UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROJETO DE UMA MATRIZ DE ÁUDIO E VÍDEO

YVES MAIA SALVATORI

ORIENTADOR: RICARDO ZELENOVSKY

TRABALHO DE CONCLUSÃO DE CURSO EM ENGENHARIA ELÉTRICA

BRASÍLIA/DF: JULHO – 2009

UNIVERSIDADE DE BRASÍLIA

FACULDADE DE TECNOLOGIA

DEPARTAMENTO DE ENGENHARIA ELÉTRICA

PROJETO DE UMA MATRIZ DE ÁUDIO E VÍDEO

YVES MAIA SALVATORI

DISSERTAÇÃO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ENGENHEIRO.

APROVADA POR:

_____

Ricardo Zelenovsky (ENE-UnB)
(Orientador)

_____

Janaína Gonçalves Guimarães
(Examinador Interno)

_____

Ícaro dos Santos
(Examinador Interno)

BRASÍLIA/DF, 9 DE JULHO DE 2009.

FICHA CATALOGRÁFICA

SALVATORI, YVES MAIA

Projeto de uma matriz de áudio e vídeo [Distrito Federal] 2009.
xi, 40p., 210 x 297 mm (ENE/FT/UnB, Engenheiro Eletricista, Engenharia Elétrica, 2009).

Trabalho de Conclusão de Curso de Graduação – Universidade de Brasília. Faculdade de Tecnologia.
Departamento de Engenharia Elétrica.

1.Matriz 2.Microcontroladores
3.Roteamento de sinais 4.Display

I. ENE/FT/UnB II. Título (série)

REFERÊNCIA BIBLIOGRÁFICA

SALVATORI, YVES M.
Projeto de uma matriz de áudio e vídeo. (Trabalho de Conclusão de Curso de Graduação), Julho/2009, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 40p.

CESSÃO DE DIREITOS

AUTOR: Yves Maia Salvatori
TÍTULO: Projeto de uma matriz de áudio e vídeo
GRAU/ANO: Engenheiro Eletricista/2009

_____

Yves Maia Salvatori
SQS 302 Bl. D, Apt. 101, Asa Sul.
CEP: 70.338-040
Brasília – DF – Brasil.

AGRADECIMENTOS

A Deus por tudo que tenho, em especial, minha saúde e minha família.

À minha família pela minha educação, amor e apoio incondicional.

Aos amigos com os quais compartilhei as dificuldades do curso e vivi momentos de estudos, descontração, alegrias e incertezas.

Aos amigos da TV Senado que me incentivaram, me ajudaram e despertaram em mim o gosto pela eletrônica.

A todos que me apoiaram.

Yves Maia Salvatori

RESUMO

Este trabalho é um projeto de um aparelho capaz de rotear sinais de áudio e vídeo de oito entradas para três saídas. O usuário utiliza um teclado numérico para comandar o destino das entradas observando os procedimentos em um “display” LCD.

São apresentadas as etapas de projeto e os aspectos mais importantes em cada uma delas, passando pela definição das funcionalidades e da arquitetura, especificação e funcionamento dos dispositivos, implementação do hardware e do software e a discussão dos resultados.

ABSTRACT

This work is a project of a device capable of routing eight inputs for three outputs of audio and video signals. The user uses a numeric keyboard to command the route of the inputs watching the procedures in a LCD display.

The steps project and the most important aspects in each one of them are presented, passing through the functionalities and archtecture definitions, device especifications and functionalities, software and hardware implementation and the discussion of the results.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SÍMBOLOS, NOMENCLATURAS E ABREVIAÇÕES

CI	*Circuito Integrado*
DIP	*Dual In Line Package*
IDE	*Integrated Drive Electronics*
LCD	*Liquid Crystal Display*
TSSOP	T*hin-Shrink Small Out line Package*

1. INTRODUÇÃO

Em emissoras de televisão é comum a necessidade de se endereçar sinais de áudio e vídeo de alguns aparelhos para diversos outros com a finalidade de monitorar os sinais, realizar cópias de mídias, fazer edições em mesas de áudio e vídeo ou até mesmo disponibilizá-los para outras emissoras. Os aparelhos responsáveis pelo endereçamento de sinais são chamados comumente de matriz. Matrizes também podem ser utilizadas em aplicações de segurança, nas quais diversas câmeras monitoram locais diferentes e o usuário escolhe a imagem de qual câmera deseja observar no monitor.

Este projeto tem como objetivo a construção de uma matriz capaz de endereçar sinais de áudio e vídeo para facilitar a cópia de fitas VHS, DVD's e outros tipos de mídia. Com a utilização do aparelho a ser projetado, será eliminada a necessidade de mudança na conexão dos cabos entre os aparelhos envolvidos na cópia toda vez que se desejar copiar em uma mídia diferente, pois todos os aparelhos estarão conectados à matriz que fará o endereçamento de acordo com a vontade do usuário. Será possível, também, a realização de cópias simultâneas de uma mesma mídia.

Nos capítulos seguintes, serão apresentadas as etapas de pojeto, a implemantação do hardware e do software e a discussão dos resultados. O capítudo 2, referente às etapas de projeto, apresenta a definição das funcionalidades a serem atendidas, a definição da lógica e arquitetura do aparelho e a definição e funcionamento dos principais dispositivos. Os capítulo 3 e 4 apresentam as etapas de realização do hardware e os procedimentos referentes à implementação do software. Por último, no capítulo 5, é realizada uma análise dos resultados obtidos.

2. ETAPAS DE PROJETO

Este capítulo enumera e aborda as principais etapas do projeto. As decisões de determinadas etapas devem ser levadas em conta somente após analisar uma decisão ou característica de outra etapa. Desta forma o projeto consiste, algumas vezes de idas e vindas de uma etapa para outra compondo, muitas vezes, processos de iteração.

2.1. Definição das funcionalidades

A primeira etapa do projeto visa identificar as funções que devem estar presentes no aparelho. Ela visa, basicamente, responder as seguintes perguntas:

- Quantas entradas e saídas a matriz deve ter?
- Quais sinais estão presentes nas entradas e saídas?
- Como é a comunicação com o usuário?

A definição da quantidade de entradas e saídas da matriz levou em conta a capacidade dos circuitos integrados pesquisados na etapa “definição dos dispositivos” e a quantidade de sinais em cada entrada e saída. Evitou-se uma grande quantidade de entradas e saídas para não tornar o processo de implementação muito trabalhoso e não encarecer o projeto.

A quantidade de sinais que cada entrada deve comportar se baseou nas características das mídias a serem reproduzidas e dos aparelhos utilizados. Geralmente as mídias possuem um sinal de vídeo composto e dois canais de áudio (CH 1 e CH2), ambos balanceados.

Definiu-se, portanto, que a matriz deveria ter 8 entradas e 3 saídas, sendo cada entrada ou saída composta de um sinal de vídeo composto e dois canais de áudio balanceados (CH 1 e CH2).

Decidiu-se que a comunicação com o usuário seria feita utilizando um teclado e um “display”. O usuário tecla a entrada escolhida e, em seguida, indica para qual saída deseja selecionar. Em cada passo o usuário deverá confirmar ou cancelar o número digitado e o display deve fornecer a indicação dos passos para a comutação.

2.2. Definição da Lógica/Arquitetura

Esta é um das principais etapas do projeto e visa fazer um levantamento dos blocos funcionais mais importantes de forma a estabelecer uma lógica de funcionamento. Ela é responsável por definir a arquitetura do aparelho e como cada bloco funcional interage com outro.

O diagrama em blocos da lógica de funcionamento é mostrado a seguir:

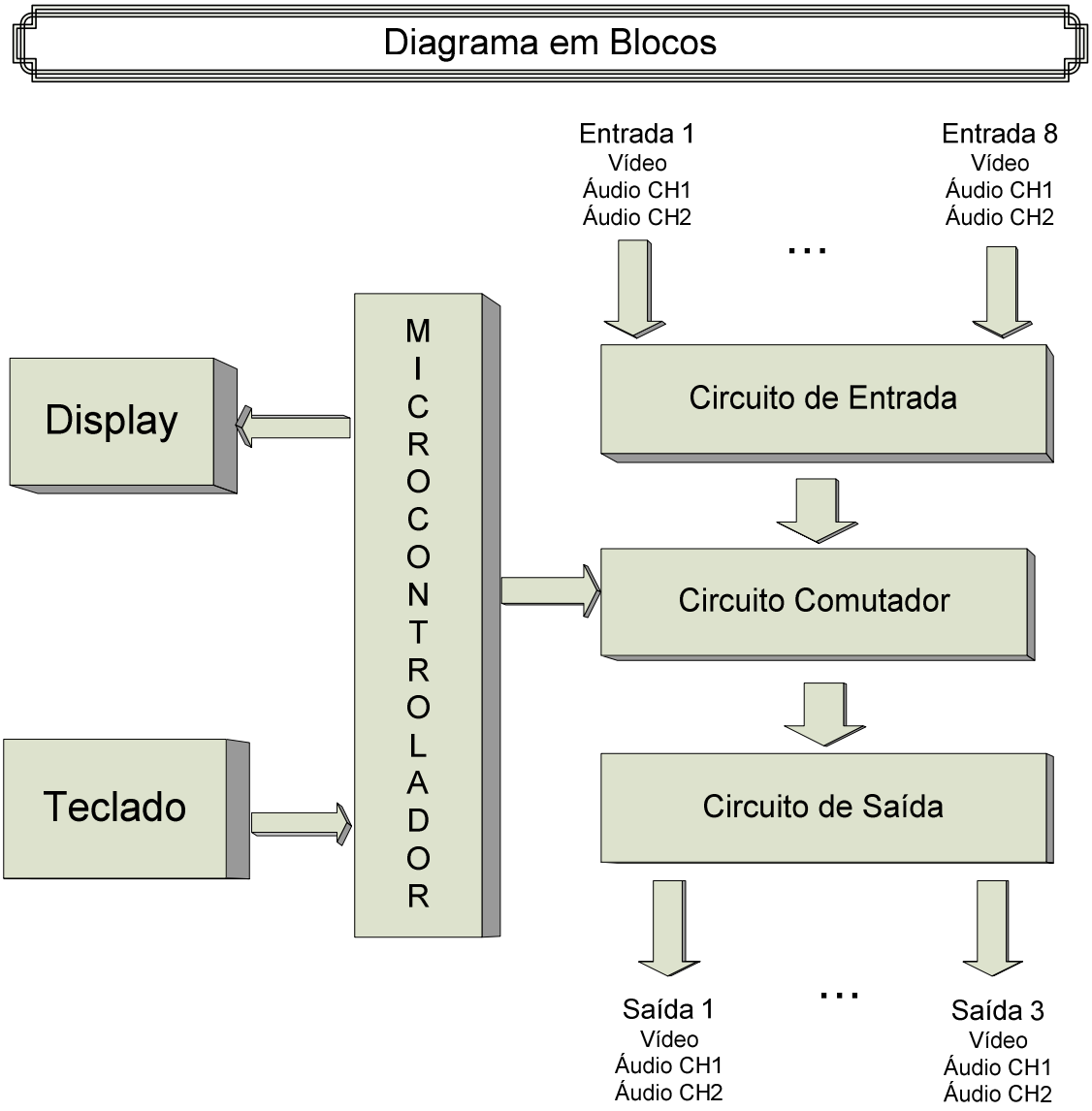


Figura 1 Diagrama em blocos

A função principal é desempenhada pelo microcontrolador que deve ler os dados digitados pelo usuário e comandar adequadamente o bloco responsável pela comutação dos sinais. O microcontrolador também deve atualizar o display com informações para interface com o usuário indicando em qual etapa se encontra o processo de comutação.

O circuito de entrada é necessário para desbalancear os sinais de áudio, eliminando o ruído devido à transmissão.

O circuito responsável pelo roteamento dos sinais é comandado pelo microcontrolador e é seguido pelo circuito de saída, que realiza o ganho dos sinais de áudio e vídeo, o balanceamento dos sinais de áudio e o casamento de impedância apropriado para entregar os sinais aos aparelhos conectados às suas saídas de maneira apropriada.

Há ainda um bloco funcional secundário que é responsável por ativar o modo gravação do microcontrolador e fazer a conversão de sinais TTL para o protocolo RS 232 e vice-versa, possibilitando a transferência do programa .

Os blocos funcionais LCD, Teclado e Microcontrolador possuem caráter de eletrônica digital e fazem a função de comando do circuito comutador. Já os circuitos de entrada e saída possuem caráter de eletrônica analógica e fazem a função de tratamento dos sinais. Desta forma, pode-se dizer que o aparelho é composto de uma parte de comando e uma parte de tratamento dos sinais.

2.3. Definição e funcionamento dos principais dispositivos

Esta etapa explica a escolha dos disposivos referentes a cada bloco funcional detalhando seu funcionamento.

2.3.1. Teclado

O teclado escolhido foi um teclado telefônico de 12 teclas do tipo matricial, pois o número de teclas é suficiente, o funcionamento é bem simples e é facilmente encontrado em sucatas de aparelhos telefônicos. Além dos botões numerados de 1 a 8 para indicar o número da entrada ou saída que se pretende rotear, escolheu-se utilizar a tecla “*” para corrigir e a tecla “#” para confirmar cada operação.

O teclado matricial utilizado é formado por 4 linhas e 3 colunas onde cada tecla apertada curto-circuita uma linha com uma coluna.

Para efetuar-se a leitura do teclado, basta conectar cada linha e cada coluna à porta paralela do microcontrolador.

A rotina que realiza a leitura identifica a tecla pressionada observando qual coluna foi curto-circuitada com qual linha. Isso é feito da seguinte maneira: inicialmente, todas as linhas e colunas apresentam nível lógico 1. Sucessivamente, faz-se aparecer nível lógico zero em cada linha, uma de cada vez e com tal velocidade que ao se pressionar uma tecla, o usuário está curto-circuitando a linha em nível lógico zero com uma coluna, levando-a para nível lógico zero também. Observando qual coluna foi para nível lógico zero e sabendo qual era a linha que estava em nível zero naquele momento, a tecla pressionada é identificada.

O contato do teclado não é perfeito, pois se trata de algo mecânico. Nesse caso, ocorre um fenômeno conhecido como “bounce”. Ao se pressionar uma tecla uma única vez, o contato ocorre e deixa de ocorrer diversas vezes. A rotina que faz a leitura da tecla pelo microcontrolador preve esse fenômeno e aguarda um determinado tempo desde a ocorrência do primeiro contato para que a tecla não seja lida mais de uma vez.

O esquema do teclado é mostrado na figura 2**.**

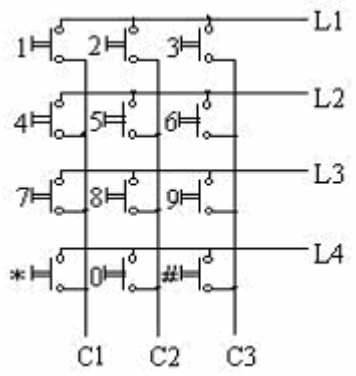


Figura 2 Esquema do teclado

2.3.2. Display

O Display escolhido foi um LCD alfa numérico gerenciado por processador Hitachi HD44780 e com dimensão 20x4, ou seja, com 4 linhas e 20 colunas. A razão para a escolha de ter quatro linhas foi a idéia de se ter o estado atual de cada saída em uma linha do

display e a comunicação para novas comutações de sinal em uma outra linha. Como são três saídas, necessitou-se de um display com 4 linhas.

O controle do LCD é feito por diversos sinais de controle: RS, RW e E. Para que um dado comando seja executado é necessário que ocorra um puslo no pino E. O comando executado depende da combinação entre os dois pinos de comando RS e RW conforme mostra a tabela 1.

RW	RS	Comando
0	0	Escreve Instrução
0	1	Escreve dado
1	0	Ler Bit de ocupado ou contador de endereços
1	1	Ler dado da CGRAM ou DDRAM

Tabela 1 Comandos do LCD de acordo com RW e RS

Os pinos D0, D1, D2, ..., D7 são os pinos pelos quais trafegam os dados ou instruções sendo possível que se utilize comunicação usando 8 ou 4 bits. Para economizar pinos do microcontrolador foi escolhido o modo de comunicação em quatro vias. Nesse modo, os pinos D0 até D4 não são utilizados e os bytes são divididos em dois “nibbles”. Envia-se primeiro o “nibble” mais significativo e em seguida o menos significativo.

O LCD executa algumas funcionalidades interessantes que são configuradas utilizando-se instruções. Entre as funcionalidades estão incremento ou decremento automático do contador de endereço, deslocamento do mostrador ou do cursor, exibição ou não do cursor, cursor piscante, “display” desligado e limpeza do “display”.

Alguns comandos devem ser passados na inicialização para configurar o modo 8 ou 4 vias, o tamanho dos caracteres (5x8 ou 5x10), a quantidade de linhas utilizadas, a presença do cursor e se é piscante ou não. Outros comandos servem para posicionar o cursor em determinada posição ou escrever caracteres no mostrador ou na memória para criação de novos caracteres.

Com relação aos aspectos físicos, o LCD ainda possui os pinos de alimentação, de contraste das letras (se mais claras ou mais escuras) e a alimentação para a luz de fundo conhecida como “backlight”. A figura 3 mostra a frente e o verso do “display” LCD utilizado.

Figura 3 Frente e verso do display LCD

Uma tabela com o resumo de todos os comandos é mostrado na tabela 2**.** Outra tabela prática útil é a tabela 3 que mostra o número que deve ser passado como dado para posicionar o cursor em determinada posição do mostrador.

COMMAND	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DESCRIPTION	Executing time fosc=250khz
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears Display & Returns to Address 0.	1.64ms
Cursor at Home	0	0	0	0	0	0	0	0	1	x	Returns Cursor to Address 0. Also returns the display being shifted to the original position. DDRAM contents remain unchanged.	1.64ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	I/D: Set Cursor Moving Direction I/D=1: Increment I/D=0: Decrement S: Specify Shift of Display S=1: The display is shifted S=0: The display is not shifted	40µs
Display ON/OFF Control	0	0	0	0	0	0	1	D	C	B	Display D=1: Display on D=0: Display off Cursor C=1: Cursor on C=0: Cursor off Brink B=1: Brink on B=0: Brink off	40µs
Cursor / Display Shift	0	0	0	0	0	1	S/C	R/L	x	x	Moves cursor or shifts the display w/o changing DD RAM contents S/C=0: Cursor Shift (RAM unchanged) S/C=1: Display Shift (RAM unchanged) R/L=1: Shift to the Right R/L=0: Shift to the Left	40µs
Function Set	0	0	0	0	1	DL	N	F	x	x	Sets data bus length (DL), # of display lines (N), and character fonts (F). DL=1: 8 bits F=0: 5x7 dots DL=0: 4 bits F=1: 5x10 dots N=0: 1 line display N=1: 2 lines display	40µs
Set CG RAM Address	0	0	0	1	Character Generator (CG) RAM Address						Sets CG RAM address. CG RAM data is sent and received after this instruction.	40µs
Set DD RAM Address	0	0	1	Display Data (DD) RAM Address / Cursor Address							Sets DD RAM address. DD Ram data is sent and received after this instruction.	40µs
Busy Flag / Address Read	0	1	BF	Address counter used for both DD & CG RAM address							Reads Busy Flag (BF) and address counter contents.	40µs
Write Data	1	0	Write Data								Writes data into DDRAM or CGRAM.	46µs
Read Data	1	1	Read Data								Reads data from DDRAM or CGRAM.	46µs

X: Don't Care

Tabela 2 Resumo de comandos do LCD

20x04

80	81	82	83	84	85	86	87	88	89	8A	8B	8C	8D	8E	8F	90	91	92	93
C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF	D0	D1	D2	D3
94	95	96	97	98	99	9A	9B	9C	9D	9E	9F	A0	A1	A2	A3	A4	A5	A6	A7
D4	D5	D6	D7	D8	D9	DA	DB	DC	DD	DE	DF	E0	E1	E2	E3	E4	E5	E6	E7

Tabela 3 Posicionamento do cursor

2.3.3. Microcontrolador

Devido a familiaridade com microcontroladores da família 8051 desenvolvida durante o curso, adotou-se como critério para escolha do microcontrolador ser pertencente à família 8051. Os outros critérios para a escolha foram apresentar memória de programa interna para evitar a utilização de memórias externas e possuir o encapsulamento do tipo DIP para que fosse possível a realização de testes em protoboard e tornar fácil a soldagem.

Com base nesses critérios, o microcontrolador escolhido foi o DS89C450 da Dallas-Maxim. Tal microcontrolador apresenta características muito além das requeridas para o projeto, sendo sua memória de programa interna do tipo flash com capacidade de 64 Kbytes, possuindo também 1 Kbyte de memória SRAM para substituir memória de dados externa e RAM interna de 384 bytes.

Esse microcontrolador suporta um relógio de até 33 MHz e é capaz de executar operações em apenas um ciclo de relógio, diferentemente da maioria dos microcontroladores da família 8051, que realizam em 12 ciclos de relógio. A pinagem do microcontrolador é mostrada na figura 4.

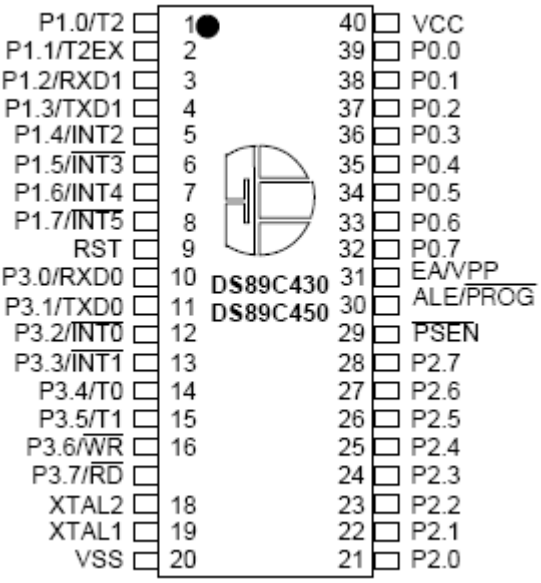


Figura 4 Pinagem do Microcontrolador

2.3.4. Circuito comutador

Enquanto todos os blocos funcionais dizem respeito ao controle da operação ou ao tratamento dos sinais o bloco funcional circuito comutador é responsável pela realização do propósito ao qual se destina o aparelho: rotear os sinais. Existem diversos circuitos integrados capazes de rotear sinais.

Os critérios para escolha do circuito integrado foram a capacidade, em termos de largura de banda, maior que 4 MHz, que é a banda de um sinal de vídeo e, novamente, o tipo de encapsulamento DIP. Para faciliar a comunicação com o microcontrolador, procurou-se CI's com possibilidade de comunicação paralela. Para reduzir a quantidade de CI's, procurou-se um com capacidade razoável de número de entradas e saídas.

Utilizando tais requisitos, o CI escolhido para realizar o roteamento dos sinais foi o MAX4360 da Dallas-Maxim que é uma matriz 8x4. Como não estava disponível para compra, foi substituído pelo MAX4456 que apresenta as mesmas características porém com maior dimensão (8x8).

Como cada CI apresenta 8 entradas, são necessários três CI's para atender as especificações de funcionalidades do projeto sendo um responsável pelo roteamento dos sinais de vídeo, outro pelo roteamento dos sinais de áudio do canal 1 e o último pelos sinais de áudio do canal 2. Dessa forma, cada aparelho conectado ao aparelho a que se destina o projeto tem cada um dos seus três sinais entrando no mesmo número de entrada de cada um dos CI's comutadores. Isso facilita a programação, já que faz com que os três CI's necessitem receber a mesma palavra de comando e impede que uma dada saída possua vídeo de um aparelho e áudio de outro.

O MAX4456 possibilita tanto comunicação serial quanto paralela. O funcionamento da comunicação paralela é habilitado com o pino SER/PAR em nível baixo e se dá da seguinte maneira: três pinos especificam a entrada (D2, D1 e D0) e outros três especificam a saída (A2, A1 e A0). Um pulso no pino WR (borda de subida) insere no primeiro banco de registradores a informação de roteamento presente nesses pinos. O pulso em WR só é válido se os pinos de seleção de chip CE e #CE permitirem, ou seja, estiverem em nível lógico um e zero respectivamente.

O segundo banco de registradores funciona de duas maneiras dependendo do nível lógico do pino EDGE/LEVEL. Para EDGE/LEVEL em nível lógico baixo, o segundo banco de registradores é transparente sempre que o pino LATCH estiver em nível lógico baixo. Para EDGE/LEVEL em nível alto, o segundo banco de registradores só passa a informação de endereçamento para a matriz na borda de subida do pino LATCH. Essa segunda forma é conveniente somente quando se deseja atualizar o roteamento de mais de uma entrada ao mesmo tempo, pois as informações vão sendo armazenadas no primeiro banco de registradores e passam todas juntas para a matriz após a permissão do segundo banco.

O CI opera com alimentação simétrica de +5 e -5 volts e aceita como entrada sinais excursionando tensões de -1,3V a 1,3V. Em cada saída há um buffer com ganho de 1V/V e em seguida uma carga resistiva de 400 ohms que pode ser habilitada (LOAD=1) ou não (LOAD=0) de acordo com o nível lógico do pino LOAD. Para manter a estabilidade, os buffers necessitam da carga, que deve ser desabilitada quando se deseja agrupar mais de um CI para obter-se uma matriz com maiores dimensões.

As figuras 5, 6 e 7 mostram, respectivamente, o esquema de funcionamento, os procedimentos para atualização dos bancos de registradores e a pinagem do CI MAX4456

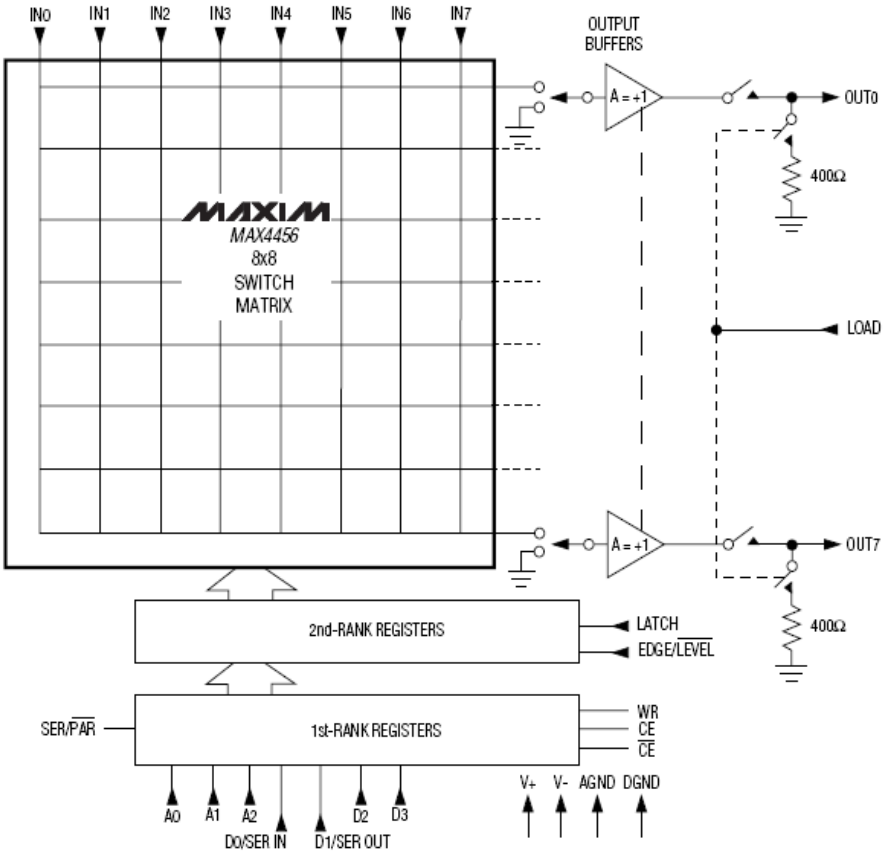


Figura 5 Esquema de funcionamento do CI Comutador

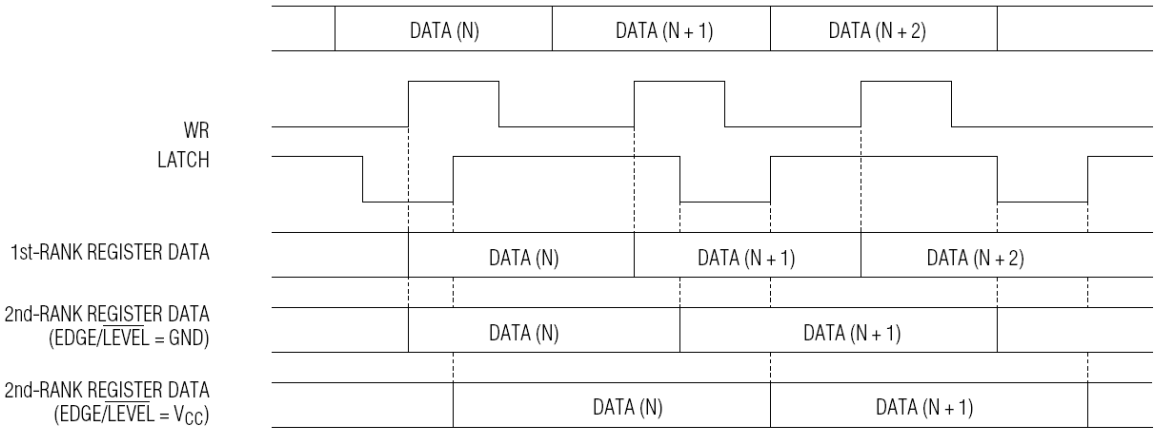


Figura 6 Procedimentos para atualização dos bancos de registradores

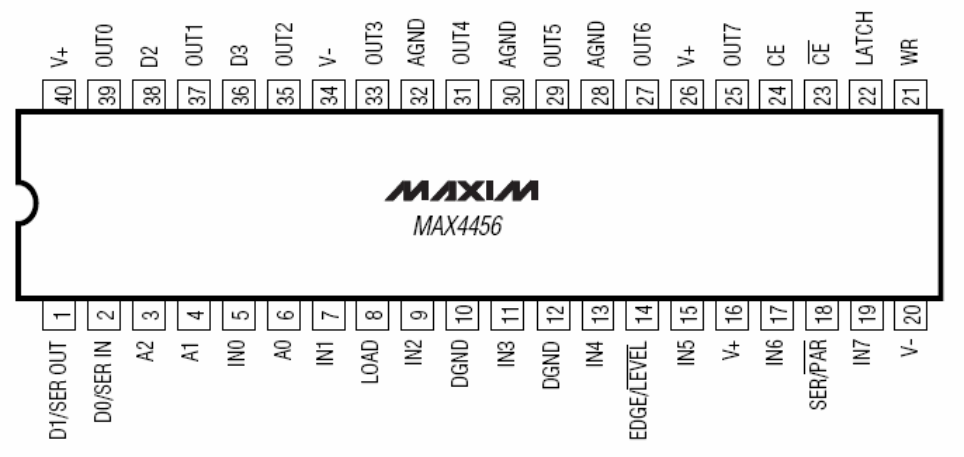


Figura 7 Pinagem do CI comutador

2.3.5. Amplificadores

a) Amplificador de Áudio

O amplificador de áudio realiza duas funções: o desbalanceamento dos sinais na entrada e o balanceamento dos sinais na saída.

Sinais balanceados possuem a mesma informação porém apresentam fases opostas. Para desbalanceá-los, basta que um deles seja defasado de 180 graus e realizada a soma. Isso corresponde a fazer a diferença dos dois sinais. Por isso, a configuração amplificador de diferenças pode ser utilizada para realizar o desbalanceamanto dos sinais na entrada.

A configuração amplificador de diferenças é mostrada na figura 8**.**

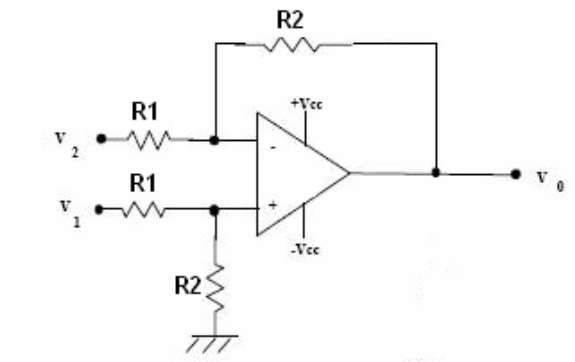


Figura 8 Configuração amplificador de diferenças

Para balancear os sinais na saída pode-se utilizar as configurações inversora e não inversora, ambas com ganho unitário.

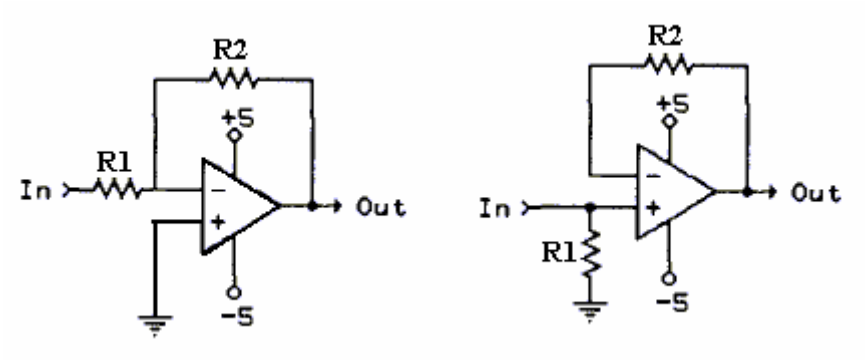


Figura 9 Configurações inversora e não-inversora respectivamente

O CI amplificador de áudio escolhido foi o LM324, que possui como vantagem a presença de quatro amplificadores operacionais dentro de um só chip. Entre suas principais características estão a largura de banda de 1,3 MHz (bem maior que necessária para sinais de áudio), a possibilidade de alimentação simétrica com valores entre 1,5 e 15 volts e offset de entrada limitado a 5 mV.

A pinagem é mostrada na figura 10.

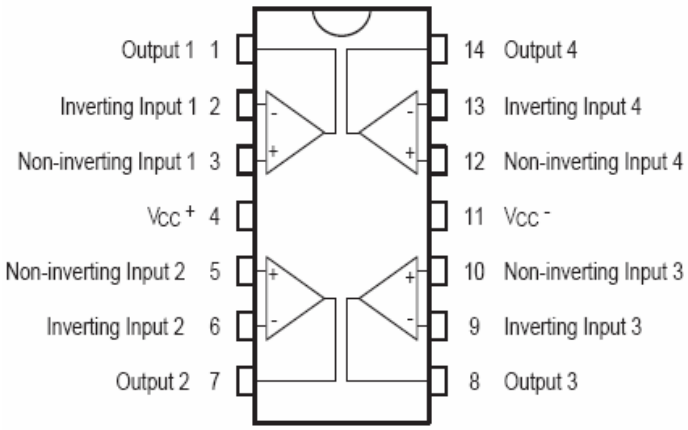


Figura 10 Pinagem do CI amplificador de áudio

b) Amplificador de Vídeo

O CI escolhido para amplificar os sinais de vídeo foi o MAX4395. Este CI é recomendado no datasheet do CI roteador dos sinais o qual prevê a seguinte configuração.

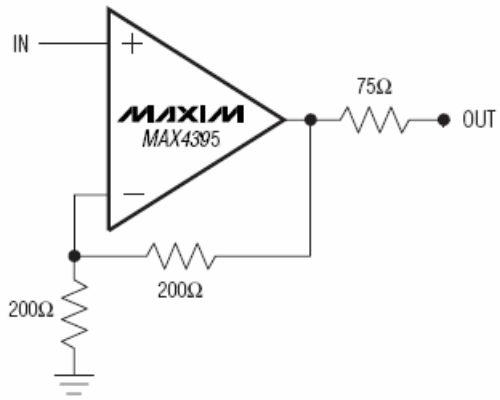


Figura 11 Esquema do CI amplificador de vídeo

O MAX4395 tem a vantagem de possuir 4 amplificadores num só chip, porém seu encapsulamento é do tipo TSSOP ,"*Thin-Shrink Small Out line Package*", o que impossibilita realização de testes em protoboard.

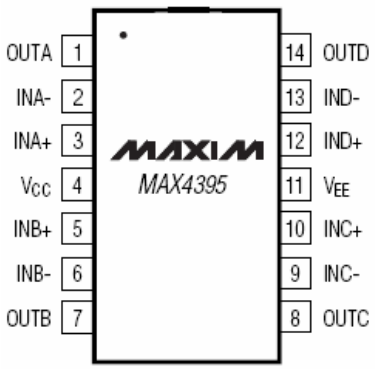


Figura 12 Pinagem do CI amplificador de vídeo

2.3.6. Outros

a) MAX232

A transferência do programa do computador para o microcontrolador é feita pela porta serial que emprega o protocolo RS 232. Os níveis de tensão empregados no protocolo RS 232 são: de -15 volts até -3 volts nível alto e de 3 volts até 15 volts nível baixo.

Logo, é necessário que os níveis de tensão sejam compatibilizados com os níveis TTL do microcontrolador. O CI MAX232 é o responsável pela compatibilização dos níveis de tensão.

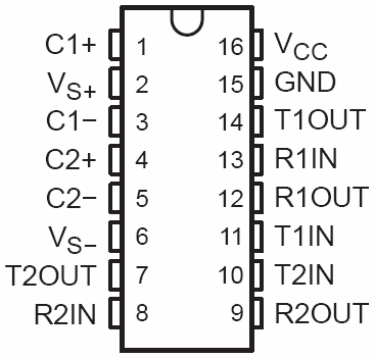


Figura 13 Pinagem do CI conversor de níveis lógicos

b) 74HCT12

O sinal DTR (*Data Terminal Ready*) vindo da porta serial, após compatibilizado pelo CI MAX232 é utilizado para controlar o envio de níveis de tensão para ativar o microcontrolador no modo gravação.

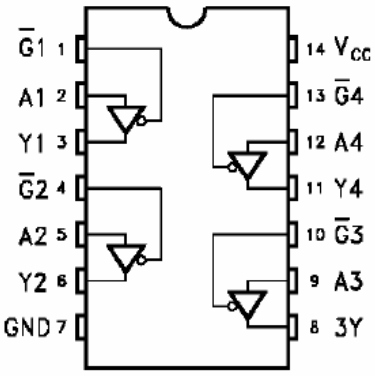


Figura 14 Pinagem do CI 74HCT125

c) MAX660

Para prover alimentação negativa de -5 volts, utilizou-se o CI MAX660. A pinagem dele é mostrada na figura 15.

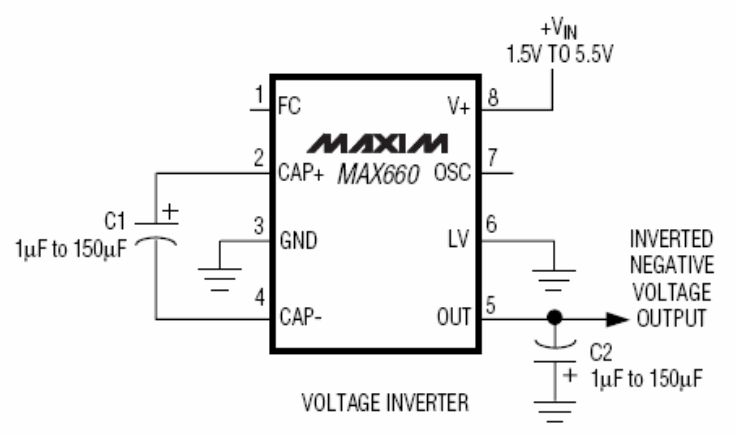


Figura 15 Pinagem do CI para prover alimentação simétrica

3. IMPLEMENTAÇÃO DO HARDWARE

A implementação do hardware foi dividida em etapas com a finalidade de isolar problemas e alcançarem objetivos específicos. O objetivo da primeira etapa é montar o circuito de comando em protoboard para verificar a correta comunicação dos dispositivos. Já a segunda etapa possui o objetivo de transferir o circuito testado em protoboard para uma placa para dar maior robustez e propiciar os testes do software sem que se preocupe com as ligações físicas dos dispositivos.

A terceira etapa realiza os testes do circuito comutador em protoboard para verificar a comutação dos sinais sendo os sinais de controle inseridos manualmente. Em seguida conecta o circuito de comando já com o software testado ao circuito comutador para verificar a correta comunicação entre os dois circuitos.

3.1. Etapa 1

Na primeira etapa, montou-se em protoboard a fonte de alimentação, o microcontrolador, os componentes necessários à sua gravação e os conectores para a comunicação com o LCD e com o teclado. O objetivo dessa etapa era testar a fonte de alimentação, a gravação do microcontrolador, a escrita no LCD e a leitura do teclado. Durante essa etapa a programação foi feita de maneira somente a testar o bom funcionamento do hardware.

A imagem da figura 16 mostra o circuito montado em protoboard na qual pode-se verificar a confecção de cabos com conectores IDE e a utilização de pinos para comunicação com o LCD de maneira improvisada para conexão na protoboard.

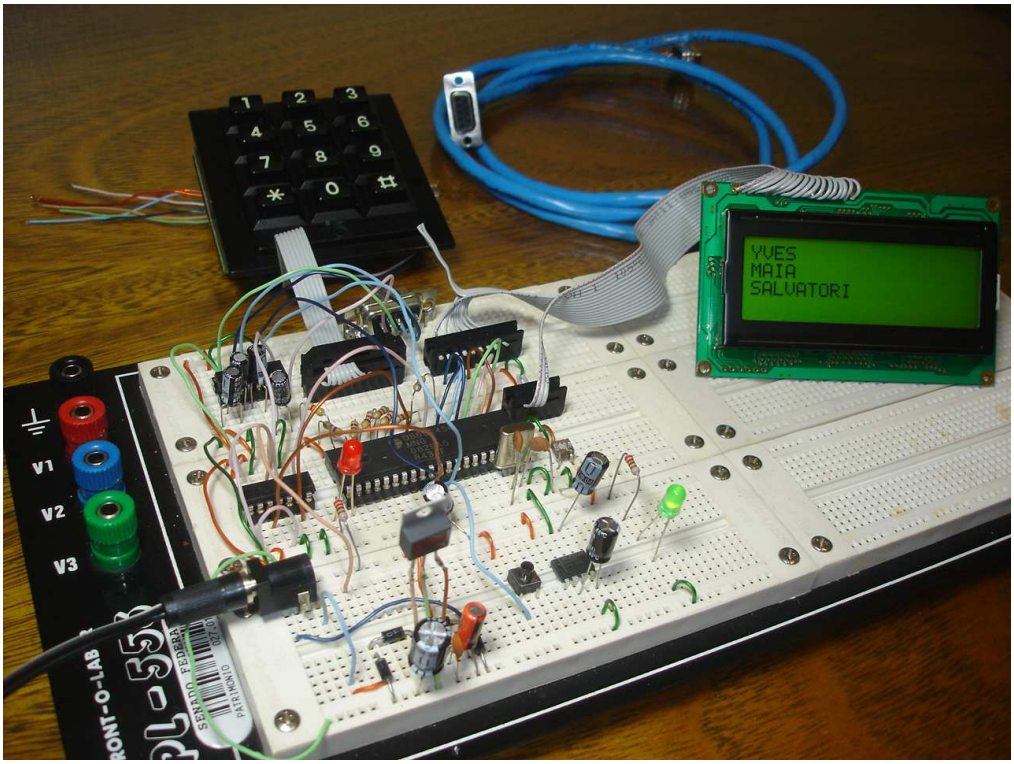


Figura 16 Circuito de comando montado em protoboard

3.2. Etapa 2

Após verificar o bom funcionamento em protoboard, o circuito foi soldado em placa padrão para evitar os inconvenientes da protoboard tais como mau contato, cuidado com o transporte para não soltar os fios, entre outros.

O processo de fabricação da placa de circuito impresso poderia atrasar bastante a etapa de desenvolvimento do software, pois para fabricação da placa de circuito impresso é necessário bom conhecimento do software de desenho do circuito para que pequenos detalhes não gerem erros na fabricação da placa. As vezes, há também que se atualizar as bibliotecas de componentes.

Preferiu-se a implementação em placa padrão à construção da placa de circuito impresso devido à versatilidade da placa padrão, pois nessa etapa o circuito encontra-se incompleto e em fase de teste, podendo outros compontentes serem acrescentados. Além disso, uma montagem mais rápida do circuito é desejada já que a etapa de programação apenas espera que o circuito esteja pronto para ser testada.

A figura 17 mostra o circuito de comando soldado na placa padrão e parafusado em um suporte de madeira. Pode-se observar a existência de pinos para conexão via cabo “flat” com conectores IDE com a finalidade de conectar o circuito de comando ao circuito comutador a ser montado posteriormente em protoboard. Por estes pinos saem os sinais de controle e de alimentação do circuito comutador.

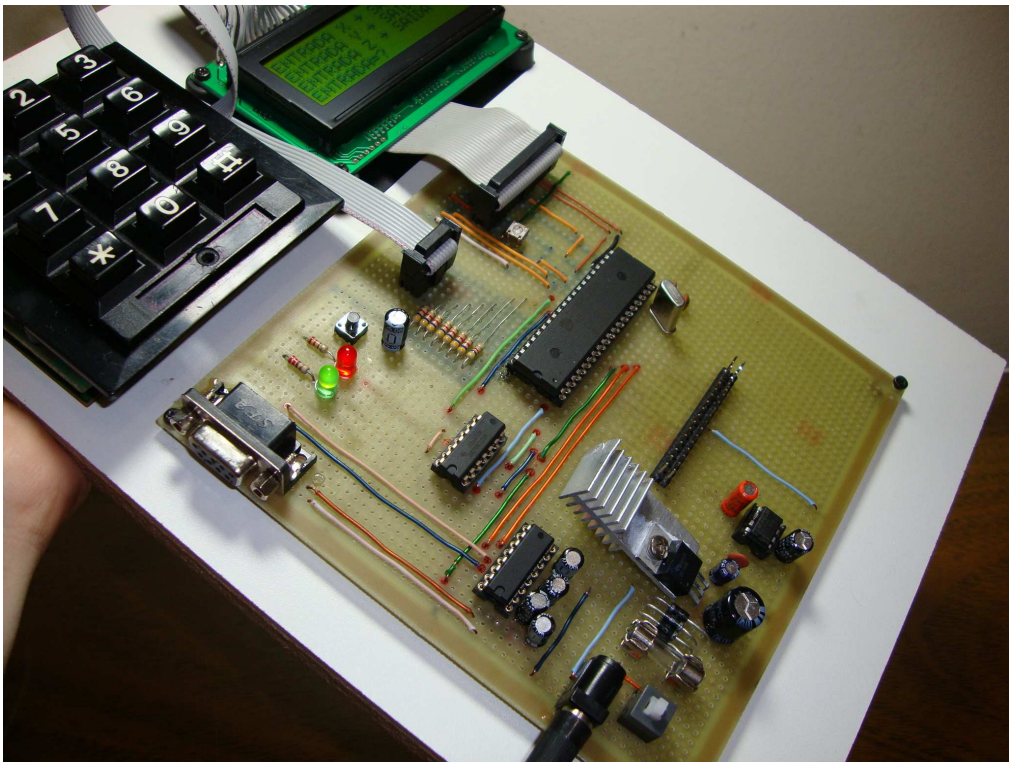


Figura 17 Circuito de comando

Com o término dessa etapa, deu-se início à programação do microcontrolador.

3.3. Etapa 3

A etapa 3 consiste em testar o circuito comutador. Após a programação, o CI MAX4456 foi montado em protoboard e aplicado manualmente os níveis lógicos necessários para realizar as comutações.

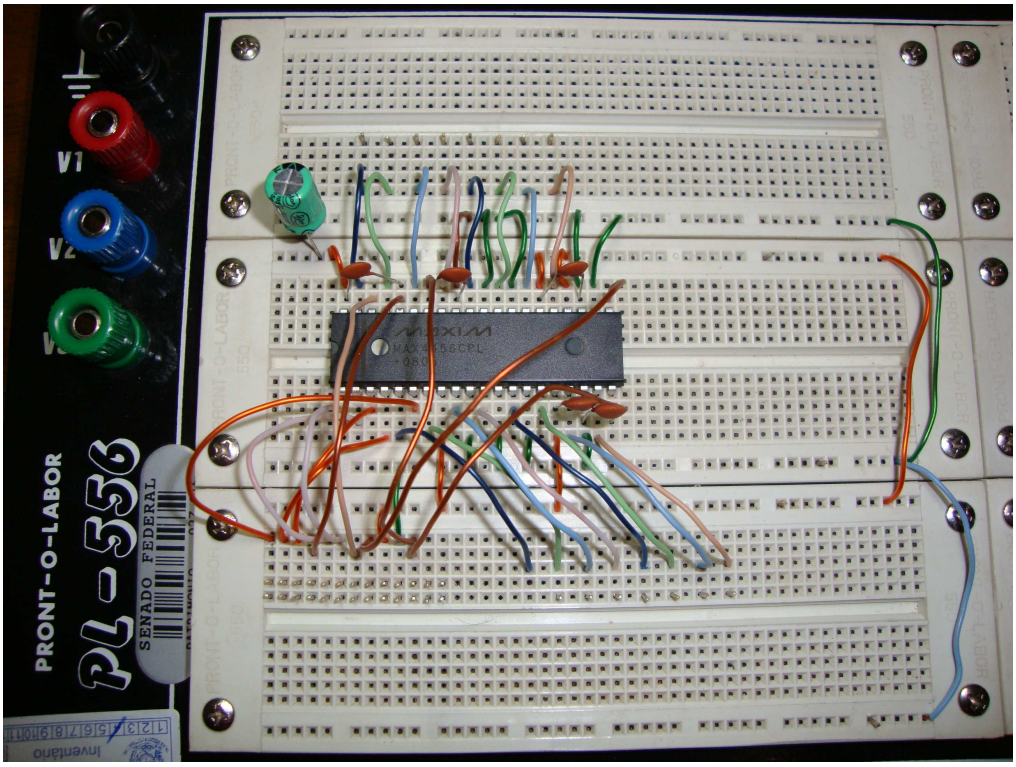


Figura 18 Circuito Comutador

Utilizou-se a fonte do circuito de comando já soldado em placa padrão para alimentar o CI. Fora utilizado um gerador de sinais para simular os sinais de áudio e vídeo nas entradas de sinais do CI. Aplicou-se um sinal de entrada senoidal com a mesma amplitude dos sinais de vídeo (um volt pico a pico) e com 1kHz de freqüência e observou-se no osciloscópio as saídas para as quais os sinais da entrada estavam sendo endereçados.

Para verificar a capacidade com relação a largura de banda requerida pelo sinal de vídeo, alterou-se a freqüência do sinal de entrada gradualmente até alcançar 4 MHz.

Após verificar o bom funcionamento do circuito comutador aplicando os níveis lógicos manualmente, seguiu-se para o teste da comutação utilizando o circuito de comando já montado e com a programação pronta. Para isso, utilizou-se cabo flat com

conector IDE e pinos na protoboard para conectar o circuito comutador ao circuito de comando.

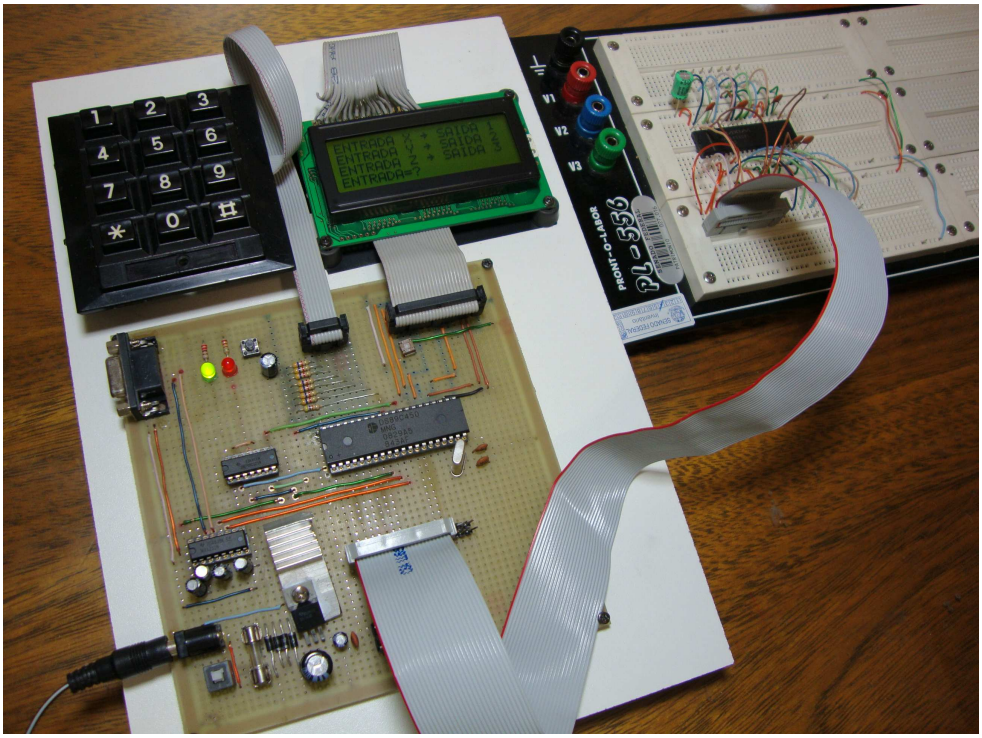


Figura 19 Conexão dos circuitos de comando e comutador

4. IMPLEMENTAÇÃO DO SOFTWARE

4.1. Funcionamento do programa

A implementação do software deve levar em consideração as especificações das funcionalidades. Como o aparelho possui três saídas, decidiu-se que as três primeiras linhas indicariam o estado das saídas 1, 2 e 3 respectivamente. A quarta linha do display ficou responsável por mostrar os procedimentos necessários à realização da comutação.

Dessa forma, ao ligar o aparelho, o display é inicializado no “estado A” com a mensagem mostrada na tabela 4**,** onde cada célula representa um caracter do mostrador.

E	N	T	R	A	D	A		X		→		S	A	I	D	A		1	
E	N	T	R	A	D	A		Y		→		S	A	I	D	A		2	
E	N	T	R	A	D	A		Z		→		S	A	I	D	A		3	
E	N	T	R	A	D	A	=	?											

Tabela 4 Mensagem do estado A do "display"

Após o usuário apertar qualquer tecla numérica, o display segue para o “estado B”, que é mostrado na tabela 5 para o caso do usuário teclar o número 5.

E	N	T	R	A	D	A		X		→		S	A	I	D	A		1	
E	N	T	R	A	D	A		Y		→		S	A	I	D	A		2	
E	N	T	R	A	D	A		Z		→		S	A	I	D	A		3	
E	N	T	R	A	D	A	=	5	?										

Tabela 5 Mensagem do estado B do "display"

O display continua no estado B perguntando ao usuário qual o valor da entrada e atualizando-se de acordo com o número da tecla pressionada até que seja pressionada a tecla confirma (#). Nesse caso o programa armazena o valor da entrada e segue o display para o “estado C”, que pergunta para o usuário para qual saída deseja endereçar o sinal de entrada já comunicado.

E	N	T	R	A	D	A		X		→		S	A	I	D	A		1	
E	N	T	R	A	D	A		Y		→		S	A	I	D	A		2	
E	N	T	R	A	D	A		Z		→		S	A	I	D	A		3	
E	N	T	R	A	D	A	=	5		→		S	A	I	D	A	?		

Tabela 6 Mensagem do estado C do "display"

Para qualquer tecla numérica que seja apertada de um até três o display segue para o “estado D” no qual permanece atualizando o número da tecla apertada até que seja pressionada a tecla confirma.

E	N	T	R	A	D	A		X		→		S	A	I	D	A		1	
E	N	T	R	A	D	A		Y		→		S	A	I	D	A		2	
E	N	T	R	A	D	A		Z		→		S	A	I	D	A		3	
E	N	T	R	A	D	A	=	5		→		S	A	I	D	A	=	1	?

Tabela 7 Mensagem do estado D do "display"

Após a confirmação, o programa executa as rotinas necessárias para realizar a comutação e o display retorna ao estado A, porém com o valor de X, Y ou Z alterado para o valor da entrada que está sendo direcionada para a dada saída.

Em qualquer dos estados, caso a tecla “CANCELA” (*) seja pressionada, o display volta ao estado A com os status anteriores de cada saída inalterados.

O diagrama da figura 20 resume o esquema da programação.

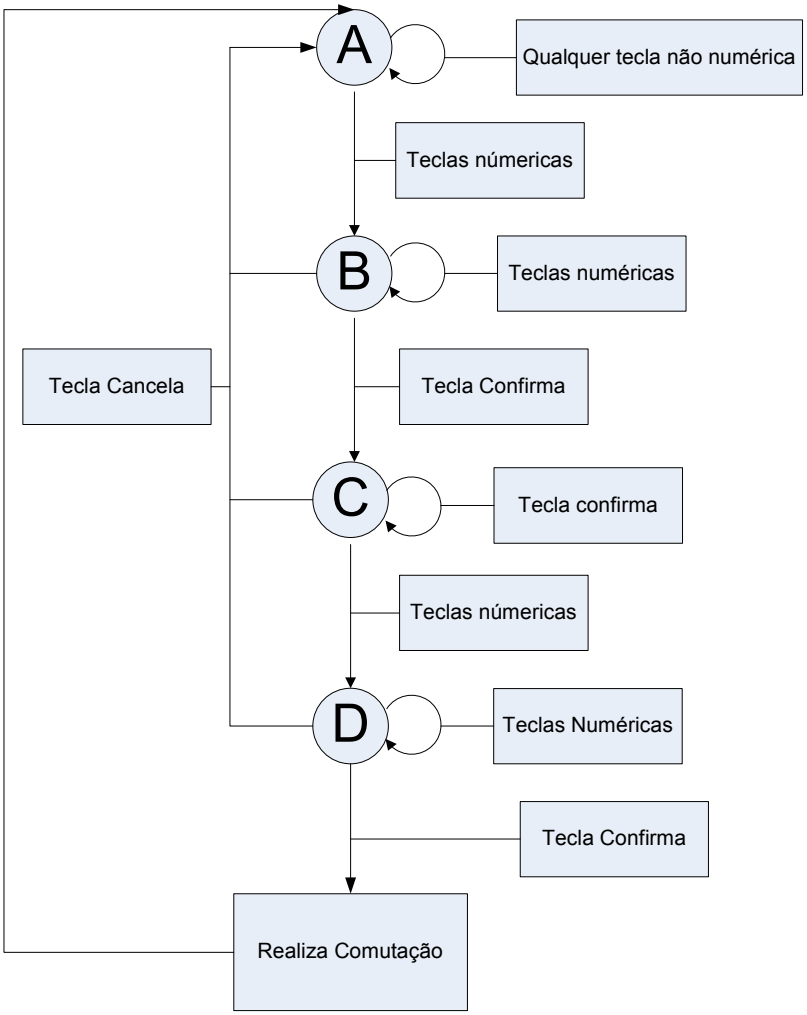


Figura 20 Estados do display

4.2. Detalhamento do programa

O programa foi escrito em linguagem assembly para o microcontrolador da família 8051 utilizado. A transferência do programa foi feita utilizando o software MTK2.

As principais rotinas do programa são a inicialização do display, a rotina responsável pela comunicação com o LCD, a leitura do teclado e a realização da comutação.

A rotina responsável pela comunicação com o LCD manipula os bytes que deverão ser entregues ao LCD para enviá-los em nibbles. Essa rotina é chamada sempre que se deseja escrever no LCD ou enviar qualquer intrução como limpeza do display ou posicionamento do cursor. Também são importantes as rotinas que servem para esperar um determinado tempo necessário para o LCD processar os comandos. Tais rotinas de temporização são de extrema importância para correta configuração de inicialização do display com as características desejadas.

Basicamente, após a inicialização e escrita da mensagem presente no estado A, o programa fica preso na rotina de leitura do teclado até que uma tecla seja apertada. A tecla apertada é identificada pela rotina de leitura do teclado que salva o código ASCII correspondente ao número da tecla em uma dada posição de memória.

Rotinas secundárias verificam em qual estado encontra-se a comutação para desviar a execução do programa para a rotina que irá salvar o valor da tecla apertada na variável entrada ou saída, de acordo com o estado do display. Em seguida, atualiza-se a mensagem do display e retorna-se à execução do programa para leitura do teclado ou para realização da comutação.

A rotina que realiza a comutação manipula os códigos ASCII salvos nas variáveis entrada e saída para correta comunicação com o CI comutador. A manipulação consiste em transformar os dois bytes de códigos ASCII, um da variável entrada e outro da variável saída, em um único byte contendo informações de ambas variáveis.

Antes de entrar na rotina que realiza a comutação, uma outra rotina verifica o bit que indica se os buffers do CI Matriz estão ligados. Caso negativo, ela desvia a execução do programa para uma outra rotina responsável por habilitar todos os buffers.

Após a rotina de realização da comutação, uma outra rotina prepara para atualizar a linha do display referente a saída comutada e retornar à rotina de leitura do teclado.

O programa foi testado várias vezes no circuito de comando já implementado em placa padrão no qual pode-se verificar o correto funcionamento do display. Para testar o correto funcionamento das rotinas de comutação, foram feitos os procedimentos necessários para comutar os sinais e verificado os níveis lógicos nos pinos do microcontrolador responsáveis por controlar o CI comutador.

O anexo A.2 mostra o código completo do software.

5. DISCUSSÃO DOS RESULTADOS

Os resultados obtidos referente ao circuito de comando e ao circuito comutador foram satisfatórios dado que fora verificado o correto endereçamento dos sinais de acordo com a vontade do usuário por meio do teclado e das informações corretamente apresentadas e atualizadas no display.

Os sinais comutados apresentaram excelente fidelidade em relação aos sinais de entrada, possuindo a mesma amplitude e fase até mesmo para sinais com elevada freqüência.

Os circuitos de entrada e de saída idealizados no diagrama em blocos mostrado na figura 1 do capítulo 2 não foram projetados, mas as configurações possíveis foram apresentadas. Esses circuitos realizam a função de balanceamento e desbalanceamento dos sinais de áudio e o casamento de impedância do sinal de vídeo. Apesar de já sugerido no datasheet do CI comutador, o CI amplificador de vídeo não foi comprado por não possuir encapsulamento DIP o que impossibilita o teste em protoboard. A implementação da placa de circuito impresso poderia resolver tal problema, mas não foi realizada para não onerar o projeto.

Os principais defeitos encontrados foram os seguintes:

• A segunda e a quarta linha do display são apagadas ao dar o reset no microcontrolador.

• Ao se pressionar um tecla por muito tempo é lida a tecla da primeira linha e mesma coluna da tecla apertada.

• As mensagens de atualização do display se perdem após a realização de diversas comutações.

Todos os defeitos encontrados são relacionados ao software e podem ser sanados com um maior rigor na programação.

6. CONCLUSÃO

O projeto do aparelho foi implementado em sua maioria e as comutações realizadas com sucesso. As etapas de projeto foram apresentadas e a escolha e funcionamento de cada dispositivo foram detalhados.

O diagrama em blocos apresentado na figura 1, no capítulo 2, mostra de maneira genérica o funcionamento do aparelho, podendo o mesmo circuito de comando servir para outras aplicações de acordo com o software nele gravado e com os circuitos a ele conectados.

Para o roteamento de sinais de áudio não balanceados os circuitos de entrada e saída tornam-se mais simples. Para o caso de roteamento apenas de sinais de vídeo, o projeto contempla todas as especificações já que o circuito comutador suporta a banda dos sinais de vídeo e o circuito casador de impedâncias presente na saída fora apresentado. A capacidade da matriz 8x 3 pode ainda ser aumentada já que os CI's comutadores possuem dimensão 8x8, mas nesse caso uma outra forma de apresentar o estado das saídas da matriz deve ser planejado.

Pode-se perceber a dificuldade em se construir um aparelho sendo que as maiores encontradas dizem respeito ao refinamento do projeto, ou seja, em torná-lo robusto para que não apresente erros. A implementação do projeto ocorreu em um nível acadêmico e uma visão mais empreendedora deve ter como preocupação a solução de todos os pequenos problemas para refinar o produto e torná-lo robusto podendo ainda torná-lo mais interativo com o usuário mostrando outras mensagens no "display".

É necessário estudar com maior cuidado as funcionalidades requeridas pelos usuários comerciais de aparelhos roteadores de sinais como matrizes de áudio e vídeo ou circuitos comutadores de vídeo utizados em segurança para atender melhor o mercado ao qual se destina o produto.

REFERÊNCIAS BIBLIOGRÁFICAS

[1] MENDONÇA, A.; ZELENOVSKY, A. "*Microcontroladores: Programação e Projeto com a Família 8051*." Rio de Janeiro: MZ Editora, agosto de 2005.

[2] RAMOS, E. SOUZA "Comunicação de LCD em 4 vias" Setembro de 2005.

[3] SEDRA, ADEL.; SMITH, KENNETH, C. "Microeletrônica" São Paulo:Pearson Makron Books, 2000.

[4] LANDO, R. ANTONIO; ALVES, S. RIOS "*Amplificador Operacional" São Paulo:Érica, 1993*

[5] Datasheet do microcontrolador disponível em: "http://datasheets.maxim-ic.com/en/ds/DS89C430-DS89C450.pdf"

[6] Datasheet do CI comutador disponível em: http://datasheets.maxim-ic.com/en/ds/MAX4359-MAX4456.pdf.

[7] Datasheet do CI para alimentação simétrica disponível em: http://datasheets.maxim-ic.com/en/ds/MAX660.pdf.

[8] Datasheet do CI amplificador de vídeo disponível em: "http://datasheets.maxim-ic.com/en/ds/MAX4389-MAX4396.pdf".

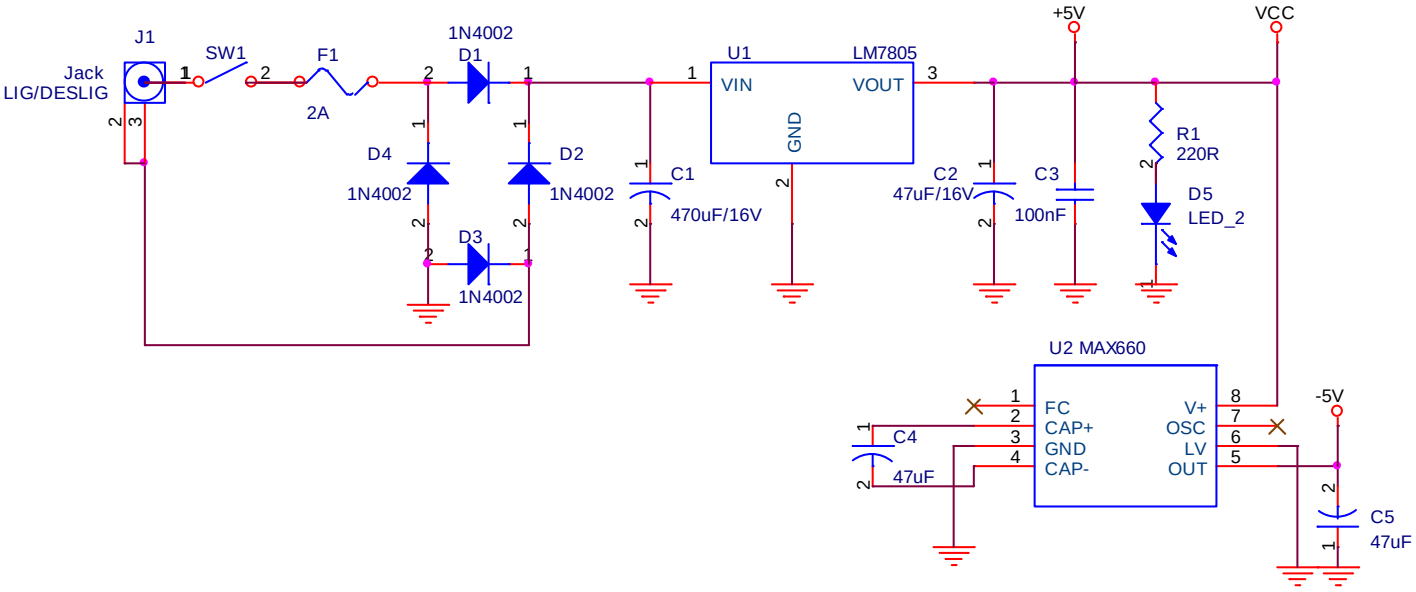
[9] Datasheet do CI conversor de níveis lógicos disponível em: "http://datasheets.maxim-ic.com/en/ds/MAX220-MAX249.pdf".

[10] Datasheet do CI amplficador de áudio disponível em: "http://www.datasheetcatalog.net/pt/datasheets_pdf/L/M/3/2/LM324.shtml".

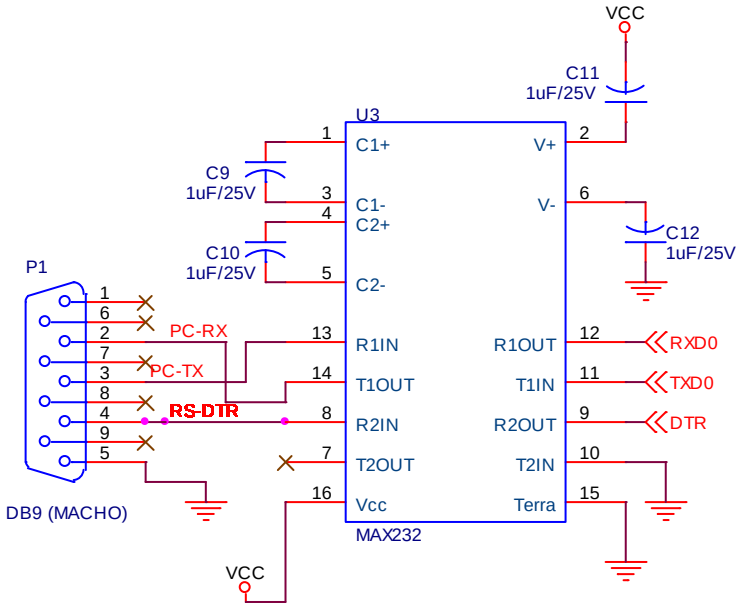
ANEXO A

A.1. Esquemático do aparelho

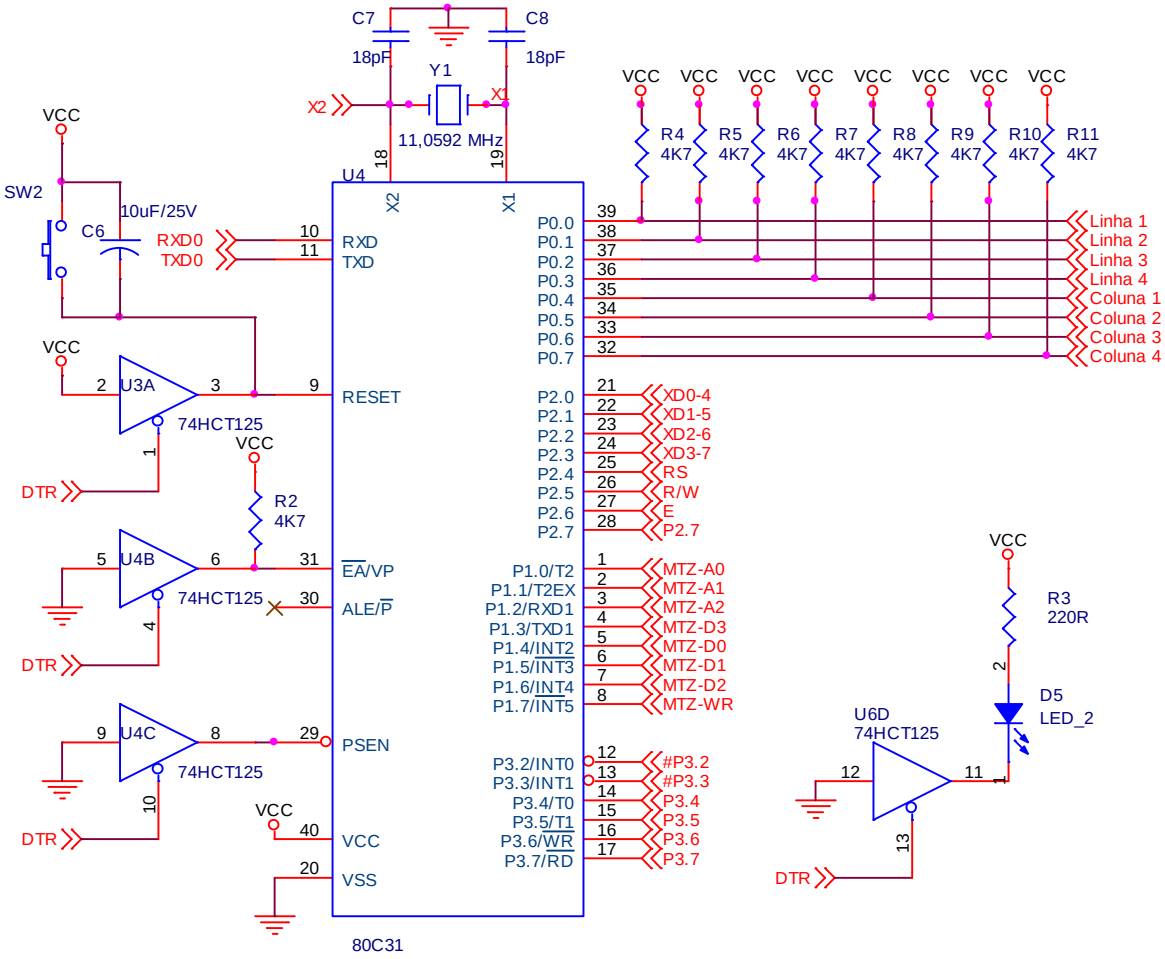
Fonte de alimentação



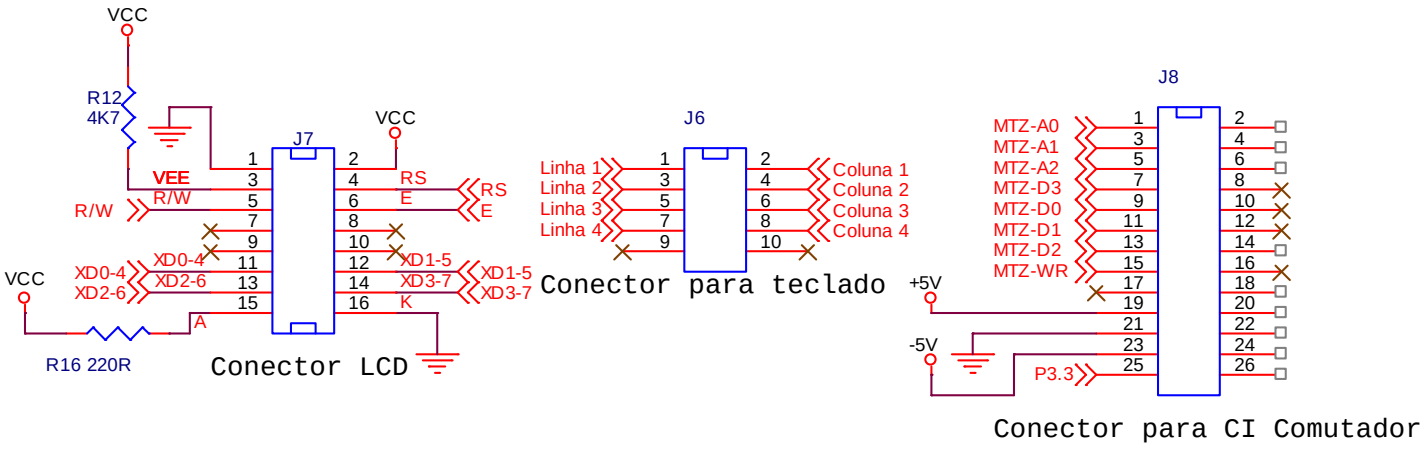
Conversor de níveis lógicos:



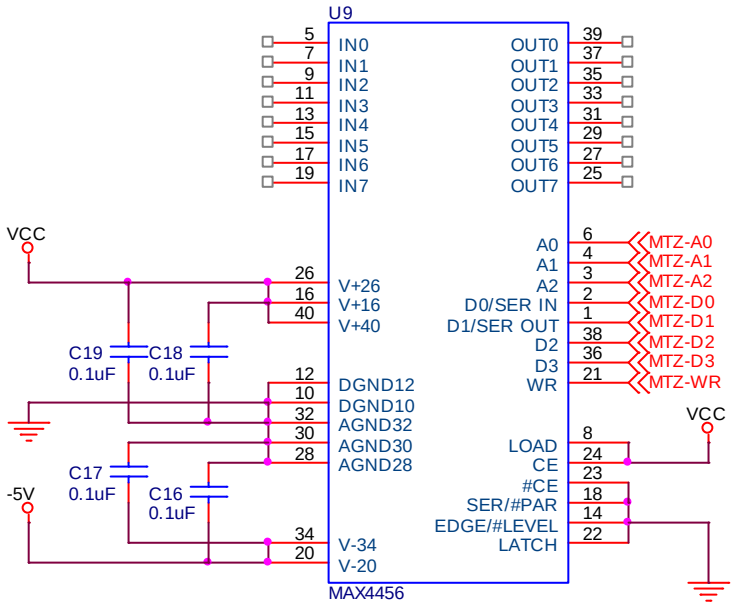
Microcontrolador:



Conectores:



Circuito comutador:



A.2. Código completo do software

```
$MOD51

LINHA4    EQU P0.0
COLUNA1   EQU P0.1
LINHA3    EQU P0.2
COLUNA2   EQU P0.3
COLUNA3   EQU P0.4
LINHA1    EQU P0.5
LINHA2    EQU P0.6

;WRI   EQU P1.7
;D2    EQU P1.6
;D1    EQU P1.5
;D0    EQU P1.4
;D3    EQU P1.3
;A2    EQU P1.2
;A1    EQU P1.1
;A0    EQU P1.0

DB4     EQU P2.0
DB5     EQU P2.1
DB6     EQU P2.2
DB7     EQU P2.3
RS      EQU P2.4
RW      EQU P2.5
EN      EQU P2.6
DADOS   EQU P2

TEMP1       EQU 100     ;tempo para o zero correr de uma linha para outra
TEMP2       EQU 110     ;tempo de espera do lcd
```

```
TEMP3          EQU 72          ;valor para esperar 40 microsegundos

ENTRADA        EQU 30H         ;armazena o valor da entrada
SAIDA          EQU 31H         ;armazena o valor da saida

ENTRADA_1      EQU 32H
ENTRADA_2      EQU 33H
ENTRADA_3      EQU 36H

SAIDA_1                EQU 37H
SAIDA_2                EQU 38H
SAIDA_3                EQU 39H


I_ETAPA                EQU 34H     ;etapa1=escolha da entrada ;etapa2=escolha da saída
                                   ;etapa3=realiza comutação
TECLA_ENTER            BIT 35.0H   ;bit para identificar "enter #"
TECLA_CANCELA          BIT 35.1H   ;bit para identificar "cancela *"
BUF                    BIT 35.2H   ;bit que verifica se buffers estão ou não ligados

ORG    0
     LJMP    INIC

ORG     50H

INIC:   MOV     P0,#0FFH
        CLR     BUF

INIT_LCD:
     MOV       DADOS,#02H      ; function set:  x,e,rw,rs,  db7,db6,db5,db4   = 0000  0010 = 02h
     SETB      EN
     CLR       EN
     LCALL     ESPERA1MS  ; rotina q tenta conciliar o tempo de reset e de power para funcionar
                          ;o lcd
MOV    A,#22H             ; function set:  4 vias, 1 linha, caractere 5x7
     LCALL     MANDA_NIBBLES
     LCALL     ESPERA_LCD
     MOV       A,#0CH          ;display on/off control: display ligado, sem cursor e sem piscar
     LCALL     MANDA_NIBBLES
     LCALL     ESPERA_LCD
     MOV       A,#06h          ; entry mode set: incremento do cursor automático
     LCALL     MANDA_NIBBLES
     LCALL     ESPERA_LCD

        LCALL          LIMPA_LCD

        MOV    I_ETAPA,#1

        MOV    ENTRADA_1,#'X'
        MOV    ENTRADA_2,#'Y'
        MOV    ENTRADA_3,#'Z'
        MOV    SAIDA_1,#'1'
        MOV    SAIDA_2,#'2'
        MOV    SAIDA_3,#'3'
```

```
INICIO:

        MOV   A,#80H
        LCALL         POSICAO
        MOV   ENTRADA,ENTRADA_1
        MOV   SAIDA,SAIDA_1
        LCALL         ESCRITA_1

        MOV   A,#0C0H
        LCALL         POSICAO
        MOV   ENTRADA,ENTRADA_2
        MOV   SAIDA,SAIDA_2
        LCALL         ESCRITA_1

        MOV   A,#94H
        LCALL         POSICAO
        MOV   ENTRADA,ENTRADA_3
        MOV   SAIDA,SAIDA_3
        LCALL         ESCRITA_1

        MOV   A,#0D4H
        LCALL         POSICAO
        LCALL         ESCRITA_2


ESCRITA_1:

        MOV   A,#'E'
      LCALL   ESCREVE_TEXTO
      MOV       A,#'N'
      LCALL   ESCREVE_TEXTO
      MOV       A,#'T'
      LCALL   ESCREVE_TEXTO
      MOV       A,#'R'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'D'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO

        MOV   A,#0FEH                   ;espaço em branco
      LCALL   ESCREVE_TEXTO
        MOV   A,ENTRADA                 ;valor da entrada
      LCALL   ESCREVE_TEXTO
        MOV   A,#0FEH                   ;espaço em branco
      LCALL   ESCREVE_TEXTO
        MOV   A,#7EH                    ; seta
      LCALL   ESCREVE_TEXTO
        MOV   A,#0FEH                   ;espaço em branco
      LCALL   ESCREVE_TEXTO

        MOV   A,#'S'
      LCALL   ESCREVE_TEXTO
```

```
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'I'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'D'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO
        MOV   A,#0FEH                   ;espaço em branco
      LCALL   ESCREVE_TEXTO
        MOV   A,SAIDA
      LCALL   ESCREVE_TEXTO
        RET

ESCRITA_2:
        MOV   A,#'E'
      LCALL   ESCREVE_TEXTO
      MOV     A,#'N'
      LCALL   ESCREVE_TEXTO
      MOV     A,#'T'
      LCALL   ESCREVE_TEXTO
      MOV     A,#'R'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'D'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'A'
      LCALL   ESCREVE_TEXTO
        MOV   A,#'='                    ;sinal de =3DH
      LCALL   ESCREVE_TEXTO
        MOV   A,#'?'                    ;interrogação 3fh
      LCALL   ESCREVE_TEXTO

LE_TECLADO:

LN1:          CLR   LINHA1
              JNB   COLUNA1,L1C1_
              JNB   COLUNA2,L1C2_
              JNB   COLUNA3,L1C3_
              SETB  LINHA1

LN2:          CLR   LINHA2
              JNB   COLUNA1,L2C1_
              JNB   COLUNA2,L2C2_
              JNB   COLUNA3,L2C3_
              SETB  LINHA2

LN3:          CLR   LINHA3
              JNB   COLUNA1,L3C1_
              JNB   COLUNA2,L3C2_
              JNB   COLUNA3,L3C3_
              SETB  LINHA3

LN4:          CLR   LINHA4
```

```
                JNB     COLUNA1,L4C1_
                JNB     COLUNA2,L4C2_
                JNB     COLUNA3,L4C3_
                SETB    LINHA4

                LJMP    LE_TECLADO

L1C1_:          LJMP    L1C1
L1C2_:          LJMP    L1C2
L1C3_:          LJMP    L1C3

L2C1_:          LJMP    L2C1
L2C2_:          LJMP    L2C2
L2C3_:          LJMP    L2C3

L3C1_:          LJMP    L3C1
L3C2_:          LJMP    L3C2
L3C3_:          LJMP    L3C3

L4C1_:          LJMP    L4C1
L4C2_:          LJMP    L4C2
L4C3_:          LJMP    L4C3


L1C1:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'1'
        LJMP    QUAL_ETAPA

L1C2:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'2'
        LJMP    QUAL_ETAPA
L1C3:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'3'
        LJMP    QUAL_ETAPA

L2C1:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'4'
        LJMP    QUAL_ETAPA

L2C2:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'5'
        LJMP    QUAL_ETAPA

L2C3:
        LCALL           ESPERA_REBOTE
        MOV     R0,#'6'
        LJMP    QUAL_ETAPA

L3C1:
        LCALL           ESPERA_REBOTE
```

```
        MOV   R0,#'7'
        LJMP  QUAL_ETAPA

L3C2:
        LCALL         ESPERA_REBOTE
        MOV   R0,#'8'
        LJMP  QUAL_ETAPA

L3C3:
        LCALL         ESPERA_REBOTE
        MOV   R0,#'9'
        LJMP  QUAL_ETAPA

L4C1:
        LCALL         ESPERA_REBOTE
        SETB  TECLA_CANCELA
        LJMP  QUAL_ETAPA

L4C2:
        LCALL         ESPERA_REBOTE
        MOV   R0,#'0'
        LJMP  QUAL_ETAPA

L4C3:
        LCALL         ESPERA_REBOTE
        SETB  TECLA_ENTER
        LJMP  QUAL_ETAPA

QUAL_ETAPA:
        MOV   A,I_ETAPA
        CJNE  A,#1,ETAPA2

ETAPA1:
        JB      TECLA_ENTER,SALVA_ENTRADA
        JB      TECLA_CANCELA,CANCELA
        MOV   A,#0DCH                   ;posiciona em l4/9
        LCALL         POSICAO
        MOV   A,R0
        LCALL  ESCREVE_TEXTO            ;o teclado ja se encarregou de guardar a tecla no
                                        ;acumulador
        MOV   A,#'?'
        LCALL  ESCREVE_TEXTO
        LJMP  LE_TECLADO

ETAPA2:
        JB      TECLA_CANCELA,CANCELA
        JB      TECLA_ENTER,SALVA_SAIDA
        MOV   A,#0E6H                   ;POSICIONA EM L4/19
        LCALL         POSICAO
        MOV   A,R0
        LCALL  ESCREVE_TEXTO
        MOV   A,#'?'
        LCALL  ESCREVE_TEXTO
        LJMP  LE_TECLADO
```

```
CANCELA:
        CLR    TECLA_CANCELA
        CLR    TECLA_ENTER
        MOV    I_ETAPA,#1
        LCALL          LIMPA_LCD
        LJMP   INICIO

SALVA_ENTRADA:
        MOV    ENTRADA,R0          ;salva a entrada
        CLR    TECLA_ENTER
        CLR    TECLA_CANCELA
        MOV    I_ETAPA,#2
        MOV    A,#0DDH             ;posição l4/10
        LCALL          POSICAO

ESCRITA_3:
        MOV    A,#0FEH                     ;espaço em branco
     LCALL   ESCREVE_TEXTO
        MOV    A,#7EH                      ;seta
     LCALL   ESCREVE_TEXTO
        MOV    A,#0FEH                     ;espaço em branco
     LCALL   ESCREVE_TEXTO
        MOV    A,#'S'
     LCALL   ESCREVE_TEXTO
        MOV    A,#'A'
     LCALL   ESCREVE_TEXTO
        MOV    A,#'I'
     LCALL   ESCREVE_TEXTO
        MOV    A,#'D'
     LCALL   ESCREVE_TEXTO
        MOV    A,#'A'
     LCALL   ESCREVE_TEXTO
        MOV    A,#3DH                      ;sinal de =
     LCALL   ESCREVE_TEXTO
        MOV    A,#3FH                      ;interrogação
     LCALL   ESCREVE_TEXTO
        LJMP   LE_TECLADO

SALVA_SAIDA:
        MOV    SAIDA,R0                    ;salva a saida
        CLR    TECLA_ENTER
        CLR    TECLA_CANCELA
        MOV    I_ETAPA,#1
        LCALL          LIMPA_LCD

ESCOLHE_LINHA:
        MOV    A,SAIDA
        CJNE   A,#'3',ALFA2
        MOV    ENTRADA_3,ENTRADA
ALFA2:  CJNE A,#'2',ALFA1
        MOV    ENTRADA_2,ENTRADA
ALFA1:  CJNE A,#'1',COMUTA
        MOV    ENTRADA_1,ENTRADA

        JB     BUF, COMUTA
```

```
        LCALL       BUFFERS_ON

COMUTA:
        MOV   A,ENTRADA
        DEC   A
        SWAP  A
        ORL   A,#0FH
        PUSH  ACC
        MOV   A,SAIDA
        DEC   A
        ORL   A,#0F0H
        MOV   SAIDA,A
        POP   ACC
        ANL   A,SAIDA
        ANL   A,#77H
        MOV   P1,A
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        SETB  P1.7
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        CLR   P1.7
        LCALL       INICIO

BUFFERS_ON:
        MOV   A,#68H        ;WR,D2,D1,D0,D3,A2,A1,A0 = 0110 1000 =Ligar todos os buffers
        MOV   P1,A
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        SETB  P1.7
        CPL   P3.3
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        LCALL ESPERA_LONGA
        CLR   P1.7
        CPL   P3.3
        SETB  BUF
        RET

MANDA_NIBBLES:
        PUSH   ACC
    ORL       DADOS,#0FH
    SWAP   A
    ORL       A,#0F0H
    ANL       DADOS,A
    SETB   EN
    CLR       EN
    POP       ACC
    ORL       DADOS,#0FH
    ORL       A,#0F0H
    ANL       DADOS,A
    SETB   EN
```

```
      CLR      EN
      RET

ESPERA1MS:
        MOV   R7,#50                ;espera 1 milisegundo
L1:   MOV     R6,#230
        DJNZ    R6,$
      DJNZ   R7,L1
      RET

ESPERA4MS:
        MOV   R7,#202               ;espera 4,1 milisegundos
L11:  MOV     R6,#230
        DJNZ    R6,$
        DJNZ    R7,L11
        RET

ESPERA_LCD:
        MOV   R6,#TEMP2             ;espera em torno de 40 microsegundos
        DJNZ    R6,$
        RET

ESPERA_LONGA:
        MOV   R7,#0FFH              ;espera 0,5 segundos
L5:     MOV   R6,#0FFH
L6:     MOV   R5,#0AH
        DJNZ    R5,$
        DJNZ    R6,L6
        DJNZ    R7,L5
        RET

ESPERA_REBOTE:
        MOV   R7,#0FFH
E1:     MOV   R6,#0FFH
E2:     MOV   R5,#0EH
        DJNZ  R5,$
        DJNZ  R6,E2
        DJNZ  R7,E1
        CPL     P1.0
        RET

ESCREVE_TEXTO:
        SETB    RS
        CLR     RW
        LCALL   MANDA_NIBBLES
        LCALL   ESPERA_LCD
        RET

POSICAO:
        CLR     RS
        CLR     RW
        LCALL   MANDA_NIBBLES
        LCALL   ESPERA_LCD
        RET
```

```
LIMPA_LCD:
        CLR     RS
        CLR     RW
        MOV     A,#01H
        LCALL   MANDA_NIBBLES
        LCALL           ESPERA4MS
        RET

END
```