UNIVERSIDADE FEDERAL DA FRONTEIRA SUL – UFFS
Ciência da Computação - 2016/1
Mestre Adriano Sanick Padilha

Circuito Digital/Braille

Kadu Marcos Grando
Adriél Schmitz J. de Paula

Chapecó - SC
2016

Resumo:

Braille é uma forma de leitura com o tato para cegos, desenvolvida por Louis Braille em 1827, sendo ele nascido na França. Essa linguagem nada mais é que um conjunto de pontos em relevo, sendo cada letra uma combinação distinta dos pontos. A partir dos 6 pontos relevantes, é possível fazer 64 combinações diferentes, sendo elas letras simples, acentuadas, pontuações, números, sinais matemáticos e até notas musicais. O projeto principal gira em torno da montagem de um circuito que leia um texto em formato *Ascii* e o converta para o código *Braille*. O resultado será mostrado com ledes, os acesos representarão os relevantes, e os apagados consequentemente os irrelevantes. Foi necessária a verificação do alfabeto inteiro (apenas letras minúsculas) no formato ascii (formato representado em base dois, ou seja, em binário), pois cada letra havia uma forma distinta de ser representada. A princípio foram propostas oito entradas, já que, o número máximo de letras, seriam oito. Porém, percebeu-se que as três primeiras entradas, se repetiam do inicio ao fim, logo, não influenciariam no resultado, então não foram usadas, sendo chamadas de X, Y e Z e as que seriam importantes, de A, B, C, D e E. Como o código *Braille* requer seis pontos distintos, logo era preciso seis possíveis saídas. Após a filtragem de entradas, pôde ser feita a retirada das futuras saídas, sendo elas S0, S1, S2, S3, S4 e S5 (cada S respectivamente representa um lede). Com isso, foi feita a retirada das expressões *booleanas*, utilizando algumas técnicas para tal procedimento. Por fim, foi realizada a montagem dos circuitos, utilizando portas logicas, flip-flops e outros componentes, tudo feito em um software de apoio, o *Proteus.*

PALAVRAS-CHAVE.: CIRCUITOS; ASCII; BRAILLE;

Sumário:

Lista De Figuras:

Lista de Tabelas:

1. Objetivo

Criação de um circuito digital utilizando postas lógicas, flip-flops e que converta letras no formato *ASCII* para o código *Braille*, tendo como representação um conjunto de seis ledes e o software *Proteus* como ferramenta de apoio para a montagem e apresentação.

2. Introdução

O projeto a seguir utiliza-se da aplicação de conhecimentos teóricos adquiridos intra e extraclasse na implementação de um circuito digital que faça a leitura de um texto no formato *Ascii* e o converta para o código *Braille*. Com a construção de uma tabela verdade obteve-se as expressões que representava as futuras saídas (sendo seis no total) e posteriormente encontradas expressões simplificadas através da montagem do mapa de Karnaugh, Mintermos e Maxtermos. Em seguida, desenvolveu-se um circuito modularizado com cinco entradas e seis saídas distintas, através do software *Proteus*. O texto de entrada fora disponibilizado pelos educadores da disciplina, tendo isso, fez-se um sistema que de acordo com a entrada, mostrasse nos ledes a letra equivalente, de acordo com o código *Braille,* sendo oito o número máximo de letras e após a letra formada, o circuito reseta todos os ledes, dando a entender que uma nova palavra seria escrita a partir do primeiro lede.

3. Circuito digital conversor do formato Ascii para o código Braille

No projeto em questão, foram feitas tabelas-verdade, mapas de karnaugh e a utilização da técnica de min e maxtermos, foram elaborados circuitos digitais utilizando decodificadores, portas lógicas e flip-flops, a partir das informações que seriam disponibilizadas pelo docente do curso, tudo modularizado em um software de apoio, o Proteus. Aplicou-se técnicas de simplificação algébrica para a retirada de expressões booleanas e montagem do circuito modularizado no software *Proteus*. Sabendo que o código *Braille* é formado por seis pontos, foram representados tais pontos utilizando ledes, sendo um (aceso) para o ponto com relevo e zero (desligado) simbolizando o ponto irrelevante. Antes de tudo foi preciso pesquisar sobre a tabela Ascii, verificando como era a representação em binário de letra por letra.

Tabela 1.1:

Letra	Código Binário
a	01100001
b	01100010
c	01100011
d	01100100
e	01100101
f	01100110
g	01100111
h	01101000
i	01101001
j	01101010
k	01101011
l	01101100
m	01101101
n	01101110
o	01101111
p	01110000
q	01110001
r	01110010
s	01110011
t	01110100
u	01110101
v	01110110
w	01110111
x	01111000
y	01111001
z	01111010

Fonte: <http://tecciencia.ufba.br/numeros-binarios/midiateca/imagens/transformacao-binaria-09.bmp>

Feita uma tabela verdade para encontrar seis possíveis saídas conforme o funcionamento do código ASCII.

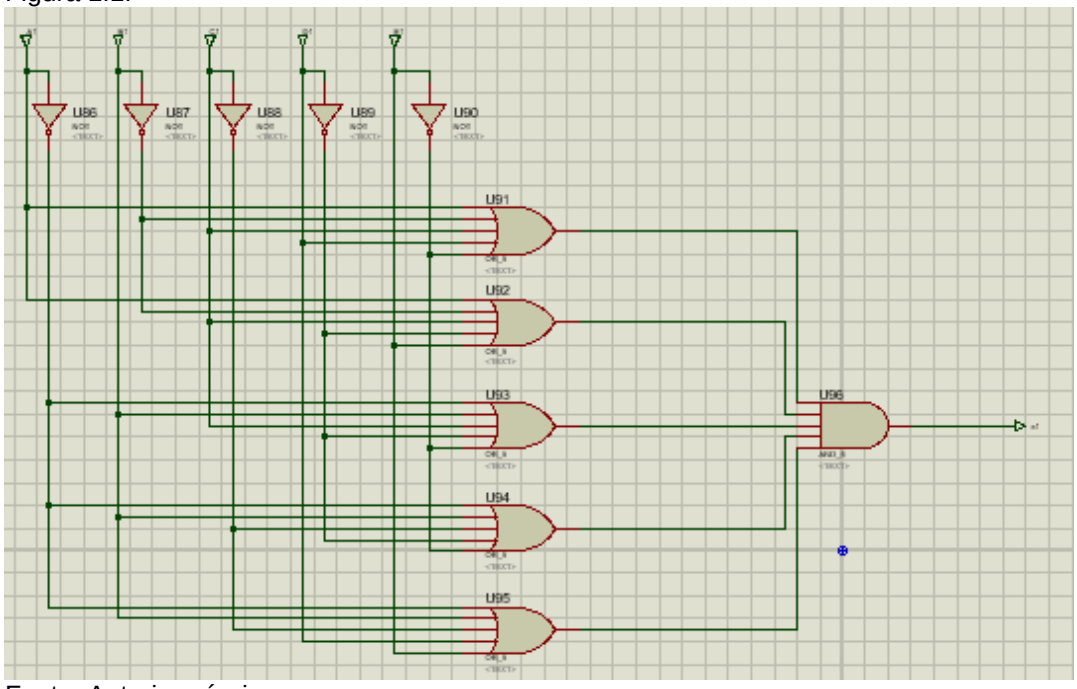
Tabela 1.2:

	Entradas									Saídas					
Letras	X	Y	Z	A	B	C	D	E	Letras	S0	S1	S2	S3	S4	S5
a	0	1	1	0	0	0	0	1	a	1	0	0	0	0	0
b	0	1	1	0	0	0	1	0	b	1	1	0	0	0	0
c	0	1	1	0	0	0	1	1	c	1	0	0	1	0	0
d	0	1	1	0	0	1	0	0	d	1	0	0	1	1	0
e	0	1	1	0	0	1	0	1	e	1	0	0	0	1	0
f	0	1	1	0	0	1	1	0	f	1	1	0	1	0	0
g	0	1	1	0	0	1	1	1	g	1	1	0	1	1	0
h	0	1	1	0	1	0	0	0	h	1	1	0	0	1	0
i	0	1	1	0	1	0	0	1	i	0	1	0	1	0	0
j	0	1	1	0	1	0	1	0	j	0	1	0	1	1	0
k	0	1	1	0	1	0	1	1	k	1	0	1	0	0	0
l	0	1	1	0	1	1	0	0	l	1	1	1	0	0	0
m	0	1	1	0	1	1	0	1	m	1	0	1	1	0	0
n	0	1	1	0	1	1	1	0	n	1	0	1	1	1	0
o	0	1	1	0	1	1	1	1	o	1	0	1	0	1	0
p	0	1	1	1	0	0	0	0	p	1	1	1	1	0	0
q	0	1	1	1	0	0	0	1	q	1	1	1	1	1	0
r	0	1	1	1	0	0	1	0	r	1	1	1	0	1	0
s	0	1	1	1	0	0	1	1	s	0	1	1	1	0	0
t	0	1	1	1	0	1	0	0	t	0	1	1	1	1	0
u	0	1	1	1	0	1	0	1	u	1	0	1	0	0	1
v	0	1	1	1	0	1	1	0	v	1	1	1	0	0	1
w	0	1	1	1	0	1	1	1	w	0	1	0	1	1	1
x	0	1	1	1	1	0	0	0	x	1	0	1	1	0	1
y	0	1	1	1	1	0	0	1	y	1	0	1	1	1	1
z	0	1	1	1	1	0	1	0	z	1	0	1	0	1	1

Nessa tabela-verdade utilizaram-se oito variáveis de entrada que são representadas no código ASCII por determinadas letras, como as três primeiras variáveis são sempre iguais não é necessário utilizá-las futuramente. Feita seis saídas para representar cada um dos ledes, S1 para o primeiro lede, S2 para o segundo e assim sucessivamente.

Para a primeira expressão usou-se Maxtermos:
(*Obs: O "~" antes da letra mostra que ela é barrada. Ex*: **~A = A barrado**).
(A+~B+C+D+~F)*(A+~B+C+~D+F)*(~A+B+C+~D+~F)*(~A+B+~C+~D+~F)*(~A+B+~C+D+F)

Figura 1.1:

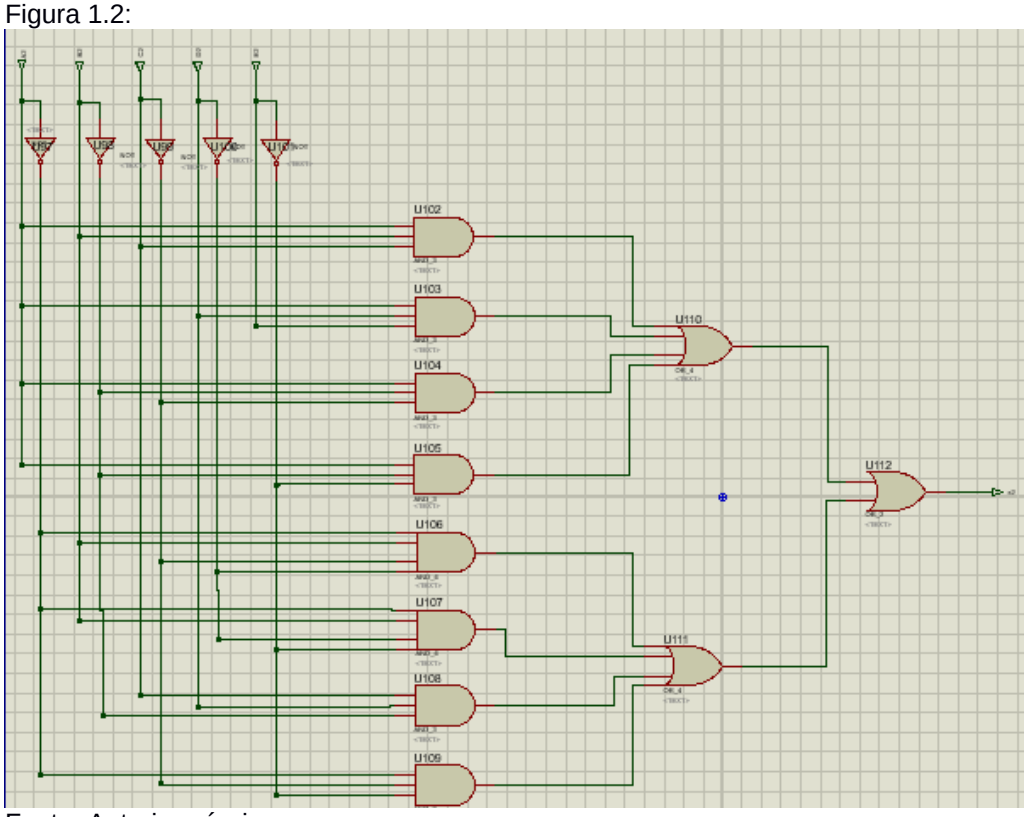


Fonte: Autoria própria.

A segunda expressão também foi utilizada a técnica de Maxtermos e depois feito a simplificação, resultando na seguinte expressão:

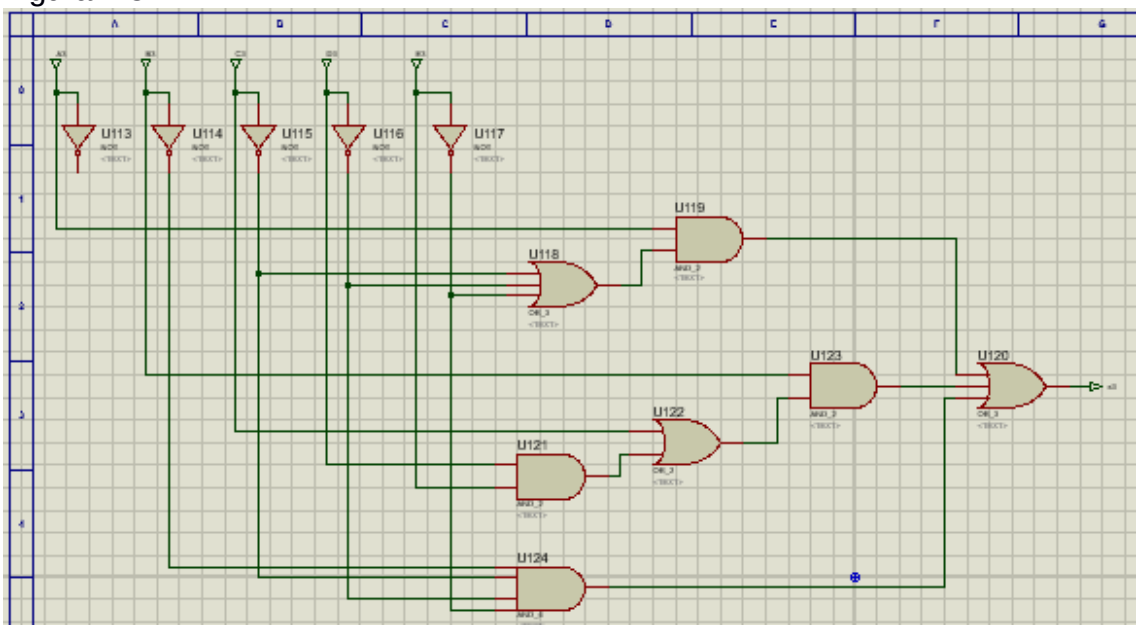
(A*B*C)+(A*D*F)+(A*~B*~C)+(A*~B*~F)+(B*~A*~C*~D)+(B*~A*~D*~F)+(C*D*~B)+(~A*~C*~F)

Figura 1.2:



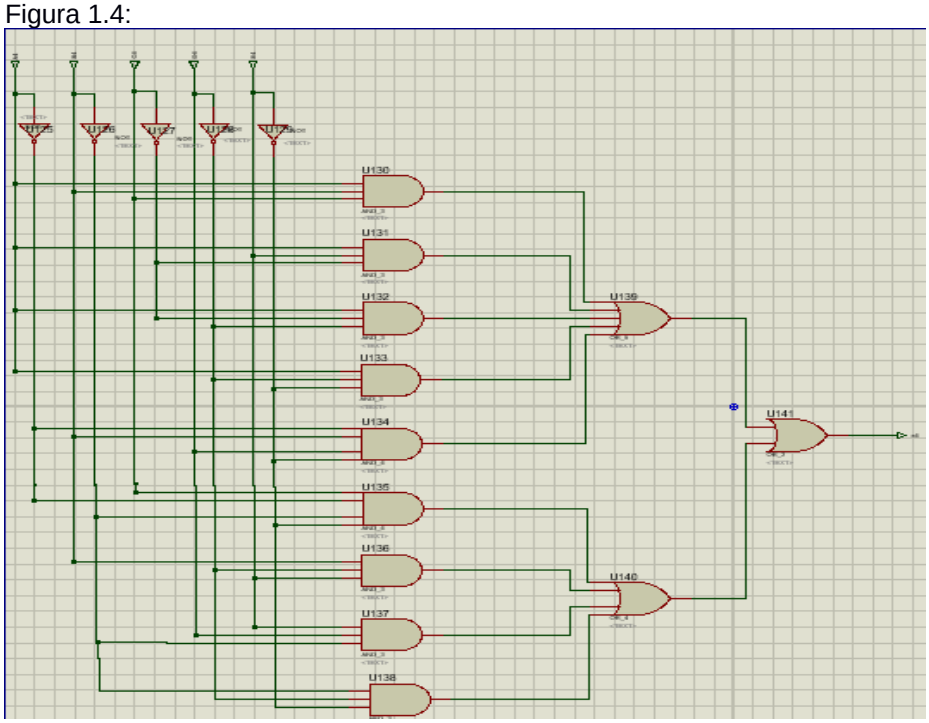
Fonte: Autoria própria.

A terceira expressão foi utilizada maxtermos e feito a simplificação, levando a seguinte expressão: **A*(~C+~D+~F)+B*(C+D*F)+(~B*~C*~D*~F)**
Figura 1.3:



Fonte: Autoria própria.

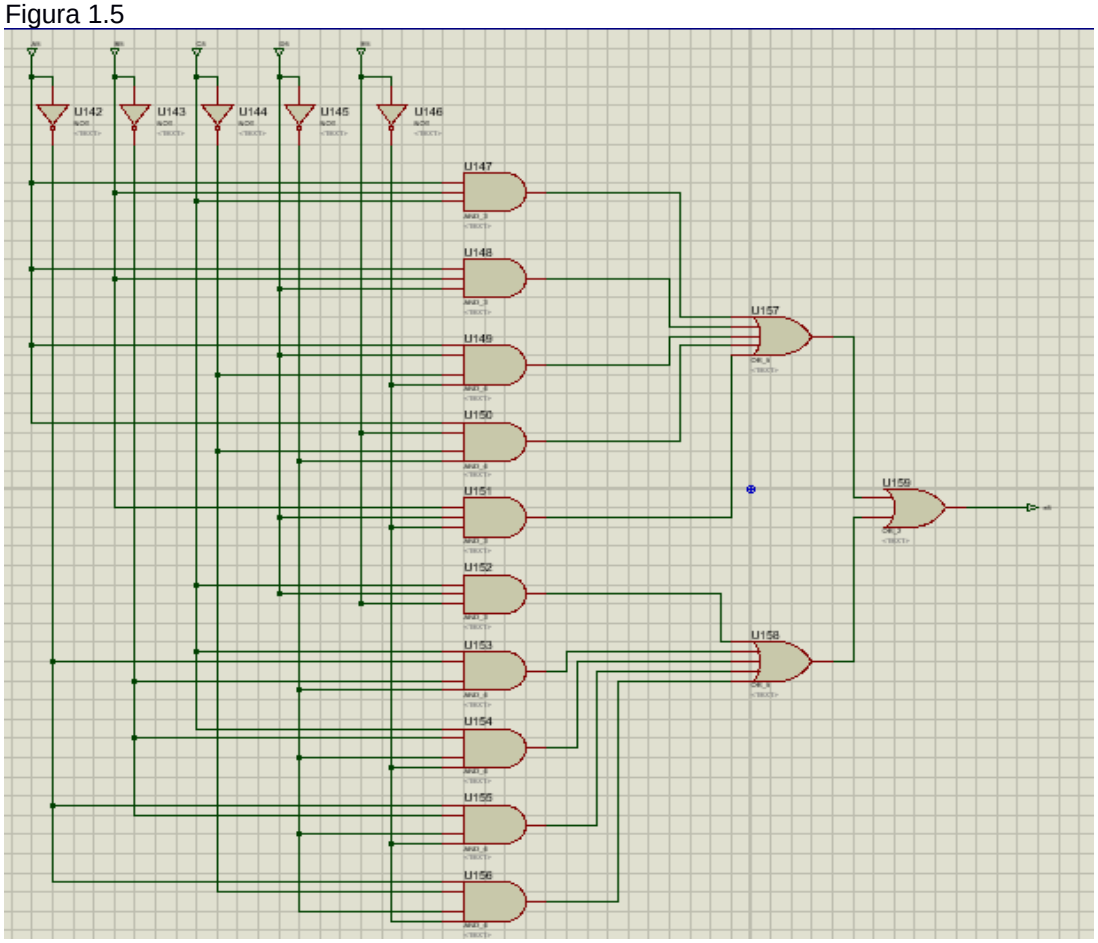
A quarta expressão também foi utilizada Max termos e feito a simplificação gerando a expressão:
(A*B*C)+(A*F*~C)+(A*~C*~D)+(A*~D*~F)+(B*D*~A*~F)+(B*F*~D)+
(C*~A*~B*~F)+(D*F*~B)+(~B*~D*~F)
Figura 1.4:



Fonte: Autoria própria.

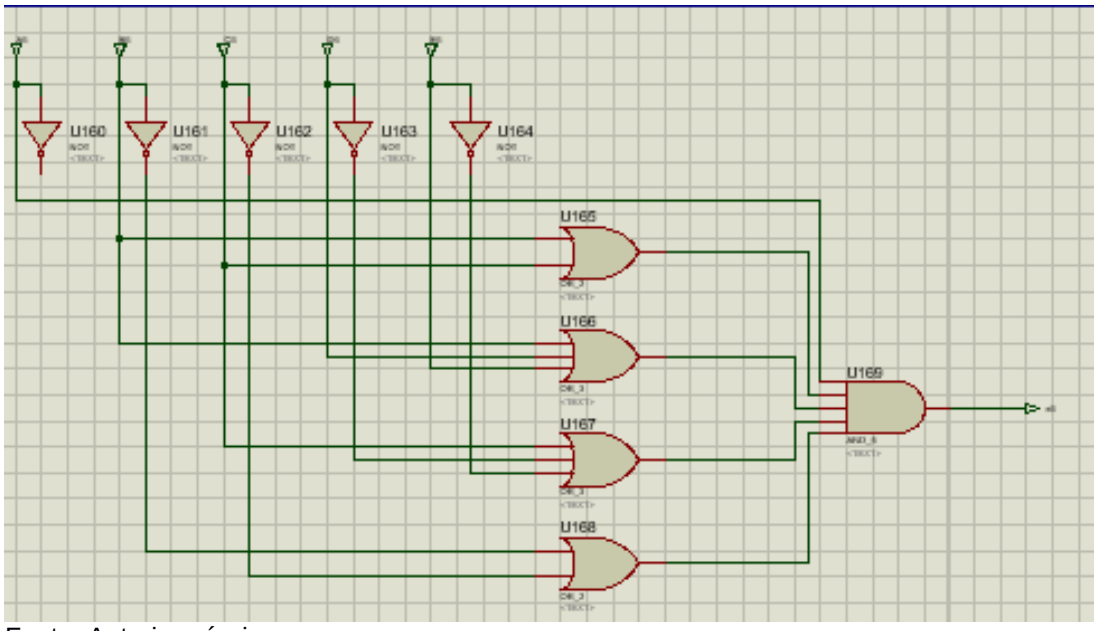
A quinta expressão foi utilizada Maxtermos e também feita a simplificação que resultou na expressão:
(A*B*C) + (A*B*D) + (A*D*~C*~F) + (A*F*~C*~D) + (B*D*~F) + (C*D*F) + (C*~A*~B*~D) + (C*~B*~D*~F) + (~A*~B*~D*~F) + (~A*~C*~D*~F)

Figura 1.5



Fonte: Autoria própria.

A sexta e última expressão foi a única a utilizar a técnica de mintermos e depois foi simplificada gerando a seguinte expressão:
A*(B + C) * (B + D + F) * (C + ~D + ~F) * (~B + ~C)
Figura 1.6:

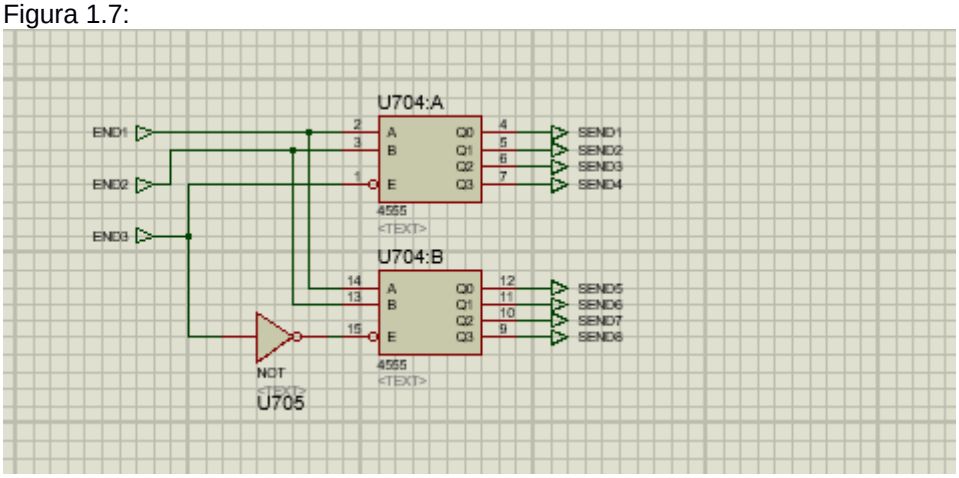


Fonte: Autoria própria.

Uma vez obtida todas as expressões necessárias montadas no proteus, foi feito o teste e o resultado obtido foi à frase "***Se você puder me olhar***". Agora havia a necessidade de transformar as expressões em um "display" de oito letras.

Para tanto, criou-se um *decoder* para fazer o endereçamento do display.

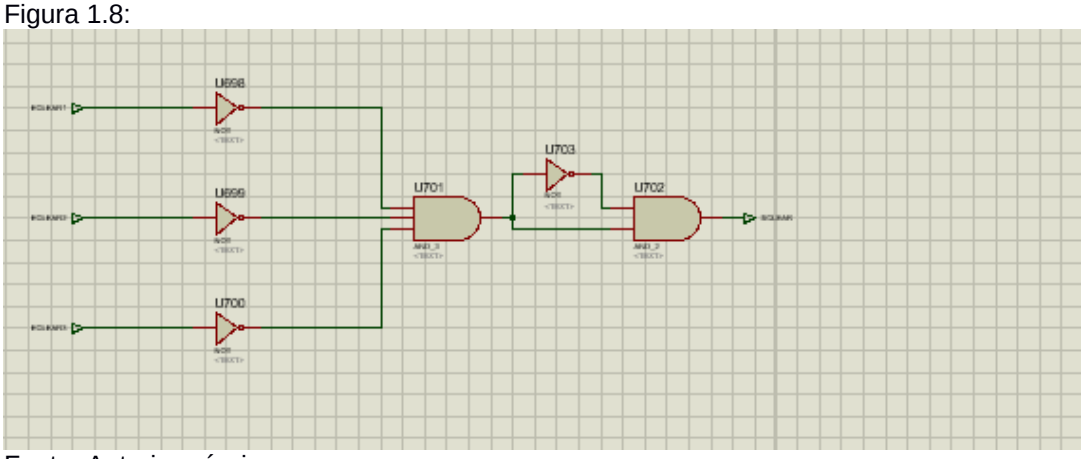
Figura 1.7:



Fonte: Autoria própria.

Um clear foi criado para fazer a limpeza dos ledes quando o endereço recebido fosse 000.

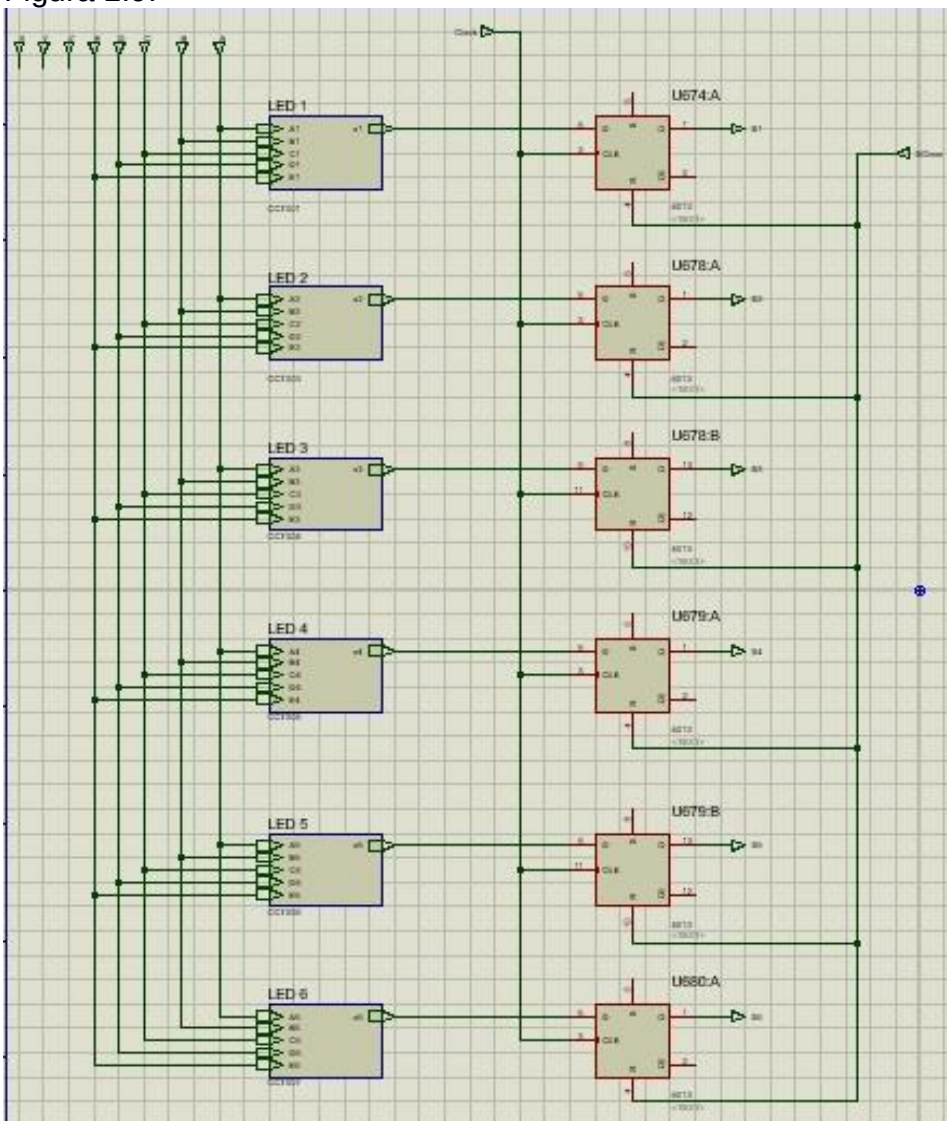
Figura 1.8:



Fonte: Autoria própria.

Pra montar a estrutura das letras foi utilizado também um flip-flop do tipo D, que recebe o Clock (decoder) e o clear, segue a estrutura das letras:

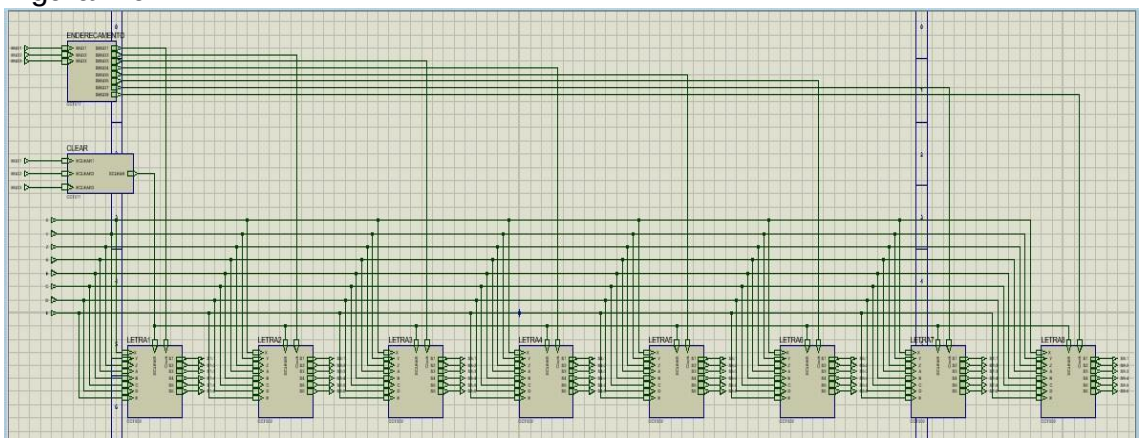
Figura 1.9:



Fonte: Autoria própria.

Após a criação da estrutura das letras, o decoder e o clear, foi testado novamente e organizado os fios de ligamento da “caixa mãe”.

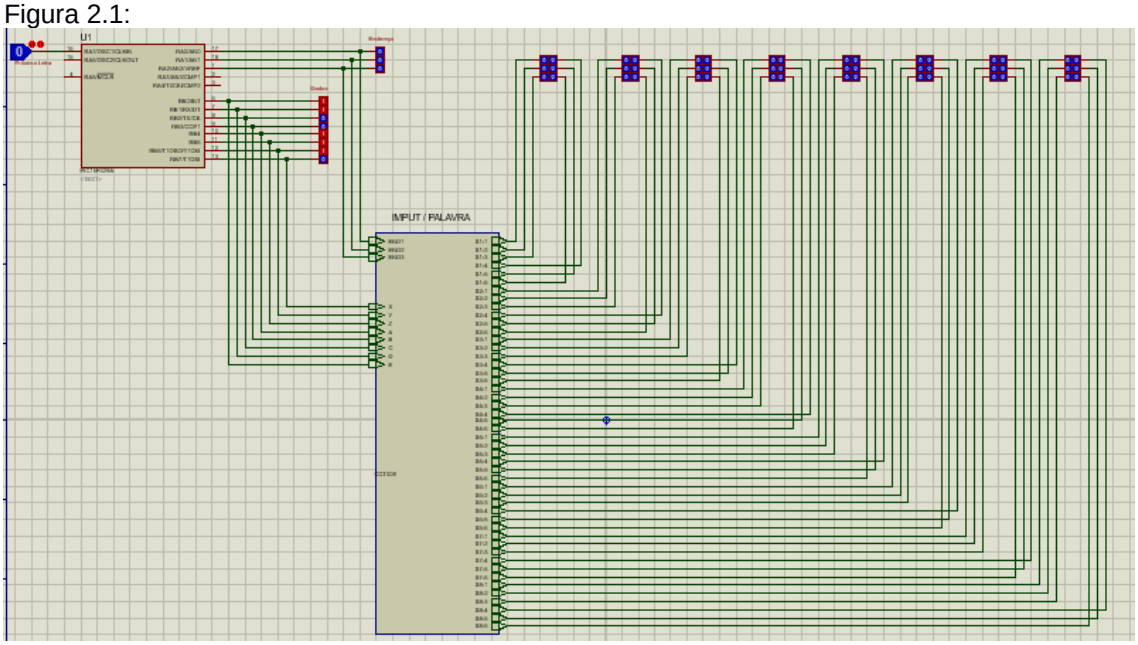
Figura 2.0:



Fonte: Autoria própria.

Visão inicial dos circuitos, mostrando a parte disponibilizada pelo professor, a caixa de acoplamento da parte de tratamento de cada letra e também os oitos conjuntos de seis ledes.

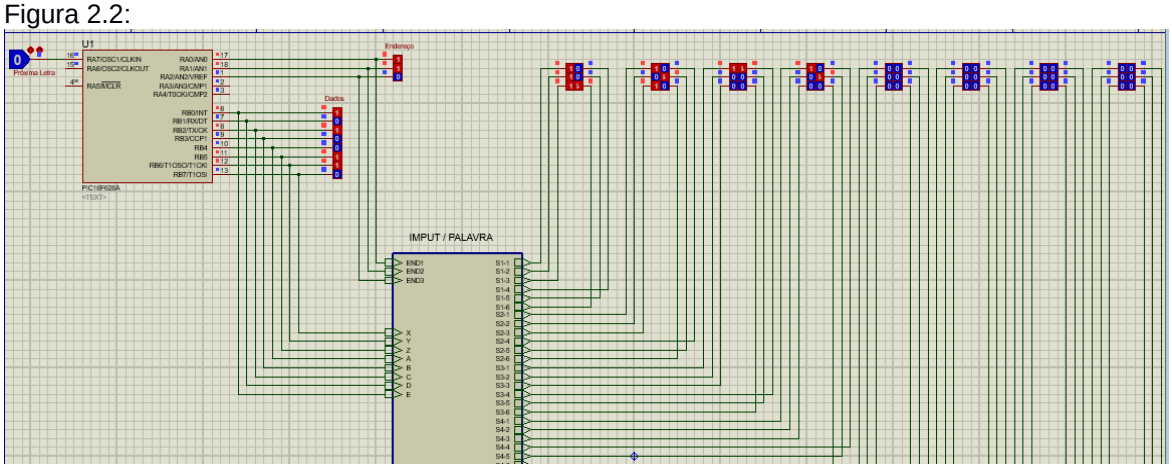
Figura 2.1:



Fonte: Autoria própria.

Exemplificação de formação da palavra “voce”.

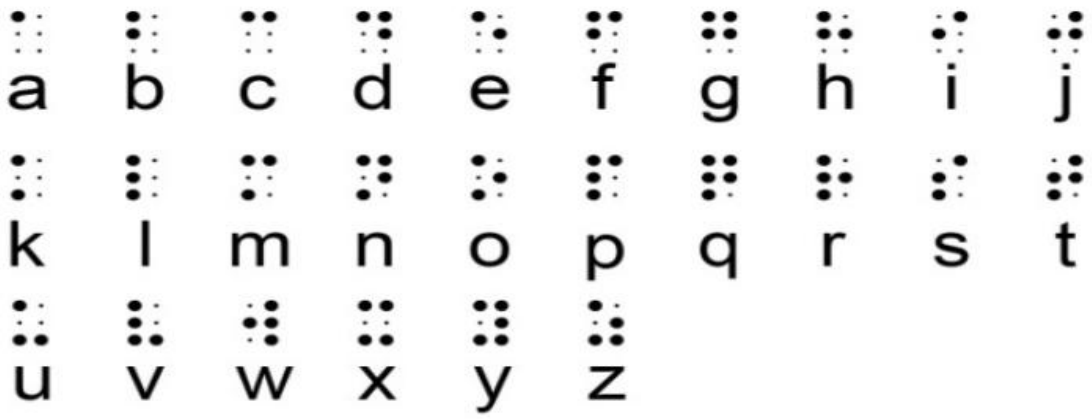
Figura 2.2:



Fonte: Autoria própria.

Sendo possível a verificação do resultado na imagem abaixo.

Figura 2.3:



Fonte: <http://www.projetoacesso.org.br/site/images/Screen%20Shot%202012-12-06%20at%204.46.41%20PM.png>

Sistema de blocos

Nesse pequeno esquema, é possível ter uma ideia resumida de como foi elaborado o trabalho.

1) Elaboração da tabela verdade
2) Simplificação dos bits não significativos
3) Retirada das expressões, incluindo suas simplificações.
4) Montagem e teste do circuito no Proteus.
5) Organização visual no Proteus.

4. Discussão e Considerações finais

Com a realização deste projeto pode-se concluir que todo e qualquer circuito necessita de total atenção desde o princípio, qualquer descuido pode acarretar grandes problemas e muitas vezes se gasta muito tempo até solucioná-los. Fez-se mais que importante à análise de cada bit de acordo com sua importância, sendo que caso um bit seja escolhido para ser utilizado de forma errônea, as suas mudanças de estado acabaram interferindo diretamente no resultado final. Vale a pena ressaltar também, os problemas apresentado pelo software de apoio, muitas vezes não salvava algumas partes, fazendo com que trabalhos tenham sido em vão. A ajuda dos monitores foi de extrema importância, auxiliando principalmente na parte de flip-flops e modularização dos componentes. Por fim, o conteúdo produzido ao final foi satisfatório, atendendo o objetivo inicial, o circuito deixa uma porta para algo futuro, como a implementação de mais letras, acentuações e palavras maiúsculas de acordo com o padrão Braille.

5. Referências bibliográficas

Curso do software Proteus. Disponível em:
< https://www.youtube.com/watch?v=BN4eg7eg0l0>. Acesso em: 10 de junho de 2016.

Micro Controlandos. **Proteus 8.1** Disponível em:
< http://microcontrolandos.blogspot.com.br/2013/04/proteus-8-professional.html>. Acesso em: 14 de junho de 2016.

Digital Electronics Course Outline. **Boolean Algebra.** Disponível em: <http://electronics-course.com/boolean-algebra>. Acesso em: 22 de junho de 2016.