**Relatório Trabalho Prático de Circuitos Digitais - Etapa 2 –Decodificador BCD**

**Lucas Nunes Santana, João Vitor Moreira de Sousa.**

Engenharia da Computação - UniEvangélica - Centro Universitário - Anápolis, GO - Brasil

[nlucas00043@gmail.com](mailto:nlucas00043@gmail.com), jvmoreirasousa@hotmail.com

***Resumo.*** *Este relatório apresenta uma abordagem a respeito da teoria e lógica utilizada para o desenvolvimento do trabalho prático de Circuitos Digitais sobre o* *Decodificador BCD com Display 7 Segmentos.*

***Palavras-Chave:*** *Decodificador BCD, Arduino, Circuitos Digitais.*

1. **Fundamentação Teórica**

As funções lógicas podem ser encontradas em apenas 2 estados distintos:

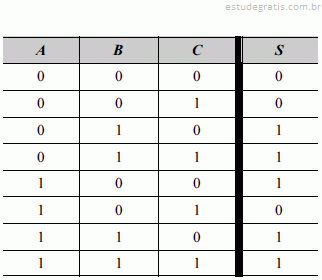
Estado 0 (zero) e Estado 1 (um).

O estado 0 representa o não ou o falso, em outras palavras significa a ausência ou a negação de algo.

O estado 1 representa o sim ou o verdadeiro, de forma sucinta o 1 é o contrário do 0.

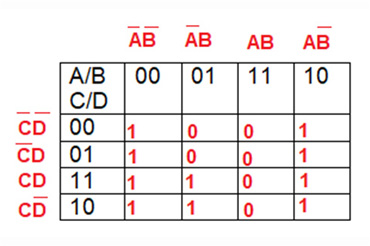
O Trabalho consiste em desenvolver um contador hexadecimal através de uma tabela verdade e dos mapas de Karnaugh. Antes de especificar a parte lógica de como foi feito esse contador é necessário os conceitos de tabela verdade e mapas de Karnaugh previamente.

A Tabela verdade é um dispositivo utilizado no estudo da lógica matemática. Com o uso desta tabela é possível definir o valor lógico de uma proposição, isto é, saber quando uma sentença é verdadeira ou falsa.



Os mapas de Karnaugh é um método gráfico usado para simplificar uma equação lógica ou converter uma tabela verdade no seu circuito lógico correspondente

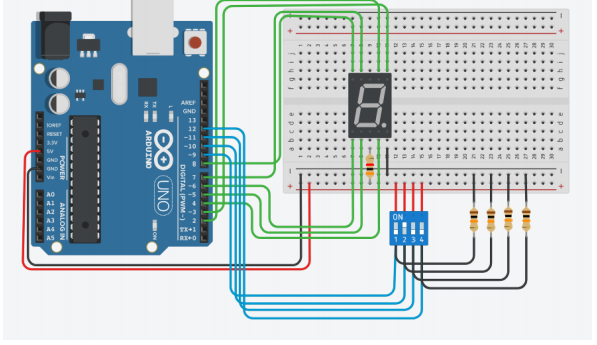
O método de leitura por "Mapa de Karnaugh" é considerado mais simples que a [álgebra booleana](https://pt.wikipedia.org/wiki/%C3%81lgebra_booleana), pois elimina o problema de erro nas simplificações. Porém quando utilizado mais de 6 entradas, esse método se torna complicado, pois fica difícil identificar as células adjacentes no mapa.



1. **Aplicação Prática**

Como foi explicado previamente o trabalho consiste em realizar um Decodificador BCD na prática utilizando Arduino. Para isso foi utilizada a linguagem de programação C e o simulador de projetos/circuitos denominado TinkerCad.

Primeiramente foi criado o circuito abaixo:



Após a criação desse circuito foi necessário fazer uma tabela verdade cujas entradas são (ABCD) e saídas (a, b, c, d, e, f, g).

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | a | b | c | d | e | f | g |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 |
| 3 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 |
| 4 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 5 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 1 |
| 6 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 7 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 |
| 8 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 9 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |
| a | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 1 | 1 |
| b | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| c | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 |
| d | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 |
| e | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |
| f | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |

Depois disso é necessário descobrir a expressão matemática de cada saída através da simplificação utilizando mapas de Karnaugh.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segA | 00 | 01 | 11 | 10 |
| 00 | **1** | **0** | **1** | **1** |
| 01 | **0** | **1** | **1** | **0** |
| 11 | **0** | **0** | **1** | **1** |
| 10 | **1** | **1** | **0** | **1** |

segA = (!A . !B . !D) + (!A . !B . C) + (!A . B . D) + (B . C . D) + (A . !B . !C) + (A . C . !D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segB | 00 | 01 | 11 | 10 |
| 00 | **1** | **1** | **1** | **1** |
| 01 | **1** | **0** | **1** | **0** |
| 11 | **0** | **1** | **0** | **0** |
| 10 | **1** | **1** | **0** | **1** |

segB = (!A . !B) + (!B . !C) + (!A . !C . !D) + (A . !C . D) + (!A . C . D) + (!B . C . !D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segC | 00 | 01 | 11 | 10 |
| 00 | **1** | **1** | **1** | **0** |
| 01 | **1** | **1** | **1** | **1** |
| 11 | **0** | **1** | **0** | **0** |
| 10 | **1** | **1** | **1** | **1** |

segC = (!A . !C) + (!A . D) + (!A . B) + (!C . D) + (A . !B)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segD | 00 | 01 | 11 | 10 |
| 00 | **1** | **0** | **1** | **1** |
| 01 | **0** | **1** | **0** | **1** |
| 11 | **1** | **1** | **0** | **1** |
| 10 | **1** | **1** | **1** | **0** |

segD = (A . !C) + (!B . !C . !D) + (B . !C . D) + (!B . C . D) + (!A . C . !D) + (B . C . !D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segE | 00 | 01 | 11 | 10 |
| 00 | **1** | **0** | **0** | **1** |
| 01 | **0** | **0** | **0** | **1** |
| 11 | **1** | **1** | **1** | **1** |
| 10 | **1** | **0** | **1** | **1** |

segE = (A . B) + (C . !D) + (A . C) + (A . !D) + (!A . !B . !D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segF | 00 | 01 | 11 | 10 |
| 00 | **1** | **0** | **0** | **0** |
| 01 | **1** | **1** | **0** | **1** |
| 11 | **0** | **0** | **1** | **1** |
| 10 | **1** | **1** | **1** | **1** |

segF = (A . !B) + (A . C) + (!A . !C . !D) + (!A . B . !C) + (B . C . !D)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| (saída) segG | 00 | 01 | 11 | 10 |
| 00 | **0** | **0** | **1** | **1** |
| 01 | **1** | **1** | **0** | **1** |
| 11 | **1** | **1** | **1** | **1** |
| 10 | **1** | **1** | **1** | **1** |

segG = (A) + (B . !C) + (!B . C) + (C . !D)