fps (*frames per second*, ou quadros por segundo), e a conversão é sincronizada conforme taxa de exibição de *pixels*. Ele fornece sinais de referência padrão de tempo, como VSYNC, HREF, PCLK (vistos na tabela 3).

2.8 VISUALIZAÇÃO DE IMAGENS

A visualização de imagens processadas eletronicamente envolve desde métodos de definição da cor até a compatibilização dos sistemas de coordenadas dos sistemas envolvidos. Todo esse esforço é desprendido a fim de gerar a imagem da maneira como foi capturada, ou se for o caso, de modo aproximado (SANTOS, 2002).

Segundo Santos (2002), existem dois tipos básicos de *displays*: os alfanuméricos (ou de segmentos) e os gráficos. Os alfanuméricos, também chamados de display de caracteres, são encontrados em visores de calculadoras e relógios de pulso, enquanto que os gráficos estão presentes em monitores de *notebooks* e telefones celulares. Mas existem outras formas de classificar esses dispositivos, principalmente:

a) quanto a sua cor: monocromáticos, tons de cinza e coloridos;

b) quanto ao tipo de iluminação: reflexivos (dependem da luz ambiente) e iluminados (possuem *backlight* - luz de fundo);
c) quanto à tecnologia de fabricação: passivos (comuns e mais baratos) e ativos (maior nitidez de imagem).

Segundo Oki Semiconductor (2002), os LCD´s são compostos por uma película de cristal líquido contida por duas lâminas de vidro que possuem trilhas condutivas para polarizar os cristais. Em seu estado normal (desenergizados) os cristais permitem que a luz atravesse diretamente, deixando-se ver a cor natural de fundo. Uma vez energizados, redirecionam a luz, que é absorvida causando a impressão de ocorrência de sombra.

O sinal aplicado para polarizar as moléculas de cristal deve ser alternado, para evitar degradação da qualidade do *display*. Isto é feito a intervalos regulares, a fim de executar o *refresh* da imagem, isto é, a imagem deve ser periodicamente reescrita, caso contrário ela gradativamente desaparece. Essa tarefa cabe a controladores dedicados, ou LCD *controllers*, embora possam existir *displays* sem esses dispositivos (SANTOS, 2002).

2.8.1 Módulos LCD

Segundo Oki Semiconductor (2002), denominam-se módulos LCD os conjuntos formados pela tela de cristal líquido, dispositivos para reflexão da luz ou iluminação e demais componentes eletrônicos, com ou sem controladores.

Segundo Santos (2002), os módulos que contém controladores são mais práticos, pois podem ser configurados através de registros internos, além de evitar o trabalho de implementar a varredura. A escrita e leitura desses registros, de forma análoga aos módulos de câmera digital, também são feitas como se o dispositivo fosse uma memória ou outro circuito digital endereçável.

No trabalho de Santos (2002) e neste, foi utilizado o LCD DG16080 que possui um controlador LC7981 da Sanyo (2002). Este LCD é do tipo monocromático e pode ser configurado para trabalhar tanto no modo caracter como no modo gráfico. No modo gráfico, este LCD trabalha com 160 colunas por 80 linhas e, sendo monocromático, cada *pixel* é representado por apenas 1 bit o que em uma matriz de 160x80 totaliza 12.800 pontos que, divididos por 8 bits faz um total de 1.600 bytes.

Segundo o manual do fabricante (SANYO, 2002), a escrita no LCD é feita da esquerda para a direita e de cima para baixo, de oito em oito *pixels*. Na tabela 5 estão todas as configurações que devem ser feitas antes de se trabalhar com este LCD e na seqüência sugeria pelo fabricante (SANYO, 2002). No apêndice A encontra-se o *software* completo em PIC BASIC do módulo MONITOR onde pode ser visto todos os passos necessários para configurar o LCD para trabalhar no modo gráfico e escrever o dado capturado pela câmera.

Tabela 5 – Configurações necessárias para o módulo M4088

Descrição	**Opções**
Número de caracteres	Índica o número de *bytes* por linha. Santos (2002) utilizou 17
Refresh do *display* (varredura)	Freqüência do *refresh* que pode ir de 1 a 256. Santos (2002) utilizou 79
Quantitade de *bits* por *byte*	Para este LCD pode ser 6,7,8. Neste trabalho foi utilizado 8 bits
Modo gráfico ou caracter	Foi escolhido módulo gráfico
Endereço mais baixo do cursor	Santos (2002) utilizou o endereço 00h
Endereço mais alto do cursos	Santos (2002) utilizou o endereço 00h
Endereço mais baixo do LCD	Foi utilizado 00h
Endereço mais alto do LCD	Foi utilizado 00h
Escrever dados no LCD	Foi escrito o dado lido na porta serial

Segundo Sanyo (2002), na configuração do módulo DG-16080 há somente um barramento que serve como endereçamento e outro que serve para dados. O procedimento para escrita no módulo é feito em duas etapas. Primeiro passa-se o endereço do registrador interno e depois o dado a ser escrito nele. Cada etapa é executada da seguinte forma:

a) joga-se o byte no barramento;
b) seleciona-se o tipo de operação (escrita/leitura) no pino R/W (pino 5);
c) através do pino RS (pino 4) é definido se o byte presente no barramento é um endereço ou dado;
d) pulsa-se o pino E (pino 6) para habilitar a operação.

O ciclo de leitura é semelhante. Na primeira etapa escreve-se o endereço do registrador a ser lido e, para executar a segunda etapa, basta aplicar nível baixo (nível lógico 0) no pino R/W. Logo após ter sido habilitada a operação, o dado está disponível no barramento para ser lido. Na tabela 6, estão dispostos os pinos do módulo DG-16080.

Tabela 6 – Descrição dos pinos do módulo DG-16080

Pino	Mnemônico	Tipo	Função
1	Vss	Alimentação	0Vcc
2	Vdd	Alimentação	Alimentação dos circuitos digitais
3	Vo	Alimentação	Alimentação do display
4	RS	Entrada digital	Seleciona instrução (1) / dado (0)
5	R/W	Entrada digital	Seleciona leitura (1) / escrita (0)
6	E	Entrada digital	Habilita operação (enable)
7 a 14	DB0 a DB7	Entrada/saída digitais	Barramento bidirecional de dados
15	CS	Entrada digital	Habilita dispositivo (chip enable)
16	Res	Entrada digital	Reset
17	Vee	Alimentação	Saída de voltagem (10V)
18 a 20	NC	-	Não conectados

3 DESENVOLVIMENTO DO TRABALHO

Este trabalho segue a mesma forma utilizada por Santos (2002), separando o desenvolvimento em partes, ou módulos, sendo um denominado CÂMERA que tem a função de capturar imagens e outro denominado MONITOR que tem a função de mostrar as imagens captadas em um *display*.

Santos (2002) ainda utiliza um terceiro módulo denominado REDE que, devido a várias dificuldades encontradas e também ao curto espaço de tempo para desenvolvimento, não está sendo utilizado neste trabalho, sendo que esta foi substituída por uma simples comunicação serial entre dois pontos.

O primeiro módulo a ser desenvolvido foi o MONITOR que manteve as mesmas características do *hardware* utilizado por Santos (2002), porém com um *software* totalmente distinto e escrito na linguagem de programação PICBASIC.

O módulo CAMERA tanto o *hardware* quanto o *software* utilizado por Santos (2002) foram totalmente reformulados. No *hardware* a principal modificação foi a utilização do microcontrolador AT89C55WD (família 8051) em substituição ao microcontrolador PIC16F877. Pois, conforme mostrado no capítulo 2.3, o microcontrolador AT89C55WD apresenta vantagens em relação ao microcontrolador PIC16F877 para esta aplicação em específico. Para este módulo foi utilizado a linguagem de programação C, mais especificamente, o compilador C51.

3.1 REQUISITOS PRINCIPAIS DO PROBLEMA A SER TRABALHADO

O que fica mais evidente nas modificações do protótipo desenvolvido por Santos (2002) é a adição de um banco de memória que tem a função de armazenar os dados capturados pela câmera para posteriormente enviar estes dados para serem mostrados no *display*. Um dos grandes problemas encontrados por Santos (2002) foi de equalizar a velocidade de transmissão entre a câmera e o display, uma vez que a câmera utilizada neste protótipo envia os dados capturados em uma velocidade muito superior do que a velocidade com que o display consegue ler estes dados causando erro no envio dos *frames*.

Resumidamente, a câmera capta o *frame* que é armazenado na memória de onde é lido para ser convertido (esta conversão é descrita no capítulo 3.3.1) e somente depois disto ele é

enviado para o *display* que somente irá receber um segundo *frame* após o total término da leitura do *frame* atual. A figura 10 mostra o diagrama funcional representando graficamente o protótipo.

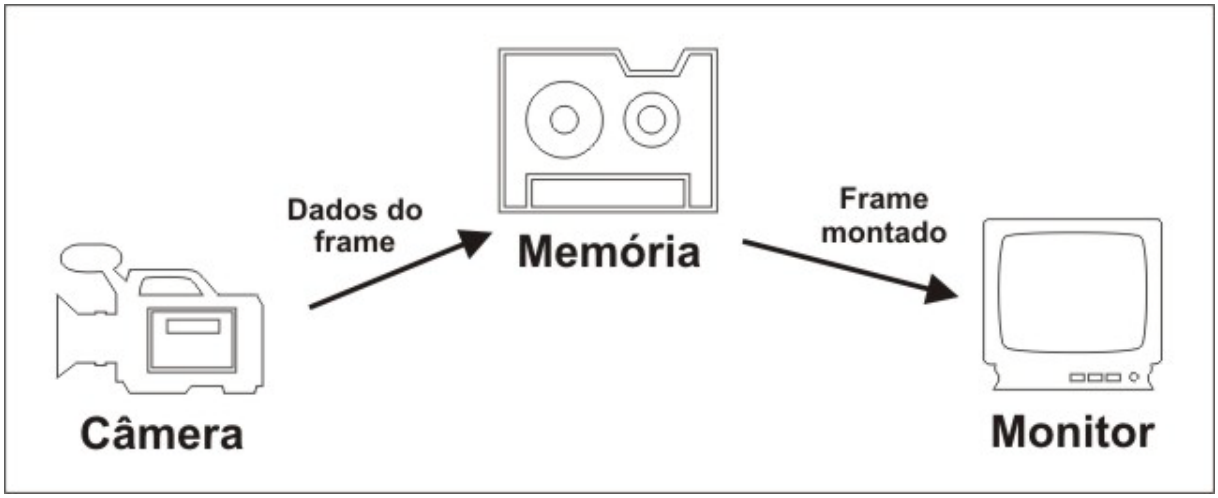


Figura 10 – Diagrama funcional do protótipo.

3.2 MONITOR

Como já mostrado anteriormente, o *hardware* deste módulo mantém as mesmas características do protótipo desenvolvido por Santos (2002), por este motivo somente serão mostradas a seguir as modificações aplicadas ao software deste módulo, porém é importante salientar que existe um erro no esquema elétrico desenvolvido por Santos (2002) que coloca o PORTC.7 (RX) no pino 25 e o PORTC.6 (TX) no pino 26, quando o correto é a disposição contrária, ou seja, PORTC.6 (TX) no pino 25 e PORTC.7 (RX) no pino 26. Como aclarado anteriormente este erro foi detectado somente no esquema elétrico, pois a montagem foi realizada de modo correto.

3.2.1 ESPECIFICAÇÃO

A primeira preocupação neste módulo é a correta configuração do *display* mantendo sempre a preocupação em equalizar o tamanho da imagem mostrada no *display* com o tamanho da imagem enviada pela câmera para que não ocorram erros na exibição destas imagens. Tanto em Santos (2002), quando neste trabalho, o tamanho da imagem enviada pela câmera e, conseqüentemente, o tamanho da imagem recebida pelo *display* é de 144 colunas por 72 linhas, totalizando 1296 bytes.

O limite horizontal do *display* pode ser configurado via software conforme pode ser visto no apêndice A. Neste mesmo apêndice também pode ser visto que o limite vertical do display tem que ser controlado através de um *flag* com um valor máximo de 1296 que é número de máximo de bytes recebido pela câmera, conforme descrito no parágrafo anterior.

Para evitar a aparição de *pixels* perdidos (comumente chamado de "lixo") no *display*, sempre é feito uma limpeza no *display* escrevendo bits em branco (valor 0) antes de escrever um *frame* vindo da câmera. Uma particularidade que foi descoberta neste *display* é que não basta reposicionar o cursor para começar a escrever na posição inicial do *display* após a escrita de um frame, é necessário refazer toda a configuração do *display.* A figura 11 mostra um fluxograma da configuração do *display* e do funcionamento do software do módulo Monitor.

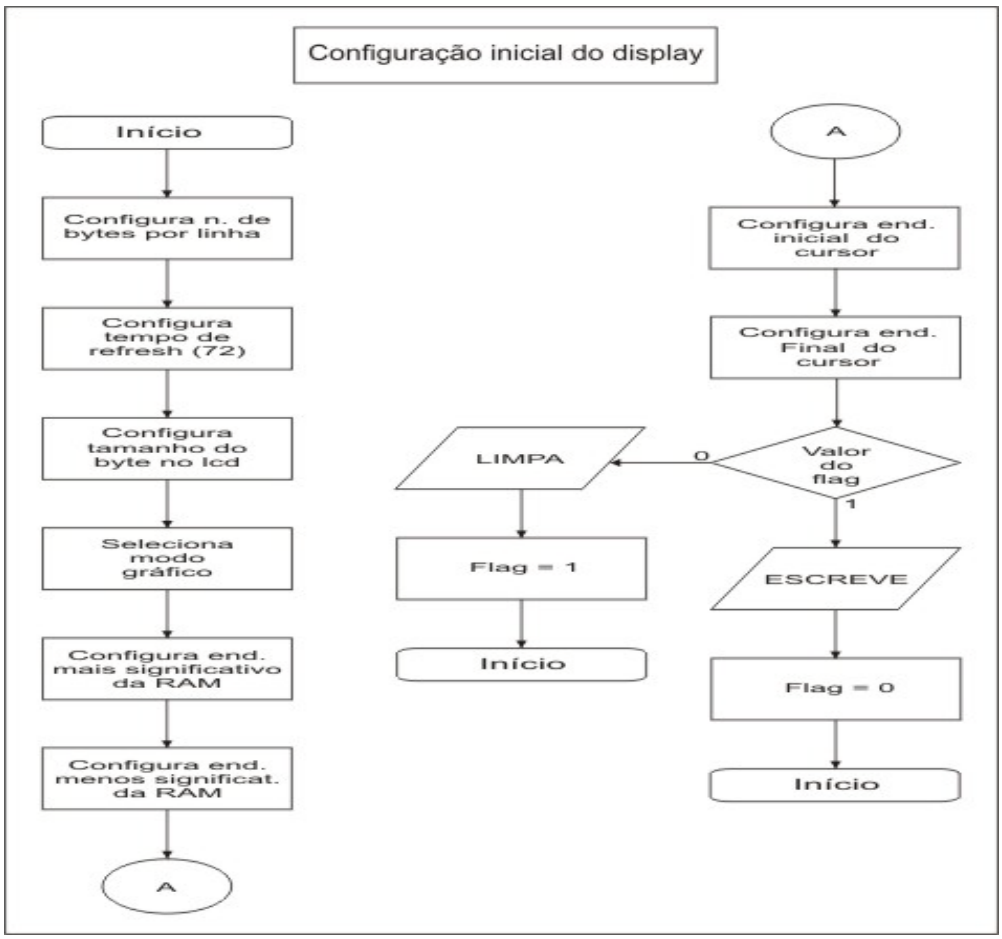


Figura 11 – Fluxograma da configuração e operação do *display*.

3.2.2 IMPLEMENTAÇÃO

O *software* que faz toda a configuração e funcionamento do *display* possui muitas linhas de código o que dificulta a exposição do mesmo em um quadro neste trabalho, por este

motivo o quadro 1 mostra a rotina ESCREVE do *software* gravado no microcontrolador e o apêndice A mostra o *software* por completo. Note que o dado que é enviado para o *display*, é o dado que foi lido da porta serial (comando SERIN2). O contador CONTA que vai de 0 a 1295 é o que controle o limite vertical do *display*.

```
ESCREVE: FOR CONTA = 0 TO 1295
          PORTB.2 = 0
          PORTB.1 = 1
          PORTD.7 = 0
          PORTD.6 = 0
          PORTD.5 = 0
          PORTD.4 = 0
          PORTD.3 = 1
          PORTD.2 = 1
          PORTD.1 = 0
          PORTD.0 = 0
          GOSUB GRAVA1
          PORTB.2 = 0
          PORTB.1 = 0
          SERIN2 PORTC.7,32,[RECEBIDO]
          PORTD = RECEBIDO
          GOSUB GRAVA2
         NEXT CONTA
         PAUSE 800
         FLAG = 0
         GOTO INIC
```

QUADRO 1 – Rotina para ler dado da serial e escrever no *display*

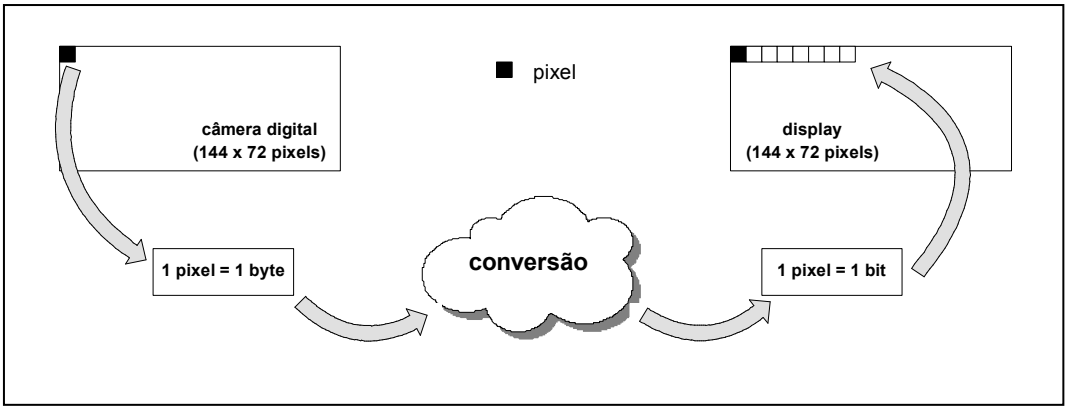
3.3 CÂMERA

Este é o módulo que mais sofreu alterações em relação ao original desenvolvido por Santos (2002), sendo que a principal delas foi a substituição do microcontrolador PIC pelo microcontrolador da família 8051, mais especificamente o modelo 89C55WD da empresa Atmel. A modificação no software ocorreu em nível de linguagem de programação que passou do *Assembly* utilizada por Santos (2002) para a linguagem de programação C, por esta última fazer parte do grupo das linguagens chamadas de alto-nível e, sendo assim, apresenta uma maior facilidade na programação.

3.3.1 ESPECIFICAÇÃO

Para não prejudicar a leitura da especificação do *hardware* deste módulo, o apêndice B contém o esquemático completo do módulo Câmera, porém uma observação importante a este esquemático deve ser ressaltada que é a ligação dos pinos de ativação de dispositivo (*chip select)* tanto da câmera quanto da memória no mesmo pino do *chip* 8051. Este tipo de ligação evita o risco de se ativar os dois dispositivos ao mesmo tempo, ou mesmo, de não ter nenhum

deles ativo, uma vez que sempre um é ativo em alta e o outro em baixa. A mesma estratégia foi utilizada na ligação dos pinos de escrita (*output enable)*. Em relação à representação do píxel lido, foi utilizado o mesmo algoritmo de Santos (2002) que é transformar o píxel lido pela câmera (com 8 bytes) em um píxel de um único bit para ser escrito no *display.* A figura 12 mostra como é feita esta conversão.



Fonte: Santos (2002, p. 57)
Figura 12 – Representação do píxel

Como a câmera em questão captura imagens em preto e branco (256 tons de cinza), basta testar o valor do *pixel* lido. Se ele for menor que 127 o *pixel* exibido no *display* será apagado (valor 0), se for maior do que 127 o *pixel* exibido no *display* será aceso (valor 1), ou seja, quanto menos luz houver o bit será apagado, em contra partida, quanto mais luz houver o bit será aceso.

3.3.2 IMPLEMENTAÇÃO

Também com a intenção de facilitar a leitura, o fluxograma explicando o funcionamento do módulo Câmera e seu *software* estão expostos, nesta seqüência, no apêndice C e D.

3.4 TÉCNICAS E FERRAMENTAS UTILIZADAS

Neste capítulo existe uma apresentação das ferramentas utilizadas no desenvolvimento do *software* e do *hardware* aplicados neste trabalho, principalmente as que não foram usadas por Santos (2002) como os ambientes de programação PICBASIC PRO e C. Em contrapartida, não serão utilizadas, e nem referenciadas, ferramentas que foram amplamente usadas por Santos (2002) como o ambiente de programação MPLAB, a ferramenta de especificação de software ABC Snapgraphics, osciloscópio e multímetro digital.

Documentações sobre o *protoboard* e sobre o gravador universal de dispositivos não foram adicionadas neste trabalho por não ter nenhuma informação adicional às informações já encontradas em Santos (2002).

3.4.1 FERRAMENTA DE ESPECIFICAÇÃO DE HARDWARE

Neste trabalho foi modificado o esquemático feito por Santos (2002) o que não exigiu ferramentas específicas para este tipo de trabalho, além do que, estes não estão disponíveis gratuitamente. Sendo assim, a edição deste esquemático ficou a cargo do CorelDraw versão 11, da Corel Corporation, que, embora não seja um software destinado a essa aplicação se mostrou bastante eficiente. Na figura 13 está uma tela do CorelDraw 11 com uma parte do esquemático que é bastante simples.

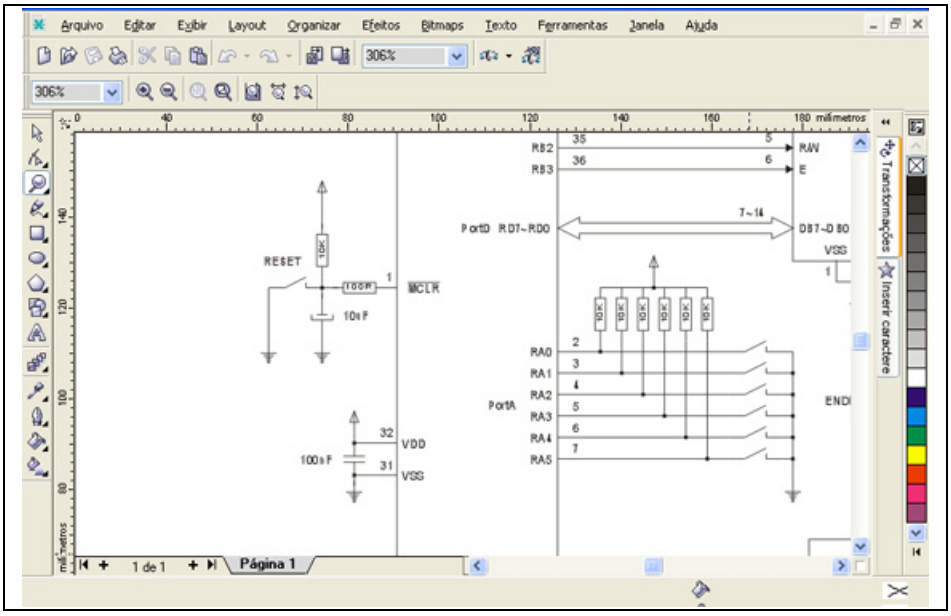


Figura 13 – Tela do CorelDraw 11

3.4.2 FERRAMENTAS DE IMPLEMENTAÇÃO

As ferramentas de implementação de hardware e de implementação de software acabam muitas vezes se fundindo num mesmo assunto. Serão, portanto, apresentadas em subtítulos deste capítulo.

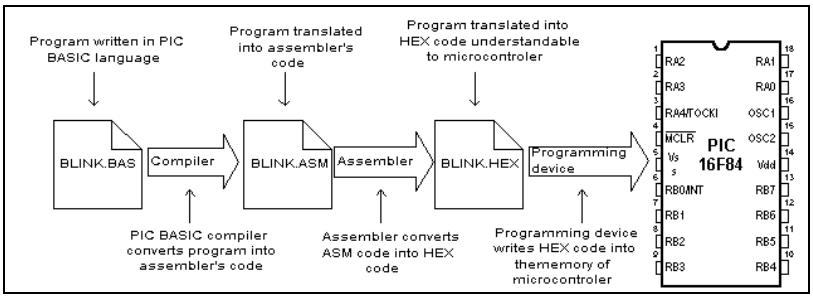
3.4.3 AMBIENTE DE DESENVOLVIMENTO

Conforme já aclarado, Santos (2002) utilizou-se do ambiente de desenvolvimento MPLAB por escolher a linguagem de programação *assembly*. Neste trabalho, porém, foi escolhida para fazer as modificações de *sofware*, a linguagem de programação PIC BASIC e a linguagem C51, ambas voltadas especificamente para o trabalho com microcontroladores.

3.4.3.1 LINGUAGEM DE PROGRAMAÇÃO PICBASIC

Segundo Mikroelektronika (2003), o PICBASIC ainda guarda muitas características da linguagem de programação BASIC que foi umas das primeiras linguagens de programação aplicadas para computadores pessoais.

O compilador BASIC é um programa que é executado em um computador e tem como função traduzir um código original BASIC em linguagem de máquina (0 e 1) entendida pelo microcontrolador. O processo de tradução de um programa em BASIC para um programa executável com extensão HEX é mostrado na figura 14. O programa escrito em PIC BASIC e gravado com extensão .BAS (qualquernome.bas) é convertido em código *assembly* (qualquernome.asm). Então, obtido o código *assembly* é traduzido para o executável .HEX (qualquernome.hex) e poderá ser gravado no PIC pelo *programming device,* ou dispositivo de gravação.



Fonte: Mikroelektronika(2003)

Figura 14 : tradução de um código basic em *assembly*

Sendo assim, foi escolhido o ambiente de programação PIC BASIC PRO que é um software que trabalha no modo console, ou seja, não possui um editor de texto onde se possa digitar o código. Este código, preferencialmente, deve ser editado em algum software de editor de texto que trabalhe somente com o modo TXT, ou modo texto simples como o EDIT (*prompt* de comando do *Windows*) ou o bloco de notas (*notepad*) do ambiente *Windows*, para que não ocorram erros no compilador PIC BASIC PRO ocasionados por marcas ou macros automáticas de editores de texto mais sofisticados como o Microsoft Word.

Uma característica em comum entre o PIC BASIC e o compilador C51 (a ser mostrado no próximo capítulo) é que ambos permitem a adição de uma rotina em *assembly* dentro do programa. Para isso basta adicionar os comandos ASM no início da rotina *assembly* e ENDASM no final da rotina. O quadro 2 mostra um exemplo destes comandos que possuem a mesma sintaxe tanto no PIC BASIC quanto no C51.

```
ASM
   BSF PORTA,0
   BCF PORTB,0
ENDASM
```

FONTE: Frino (2002)
QUADRO 2 – comandos ASM e ENDASM

O PIC BASIC PRO roda no *prompt* de comando do sistema operacional *Windows* e para compilar um programa devem ser seguidas as seguintes etapas:

a) gravar o código feito em qualquer editor de texto com a extensão .BAS (exemplo: qualquernome.bas);
b) gravar este arquivo no mesmo diretório onde se encontra o PIC BASIC PRO, normalmente c:\picbasic;
c) digitar PBP mais o nome do arquivo sem a extensão para disparar o compilador que trará mensagens de erro ou voltará ao *prompt* do DOS caso o programa não contenha nenhum erro. Após o comando PBP podem ser aplicadas algumas diretivas. Na tabela 7 podem ser vistas duas diretivas que são diretamente relacionadas com este trabalho.

Tabela 7 – Diretivas de compilação do PIC BASIC PRO

Diretiva	Exemplo	Descrição
A	PBP –a qualquernome	usa um compilador assembler diferente do padrão utilizando pelo PIC BASIC PRO
P	PBP –P 16F877 qualquernome	Seleciona que o programa *qualquernome* será convertido para um microcontrolador PIC modelo 16F877

Fonte : Adaptado de Frino (2002).

Em Frino (2002) existe um manual completo do PIC BASIC e vários fontes exemplos. Como a linguagem de programação PICBASIC pode ser utilizada somente em microcontroladores PIC, foi utilizada somente no módulo onde se encontra o display e que por sua vez, utiliza o microcontrolador PIC.

3.4.3.2 LINGUAGEM DE PROGRAMAÇÃO C51

Como no PICBASIC, a linguagem de programação entendida pelo compilador C51 possui todas as características das primeiras versões da linguagem C que é amplamente utilizada em programação de computadores e que hoje em dia já possui diferentes versões com o C++ e outras. (KEIL, 2000)

A versão do compilador C51 utilizada neste trabalho é direcionada para o uso em todos os microcontroladores da família 8051 e pode trabalhar no *prompt* de comando de qualquer versão do sistema operacional Windows de 32 bits, porém já existem versões mais modernas deste compilador que já executam dentro das janelas do sistema operacional Windows. O compilador C51 traz novas bibliotecas em relação ao compilador C padrão, além de modificar algumas já existentes, para facilitar a programação direcionada a microcontroladores e também para suprir algumas dificuldades que o compilador C padrão possui para executar algumas funções como, por exemplo, operações de I/O (KEIL, 2002).

Este compilador trabalha da mesma maneira que o PICBASIC, ou seja, pode-se utilizar qualquer editor de texto para digitar o código do programa na linguagem C. Este texto deverá possuir a extensão .C e ser gravado no mesmo diretório onde se encontra o C51 para que o programa possa ser encontrado pelo compilador, como por exemplo, qualquernome.c.

Para se compilar um programa e gerar um arquivo com código de máquina (extenção .hex) que é o arquivo que deve ser gravado no microcontrolador, são necessários 3 etapas que

estão descritas na tabela 8, lembrando que os comandos expostos nesta tabela devem ser digitados no *prompt* de comando :

Tabela 8 – passos para compilar um programa no C51

Comando	Exemplo	Resultado
C51	C51 qualquernome.c	Compila o programa e gera o arquivo qualquernome.obj
L51	L51qualquernome.obj	Adiciona as bibliotecas do C51
OHS51	OHS51 qualquernome	Gera arquivo com extensão HEX

O compilador C51 utiliza várias bibliotecas específicas e necessárias para executar qualquer *software* desenvolvido neste compilador. Entre estas, existe a biblioteca chamada REG51.H que merece atenção especial por conter os rótulos utilizados durante a escrita do programa como, por exemplo, rotular o pino 13 do *chip* 8051 (neste trabalho ligado ao pino 20 da câmera - Vsync) para poder escrever um dado neste pino somente escrevendo Vsync = 1. No apêndice E está esta biblioteca completa com todas as modificações realizadas para executar o *software* do módulo CÂMERA.

3.5 OPERACIONALIDADE DA IMPLEMENTAÇÃO

Este protótipo quase não possui interação com o usuário, uma vez que basta ligar os módulos (display e câmera) na energia elétrica para que o mesmo comece a trabalhar.

3.6 RESULTADOS E DISCUSSÃO

Ao final deste trabalho, os módulos CÂMERA e MONITOR apresentaram funcionamento correto, apesar de não satisfatório. O funcionamento está correto, pois foi comprovado através do uso do osciloscópio e também dos dados apresentados no *display*, que a imagem está sendo capturada pela câmera, gravada/lida no banco de memória, transmitida pela serial e visualizada no *display*.

Não satisfatória porque os dados mostrados no *display* não estão organizados, ou seja, é possível identificar modificação na luminosidade (o que prova o funcionamento do protótipo), porém não é possível identificar uma imagem completa como, por exemplo, o rosto de uma pessoa ou uma sala. Nas figuras 15 e 16 é possível identificar a diferença de luminosidade mostrada no *display*, sendo que a figura 15 apresenta o protótipo exposto a uma fonte de luz direta fazendo com que o *display* fique preenchido (bits acesos) quase que por completo e a figura 16 mostra o protótipo exposto a pouca quantidade de luz.

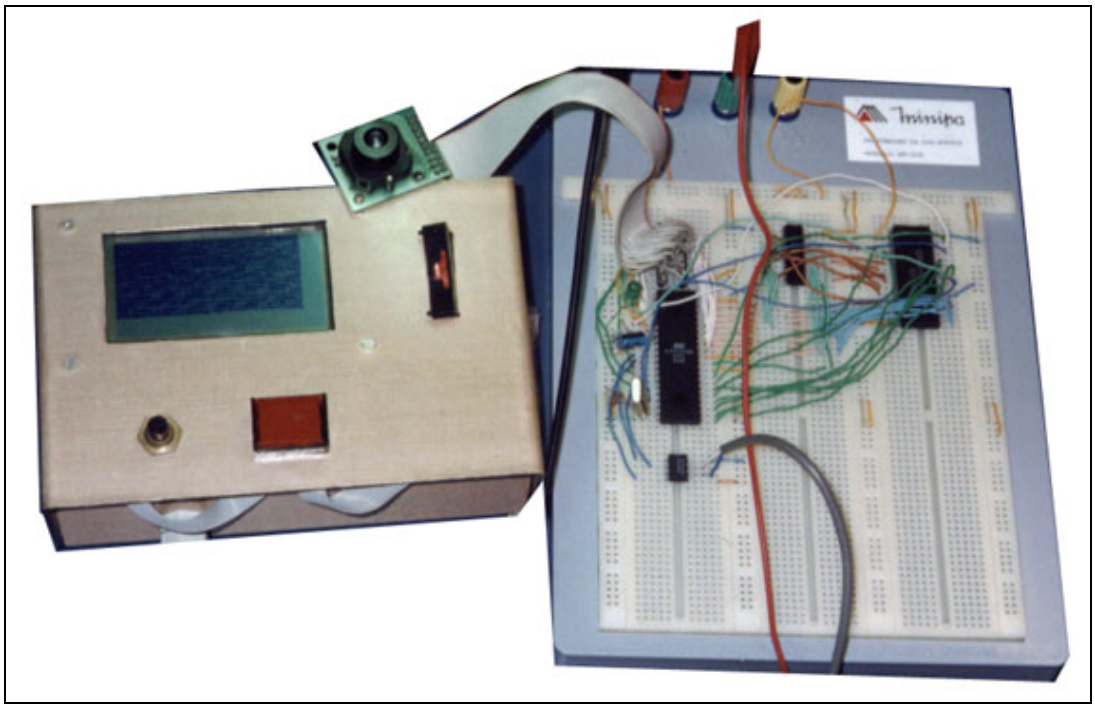


Figura 15 – Protótipo com fonte de luz direta

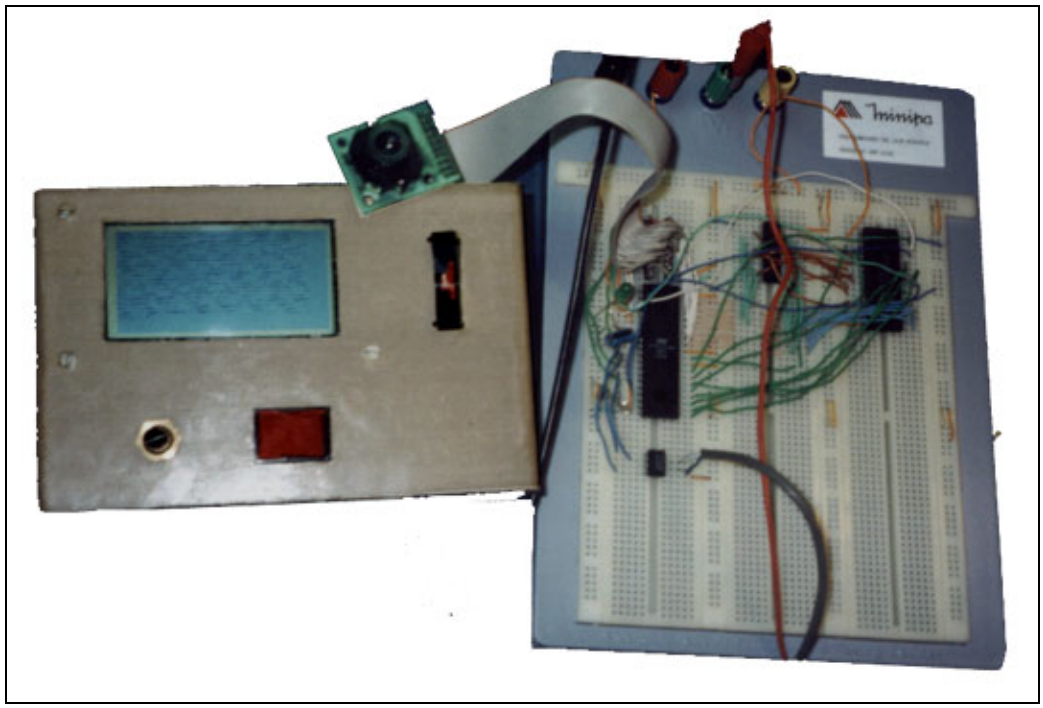


Figura 16 – Protótipo com fonte de luz indireta

Outro teste que comprovou o funcionamento do módulo MONITOR, da gravação/leitura da memória e da comunicação serial foi realizado através da alteração do *software* do módulo CÂMERA que, ao invés de gravar/ler na memória e transmitir dados coletados da câmera, passou a gravar/ler na memória e a transmitir o valor fixo 85, que em binário representa 01010101 e que no *display* irá resultar em uma seqüência de bit apagado e aceso formando colunas.

O apêndice F mostra estas modificações nas rotinas LE_QUADRO e TRANSMITE_FRAME do *software* do módulo CÂMERA, lembrando que, neste caso, todas as configurações da câmera e o algoritmo de conversão do frame se tornam desnecessários devido ao protótipo passar a trabalhar com um valor fixo.

4 CONCLUSÕES

Ao final deste trabalho, ficou claro a importância do conhecimento do ambiente onde se está interagindo (desenvolvendo o *software*). Neste trabalho, além de diferentes aparatos, como a câmera digital e o *display*, foram utilizadas duas arquiteturas distintas em dois microcontroladores igualmente distintos e versáteis, além é claro, da utilização de duas linguagens de programação.

Foi sentido a mesma dificuldade encontrada por Santos (2002) que é a pouca documentação a respeito do *display* e da câmera envolvidos neste trabalho, sendo que algumas dúvidas tiveram que ser resolvidas pela técnica de experimentação, e isto somado ao curto espaço de tempo para a finalização deste trabalho, acabou impossibilitando o total cumprimento dos objetivos previamente estipulados, porém os resultados alcançados podem ser considerados válidos pelo sucesso na implementação do banco de memória, da visualização dos dados no *display*, além da utilização de duas tecnologias distintas (PIC e 8051) e também de duas linguagens de programação (C e PICBASIC).

Como o problema da velocidade de transmissão encontrado por Santos (2002) foi resolvido com a adição do banco de memória, fica como principal causa da não visualização da imagem a correta configuração da câmera através dos controles do HREF, VSYNC e PCLK.

4.1 EXTENSÕES

Uma sugestão seria a implementação de comunicação com microcomputadores, a fim de possibilitar a interação necessária ao desenvolvimento de um sistema mais abrangente de automação residencial.

Outra sugestão seria a própria viabilização deste projeto com a visualização perfeita da imagem captada pela câmera, que poderá ser conseguida com a troca do display gráfico por um de alta definição, além de um estudo mais aprofundado da inicialização e controle da câmera através dos pinos do HREF, VSYNC e PCLK. Além disso, as taxas de transmissão ainda poderiam ser melhoradas utilizando-se de microcontroladores mais rápidos e/ou tecnologias como USB e *firewire* para a transmissão dos dados.

REFERÊNCIAS BIBLIOGRÁFICAS

ATOS. **Boletim técnico EP-03/00:** informações básicas para implantação de uma rede de controladores, São Paulo, fev. 2000. Disponível em: <http://www.atos.com.br/download/default.asp>. Acesso em: 23 jan. 2004.

AURESIDE. **Associação Brasileira de Automação Residencial**, São Paulo, jan. 2004. Disponível em: <http://www.aureside.org.br>. Acesso em: 20 jan. 2004.

CLUBE DAS REDES, **CLUBE DAS REDES.** Rio de Janeiro, [2003]. Disponível em: <http://www.clubedasredes.eti.br/hard0006.htm>. Acesso em: 15 mar. 2004.

COMEDIA. **d-m4088.pdf.** M4088 1/4" B/W camera module with digital output, [S.l.], 09 nov. 1999. Disponível em: <http://www.electronics123.com/amazon/datasheet/d-m4088.pdf>. Acesso em: 12 fev. 2004.

ELETRÔNICA SENAI. **Eletrônica Básica,** São Paulo, [2002?]. Disponível em: <http://www.eletronicasenai.hpg.ig.com.br>. Acesso em: 15 mar. 2004.

FRANCO, Lúcia Regina Horta R. **EIA RS-485 field bus**, Itajubá, nov. 2001. Disponível em: <http://www.iee.efei.br/~gaii/rs485/hp_rs485.htm >. Acesso em: 20 fev. 2004.

FRINO. **Luis Frino**, Mar Del Plata, [2002?]. Disponível em: <http://www.frino.com.ar>. Acesso em 23 jan. 2004.

KEIL Software, EUA, [2004]. Disponível em: <http://www.keil.com.>. Acesso em 23 jan. 2004.

KLITZKE, Marcelo. **Protótipo de hardware para aquisição e transmissão de imagens via padrão serial RS-485**. 1999. 60 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

MICROCHIP, Taiwan, [2004?]. Disponível em: <http://www.microchip.com>. Acesso em: 16 jan. 2004.

MIKROELEKTRONIKA. **Basic for microcontrollers.** Belgrado. [2003]. Disponível em <http://www.mikroelektronika.co.yu/english/product/books/picbasicbook>. Acesso em 10 fev. 2004.

NATIONAL SEMICONDUCTOR. **DS3695.pdf.** DS3695/DS3695T/DS3696/DS3697 Multipoint RS485/RS422 Transceivers/Repeaters, [S.l.], 06 out. 1998. Disponível em: <http://www.national.com/pf/DS/DS3696.html>. Acesso em: 16 jan. 2004.

NEC. **NEC Corporation.** Japão [2002]. Disponível em <http://www.necel.com/memory/pdfs/m11657ejbv0ds00.pdf>. Acesso em: 16 jan. 2004.

NICOLOSI, Denys Emilio Campion. **Microcontrolador 8051 detalhado**. São Paulo: Ed. Erica, 2000.

OKISEMI. **Oki Semiconductor**, Japão [2002]. Disponível em < http://www.okisemi.com/jp/datadocs/doc-eng/msm5298a.pdf>. Acesso em: 12 fev. 2004.

OMNIVISION TECHNOLOGIES. **OV5017DS.pdf.** OV5017, Sunnyvale, 03 set.. 1998. Disponível em: <http://www.electronics123.com/amazon/datasheet/OV5017DS.pdf>. Acesso em: 16 fev. 2004.

SANTOS, Ângelo Dias. **Protótipo de hardware e software para captura e visualização de imagens compartilhadas via interface digital serial diferencial balanceada**. 2002. 115 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

SANYO. **LC7981.pdf.** LCD dot matrix graphic display controller, [S.l.], [S.d.]. Disponível em: <http://www.hantronix.com/down/lc7981.pdf>. Acesso em: 16 fev. 2004.

UEM. **Universidade Estadual de Maringá.** Departamento de Informática. Maringá, [fev. 2003]. Disponível em: <http://www.din.uem.br/~edigital/latch1.htm>. Acesso em 12 fev. 2004.

UFPB. **Universidade Federal da Paraíba.** Departamento de Informática, [out. 2003]. Disponível em <http://www.di.ufpb.br/raimundo/PCaFundo/memoria/memoria.htm>. acesso em 12 fev. 2004.

APÊNDICE A – Software do módulo MONITOR

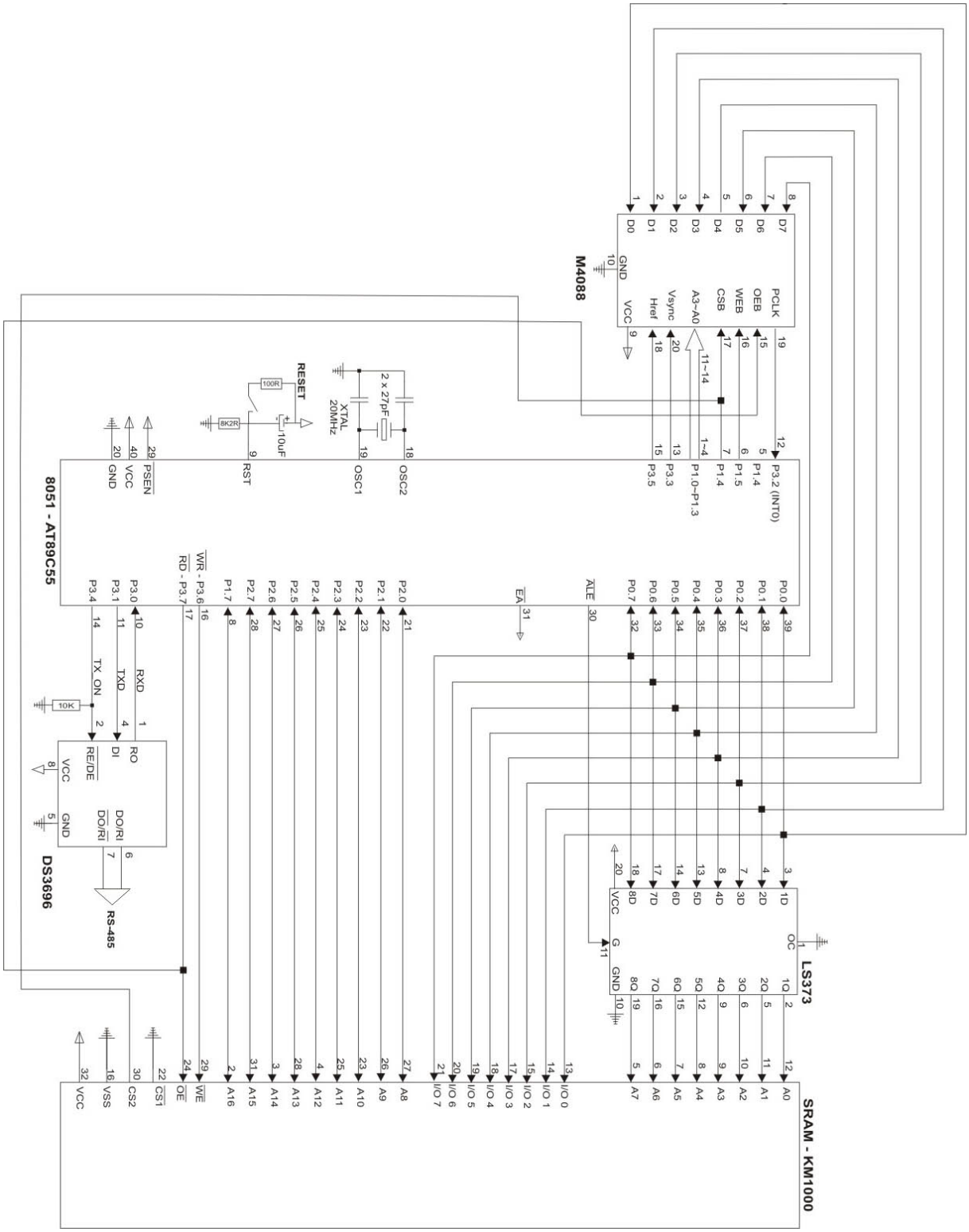
```
ASM
  device  pic16F877, hs_osc, wdt_off, pwrt_on, bod_off, lvp_off, cpd_off, wrt_off, protect_off
ENDASM                          'DIRETIVA PARA FIM DE BLOCO EM ASSEMBLY
DEFINE OSC 20                   'CRISTAL DE 20Mhz
DEFINE LCD_DREG PORTD           'SELECIONA PORTD COMO DADOS
DEFINE LCD_DBIT 0               'DADOS COME€A EM PORTD,0
DEFINE LCD_RSREG PORTB
DEFINE LCD_RSBIT 1              'SELECIONA PORTB,1 PARA RS
DEFINE LCD_RWREG PORTB
DEFINE LCD_RWBIT 2              'SELECIONA PORTB,2 PARA RW
DEFINE LCD_EREG PORTB
DEFINE LCD_EBIT 3               'SELECIONA PORTB,3 PARA E
CONTA VAR WORD                 'CONTADOR
FLAG VAR BIT                    'VARIµVEL PARA CONTROLE LIMPA ESCREVE
RECEBIDO VAR BYTE               'VARIµVEL QUE RECEBE DADO DA SERIAL
TRISB = 0                       'PORTB COMO SAIDA
TRISD = 0                       'PORTD COMO SAIDA
TRISC.7 = 0                     'PORTC.7 COMO SAIDA
TRISC.5 = 0                     'PORTC.5 COMO SAIDA
TRISC.6 = 1                     'PORTC.6 COMO ENTRADA
PAUSEUS 5                       'PAUSA DE 5 MILISEGUNDOS
PORTA = 0
PORTB = 0
PORTC = 0
PORTD = 0                       'ZERA PORTAS A,B,C,D
FLAG = 0                        'ZERA CONTADOR
PORTC.5 = 0                     'GARANTE RS369 COMO RECEPCAO

'INICIALIZA€ÇO DO LCD DE ACORDO COM O MANUAL DO CONTROLADOR
        'CONFIGURA€ÇO QUANTIDADE DE PONTOS POR LINHA
         'SELECIONA REGISTRO DE INSTRU€ÇO

INIC:   PORTB.2 = 0                      'RW = 0
        PORTB.1 = 1                      'RS = 1
        PORTD.7 = 0
        PORTD.6 = 0
        PORTD.5 = 0
        PORTD.4 = 0
        PORTD.3 = 0
        PORTD.2 = 0
        PORTD.1 = 1
        PORTD.0 = 0                     'ENDERE€AMENTO DO PORTD

        GOSUB GRAVA1

         'REGISTO DO NéMERO DE CARACTER
        PORTB.2 = 0
        PORTB.1 = 0
        PORTD = 17                      'LIMITE DE 18 BYTES NA HORIZONTAL

        GOSUB GRAVA2

         'CONFIGURA€ÇO REFRESH
         'SELECIONA REGISTRO DE INSTRU€ÇO
        PORTB.2 = 0
        PORTB.1 = 1
        PORTD.7 = 0
        PORTD.6 = 0
        PORTD.5 = 0
        PORTD.4 = 0
        PORTD.3 = 0
        PORTD.2 = 0
        PORTD.1 = 1
        PORTD.0 = 1                     'ENDERE€AMENTO DO PORTD

        GOSUB GRAVA1

          'SELE€ÇO DO TEMPO DE REFRESH
        PORTB.2 = 0
        PORTB.1 = 0
        PORTD = 79

        GOSUB GRAVA2
```

```
'CONFIGURAEÇO QUANTOS BIT POR BYTE NO LCD (8)
 'SELECIONA REGISTRO DE INSTRUEÇO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0
PORTD.4 = 0
PORTD.3 = 0
PORTD.2 = 0
PORTD.1 = 0
PORTD.0 = 1

GOSUB GRAVA1

 'SELECIONA 8 BITS POR BYTE
PORTB.2 = 0                        'RW = 0
PORTB.1 = 0                        'RS = 0
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0
PORTD.4 = 0
PORTD.3 = 0
PORTD.2 = 1
PORTD.1 = 1
PORTD.0 = 1                        'ENDEREEAMENTO DO PORTD

GOSUB GRAVA2

'CONFIGURAEÇO MODO GRµFICO
 'SELECIONA REGISTRO DE INSTRUEÇO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD   = 0                         'ENDEREEAMENTO DO PORTD

GOSUB GRAVA1

 'SELECIONA MODO GRµFICO
PORTB.2 = 0                       'RW = 0
PORTB.1 = 0                       'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 1
PORTD.4 = 1
PORTD.3 = 0
PORTD.2 = 0
PORTD.1 = 1
PORTD.0 = 0                       'ENDEREEAMENTO DO PORTD

GOSUB GRAVA2

'CONFIGURAEÇO ENDEREEO LOWER DA RAM
 'SELECIONA REGISTRO DE INSTRUEÇO
 'CONFIGURA NOVAMENTE PARA ZERAR O CONTADOR DA RAM E COMEEAR
 'A ESCREVER NO DISPLAY
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0
PORTD.4 = 0
PORTD.3 = 1
PORTD.2 = 0
PORTD.1 = 1
PORTD.0 = 0                         'ENDEREEAMENTO DO PORTD

GOSUB GRAVA1

 'SELECIONA ENDEREEO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 0                         'RS = 0
PORTD = 0

GOSUB GRAVA2
```

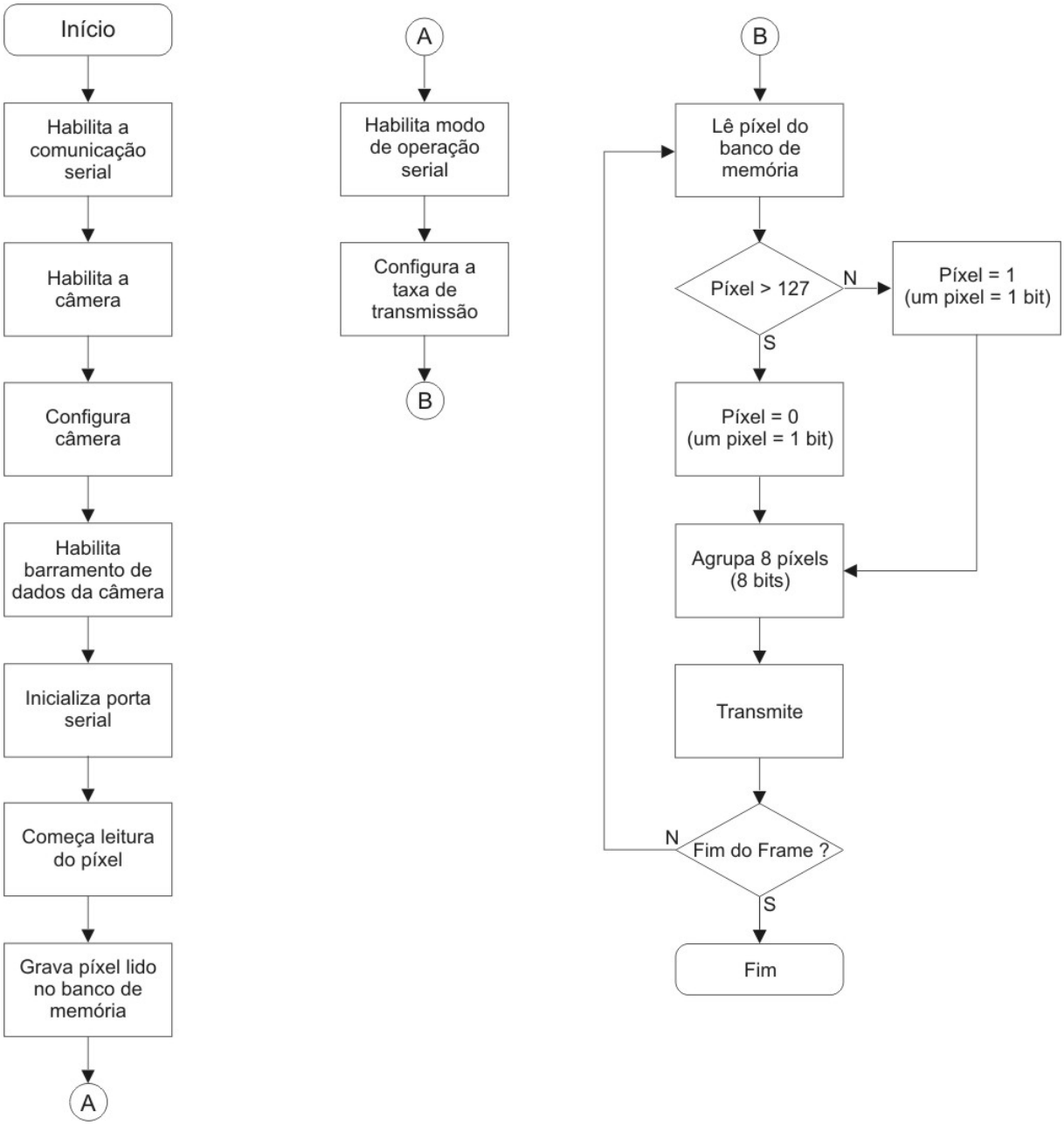
```
'CONFIGURAEÇO ENDEREEO UPPER DA RAM
 'SELECIONA REGISTRO DE INSTRUEÇO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0
PORTD.4 = 0
PORTD.3 = 1
PORTD.2 = 0
PORTD.1 = 1
PORTD.0 = 1                         'ENDEREEAMENTO DO PORTD

GOSUB GRAVA1

 'SELECIONA ENDEREEO
PORTB.2 = 0                        'RW = 0
PORTB.1 = 0                        'RS = 0
PORTD = 0                    'ENDEREEAMENTO DO PORTD

GOSUB GRAVA2

 'CONFIGURAEÇO ENDEREEO START LOWER
 'SELECIONA REGISTRO DE INSTRUEÇO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0

PORTD.4 = 0
PORTD.3 = 1
PORTD.2 = 0
PORTD.1 = 0
PORTD.0 = 0                         'ENDEREEAMENTO DO PORTD

GOSUB GRAVA1

 'SELECIONA ENDEREEO
PORTB.2 = 0                        'RW = 0
PORTB.1 = 0                        'RS = 0
PORTD = 0

GOSUB GRAVA2


 'CONFIGURAEÇO ENDEREEO START UPER
 'SELECIONA REGISTRO DE INSTRUEÇO
PORTB.2 = 0                         'RW = 0
PORTB.1 = 1                         'RS = 1
PORTD.7 = 0
PORTD.6 = 0
PORTD.5 = 0
PORTD.4 = 0
PORTD.3 = 1
PORTD.2 = 0
PORTD.1 = 0
PORTD.0 = 1                         'ENDEREEAMENTO DO PORTD

GOSUB GRAVA1

 'SELECIONA ENDEREEO
PORTB.2 = 0                        'RW = 0
PORTB.1 = 0                        'RS = 0
PORTD = 0                    'ENDEREEAMENTO DO PORTD

GOSUB GRAVA2


IF FLAG = 0 THEN
 GOTO LIMPA
ELSE
 GOTO ESCREVE
ENDIF   'TESTE CONTROLE SE FOR 0 LIMPA DISPLAY SENÇO ESCREVE
```

```
         'ROTINA PARA LIMPAR O LCD ESCREVENDO 00000000
         'SELECIONA REGISTRO DA RAM

LIMPA:   FOR CONTA = 0 TO 1440  'PARA LIMPAR SÇO 18 BYTES POR 80 LINHAS = 1440 BYTES
          PORTB.2 = 0
          PORTB.1 = 1
          PORTD.7 = 0
          PORTD.6 = 0
          PORTD.5 = 0
          PORTD.4 = 0
          PORTD.3 = 1
          PORTD.2 = 1
          PORTD.1 = 0
          PORTD.0 = 0

          GOSUB GRAVA1

          'MODO DO REGISTRO DE CONTROLE - JOGA A NO ENDERE€O 00
          PORTB.2 = 0
          PORTB.1 = 0
          PORTD = 0

          GOSUB GRAVA2
         NEXT CONTA
         FLAG=1
         GOTO INIC

         'ROTINA PARA ESCREVER NO LCD
         'SELECIONA REGISTRO DA RAM

ESCREVE: FOR CONTA = 0 TO 1295 'ESCREVE EM 18 BYTES POR 72 LINHAS (1296)
          PORTB.2 = 0
          PORTB.1 = 1
          PORTD.7 = 0
          PORTD.6 = 0
          PORTD.5 = 0
          PORTD.4 = 0
          PORTD.3 = 1
          PORTD.2 = 1
          PORTD.1 = 0
          PORTD.0 = 0

          GOSUB GRAVA1

         'MODO DO REGISTRO DE CONTROLE - ESCREVE O VALOR DE RECEBIDO NO DISPLAY
          PORTB.2 = 0
          PORTB.1 = 0
          SERIN2 PORTC.7,84,[RECEBIDO]   ' LÒ SERIAL E GRAVA NA VARIµVEL RECEBIDO
          PORTD = RECEBIDO               ' ESCREVE O VALOR DE RECEBIDO NA SERIAL

          GOSUB GRAVA2

         NEXT CONTA
         PAUSE 800  'PAUSA PARA VISUALIZA€ÇO DA TELA CHEIA ANTES DE LIMPAR
         FLAG = 0
         GOTO INIC


GRAVA1: PORTB.2 = 0                     'SETA RW
        PORTB.1 = 1                     'APAGA RS
        PORTB.3 = 1                     'SETA/APAGA E
        PAUSEUS 5
        PORTB.3 = 0
        PAUSEUS 5                       'PAUSE DE 5 MILISEGUNDOS
        RETURN

GRAVA2: PORTB.2 = 0                     'APAGA RW
        PORTB.1 = 0                     'APAGA RS
        PORTB.3 = 1                     'SETA/APAGA E
        PAUSEUS 5                       'PAUSE DE 5 MILISEGUNDOS
        PORTB.3 = 0
        PAUSEUS 5
        RETURN
```

APÊNDICE B – Esquema Elétrico do módulo CÂMERA



APÊNDICE C – Fluxograma do módulo CÂMERA



APÊNDICE D – Software do módulo CÂMERA

```
/*Programa do módulo CAMERA*/
/*Jorge de Assis Merege Neto*/

#include <reg51.h>
#include <intrins.h>
#define count 250
#define eot   04

unsigned int x, y;
unsigned char xdata dado, memoria[64000];
unsigned int pointer = 0; /* variavel para guardar posicao de gravacao da memoria */
unsigned int tamanhoframe = 0;

void transmite_frame() /* função para rotacionar byte e ligar se menor que 127 ou apagar se
maior que 128 */
{
unsigned char temp, dadosaida;
unsigned int i,j;
unsigned int contador;

CSB = 1;
TI = 0;
for (j=0;j<tamanhoframe;j=j+8)
  {

   dadosaida = 0;
   for (i=j;i<(j+8);i++)   /*conta de 8 em 8. Converte frame de 10368 para 1296 bytes */
   {
    temp = memoria[i];
    if (temp < 128)
       dadosaida = dadosaida << 1;
    else
       dadosaida = (dadosaida << 1)+1;  /*quanto mais luz, acende o bit*/
   };

   SBUF = dadosaida;
   while (TI==0);
   TI = 0;
   for(contador=0; contador <50 ; contador++);

  }
}

void le_quadro()
{
   char temp;
   char le_frame;
   temp = 0;
   pointer = 0;
   le_frame = 1;
   P17 = 0;
   while (VSYNC == 0);
   while (VSYNC == 1);
   while (le_frame == 1)
   {
       while (HREF == 0);
       while (HREF == 1)
       {
         while (PCLK == 1);
         CSB = 0;
         temp = dado;
         CSB = 1;
         while (PCLK == 0);
         memoria[pointer] = temp;
         pointer++;
       }
    if( pointer > 10366 )
       le_frame = 0;
   }
   tamanhoframe = pointer;
   IE = 0;
```

```
/*        ; CONFIGURA MODO DE OPERACAO DO CANAL SERIAL      */

        PCON = PCON | 0x80;
        TR0 = 0;
        TR1 = 0;

/*        ; CONFIGURA TEMP/CONT E TAXA DE TRANSMISSAO  */

        TMOD = 0x20;
        TH1 = count;
        TL1 = count;
        TR1 = 1;
        SCON = 0x40;
       transmite_frame();
    P17 = 1;
    while (1);
}

void sfr1()        /* habilita modo de frame unico (single frame) */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador FCTL = 0001 */
  A2  = 0; /* endereco do registrador FCTL = 0001 */
  A1  = 0; /* endereco do registrador FCTL = 0001 */
  A0  = 1; /* endereco do registrador FCTL = 0001 */
  P0  = 0x40; /* habilita modo frame unico jogando 01000000 no port p0 */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void pcks() /* habilita PCLK apenas para pixel valido */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador MCTL = 0101 */
  A2  = 1; /* endereco do registrador MCTL = 0101 */
  A1  = 0; /* endereco do registrador MCTL = 0101 */
  A0  = 1; /* endereco do registrador MCTL = 0101 */
  P0  = 0x4; /* habilita PCKS jogando 00000100 no port p0 */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void fps() /* habilita taxa de varredura da camera em 0,5 frames por segundo */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador FRCTL = 0100 */
  A2  = 1; /* endereco do registrador FRCTL = 0100 */
  A1  = 0; /* endereco do registrador FRCTL = 0100 */
  A0  = 0; /* endereco do registrador FRCTL = 0100 */
  P0  = 0x3F; /* habilita taxa de varredura da camera jogando xx111111 (0,5 fps) no port p0 */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void hwctl() /* delimita tamanho da imagem na horizontal */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador HWCTL = 0110 */
  A2  = 1; /* endereco do registrador HWCTL = 0110 */
  A1  = 1; /* endereco do registrador HWCTL = 0110 */
  A0  = 0; /* endereco do registrador HWCTL = 0110 */
  P0  = 0x5A; /* inicio (bloco 5) e final (bloco 10) do frame na horizontal */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void vwctl() /* delimita tamanho da imagem na vertical */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador VWCTL = 0111 */
```

```
  A2  = 1; /* endereco do registrador VWCTL = 0111 */
  A1  = 1; /* endereco do registrador VWCTL = 0111 */
  A0  = 1; /* endereco do registrador VWCTL = 0111 */
  P0  = 0x69; /* inicio (bloco 6) e final (bloco 9) do frame na vertical */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void fset()  /* dispara single frame - inicia trabalho de captacao de imagem */
 {
  unsigned char i;
  WEB = 0;
  A3  = 0; /* endereco do registrador FCTL = 0001 */
  A2  = 0; /* endereco do registrador FCTL = 0001 */
  A1  = 0; /* endereco do registrador FCTL = 0001 */
  A0  = 1; /* endereco do registrador FCTL = 0001 */
  P0  = 0xC0; /* inicia captacao de imagem */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

void vport() /* configuracao para leitura de dados da camera */
 {
  unsigned char i;
  WEB = 0;
  A3  = 1; /* endereco do registrador VPORT = 10xxxxxx */
  A2  = 0; /* endereco do registrador VPORT = 10xxxxxx */
  for(i=0; i<=10;i++); /* loop para estabilizacao dos dados no barramento */
  WEB = 1;
 }

/************* fim na inicializacao da camera ****************************/

/* inicializa porta serial */

void serial_init()
 {
  SCON  = 0x50;                     /* mode 1: 8-bit UART, enable receiver   */
  TMOD |= 0x20;                     /* timer 1 mode 2: 8-Bit reload          */
  TH1   = 0xfd;                     /* reload value  baud 9600                 */
                                    /* valor de carga para 171.875 será de 255 com um cristal
de 33mhz */
  TR1   = 1;                        /* timer 1 run                           */
  ES    = 1;                        /* enable serial port interrupt          */
 }

            /* inicio do programa principal */
main()
{
  unsigned int i;
  P17 = 1;
  for (i=0; i < 8000; i++);
  P17 = 0;
  for (i=0; i < 8000; i++);
  P17 = 1;
  for (i=0; i < 8000; i++);
  P17 = 0;
  for (i=0; i < 8000; i++);  /*pisca o LED para confirmar que o programa está executando */
  P17 = 1;

  TX_ON = 1; /* HABILITA SERIAL PARA TRANSMISSAO
  CSB = 0;   /* habilita dispositivo camera */
  sfr1();
  pcks();
  fps();
  hwctl();
  vwctl();
  fset();
  vport();

/*  IE = 0x81;  joga 10000001 no ie. Habilita global enable e external enable */
  le_quadro();
  for (i=0; i < 8000; i++);
  P17 = 0;
  for (i=0; i < 8000; i++);
```

```
  P17 = 1;
  for (i=0; i < 8000; i++);
  P17 = 0;
  for (i=0; i < 8000; i++);
  P17 = 1;
  while (1);

}
```

APÊNDICE E – Biblioteca REG51.H

```
/*
 *-----------------------------------------------------------------------------
 *  Copyright (c) KEIL ELEKTRONIK GmbH and Franklin Software, Inc., 1987-1992
 *-----------------------------------------------------------------------------
 */
/*  Register Declarations for 8051 Processor */

/*  BYTE Register  */
sfr P0   = 0x80;
sfr P1   = 0x90;
sfr P2   = 0xA0;
sfr P3   = 0xB0;
sfr PSW  = 0xD0;
sfr ACC  = 0xE0;
sfr B    = 0xF0;
sfr SP   = 0x81;
sfr DPL  = 0x82;
sfr DPH  = 0x83;
sfr PCON = 0x87;
sfr TCON = 0x88;
sfr TMOD = 0x89;
sfr TL0  = 0x8A;
sfr TL1  = 0x8B;
sfr TH0  = 0x8C;
sfr TH1  = 0x8D;
sfr IE   = 0xA8;
sfr IP   = 0xB8;
sfr SCON = 0x98;
sfr SBUF = 0x99;


/*  BIT Register  */
/*  PSW   */
sbit CY   = 0xD7;
sbit AC   = 0xD6;
sbit F0   = 0xD5;
sbit RS1  = 0xD4;
sbit RS0  = 0xD3;
sbit OV   = 0xD2;
sbit P    = 0xD0;
sbit RW   = 0xB2;
sbit RS   = 0xB3;
sbit P00  = 0x80;
sbit P27  = 0xA7;
sbit P10  = 0x90;
sbit P11  = 0x91;
sbit P12  = 0x92;
sbit P13  = 0x93;
sbit P17 =  0x97;
sbit P35  = 0xB5;
sbit P32  = 0xB2;
sbit P33  = 0xB3;

sbit VSYNC = 0xB3;  /* sincronismo vertical */
sbit HREF = 0xB5;   /* sincronismo horizontal */
sbit PCLK = 0xB2;   /* clock de pixel */

sbit CSB  = 0x96;  /*seta endereço do registrador CSB da camera - ligado no P1.6 do Atmel*/
sbit OEB  = 0x94;  /*seta endereço do registrador OEB da camera - ligado no P1.4 do Atmel*/
sbit WEB  = 0x95;  /*seta endereço do registrador WEB da camera - ligado no P1.5 do Atmel*/
sbit A0   = 0x90;  /*seta pinoA0 de configuração da camera - ligado no P1.0 do Atmel*/
sbit A1   = 0x91;  /*seta pinoA1 de configuração da camera - ligado no P1.1 do Atmel*/
sbit A2   = 0x92;  /*seta pinoA2 de configuração da camera - ligado no P1.2 do Atmel*/
sbit A3   = 0x93;  /*seta pinoA3 de configuração da camera - ligado no P1.3 do Atmel*/

/*  TCON  */
sbit TF1  = 0x8F;
sbit TR1  = 0x8E;
sbit TF0  = 0x8D;
sbit TR0  = 0x8C;
sbit IE1  = 0x8B;
sbit IT1  = 0x8A;
```

```
sbit IE0  = 0x89;
sbit IT0  = 0x88;

/*  IE   */
sbit EA   = 0xAF;
sbit ES   = 0xAC;
sbit ET1  = 0xAB;
sbit EX1  = 0xAA;
sbit ET0  = 0xA9;
sbit EX0  = 0xA8;

/*  IP   */
sbit PS   = 0xBC;
sbit PT1  = 0xBB;
sbit PX1  = 0xBA;
sbit PT0  = 0xB9;
sbit PX0  = 0xB8;

/*  P3  */
sbit RD   = 0xB7;
sbit WR   = 0xB6;
sbit T1   = 0xB5;
sbit T0   = 0xB4;
sbit INT1 = 0xB3;
sbit INT0 = 0xB2;
sbit TXD  = 0xB1;
sbit RXD  = 0xB0;
sbit TX_ON = 0xB4;

/*  SCON  */
sbit SM0  = 0x9F;
sbit SM1  = 0x9E;
sbit SM2  = 0x9D;
sbit REN  = 0x9C;
sbit TB8  = 0x9B;
sbit RB8  = 0x9A;
sbit TI   = 0x99;
sbit RI   = 0x98;
sbit Pino0 = 0xe0;
sbit Pino1 = 0xe1;
sbit Pino2 = 0xe2;
sbit Pino3 = 0xe3;
```

APÊNDICE F – Modificações nas rotinas LE_QUADRO e TRANSMITE_FRAME para o protótipo trabalhar com um valor fixo

```
void transmite_frame() /* função para rotacionar byte e ligar se menor que 127 ou apagar se
maior que 128 */
{

unsigned char temp, dadosaida;
unsigned int i,j;
unsigned int contador;

CSB = 1;
TI = 0;
for (j=0;j<tamanhoframe;j=j+8)
  {

   dadosaida = 0;
   for (i=j;i<(j+8);i++)  /*continua dividindo o frame por 8*/
   {
    temp = memoria[i];
    /*if (temp < 128)
       dadosaida = dadosaida << 1;
    else
       dadosaida = (dadosaida << 1)+1; */
   };  /* não aplica o algoritmo de conversão */

   SBUF = temp;

   while (TI==0);
   TI = 0;
   for(contador=0; contador <50 ; contador++);

  }
}


void le_quadro()
{

   char temp;
   char le_frame;
   temp = 0;
   pointer = 0;
   le_frame = 1;
   P17 = 0;
   while (VSYNC == 0);
   while (VSYNC == 1);

   while (le_frame == 1)
   {

       while (HREF == 0);

       while (HREF == 1)
       {
         while (PCLK == 1);
         CSB = 0;
         temp = dado;
         CSB = 1;
         while (PCLK == 0);
         memoria[pointer] = 85;  /*memória recebe o valor fixo 85 (01010101) */
         pointer++;
       }
    if( pointer > 10366 )
       le_frame = 0;
   }
   tamanhoframe = pointer;
   IE = 0;
```

ANEXO A – Esquema elétrico do módulo MONITOR