



DEPARTAMENTO DE ELÉCTRICA Y ELECTRÓNICA
INGENIERÍA EN ELECTRÓNICA Y TELECOMUNICACIONES

“CIRCUITOS DIGITALES”



PRODUCTO DE UNIDAD

TEMA:

APLICACIONES DE COMPUERTAS LÓGICAS

AUTORES:

HUGO DAVID ANDRADE SOLORIZANO

JAMES MARSHALL FLORES PROAÑO

DOCENTE:

ING. DARWIN OMAR ALULEMA FLORES

NRC: 9434

Quito - Ecuador, 22 de junio de 2020

1. PLANTEAMIENTO DEL PROBLEMA

La detección de errores en códigos binarios es muy importante para saber identificar si existe algún fallo en sistemas de comunicación digital, los cuales pueden sufrir alteraciones, por diferentes fenómenos físicos, al momento de la transmisión de información que parte desde un transmisor hacia un receptor. Es por ello que nuestro propósito en este trabajo de investigación es realizar un programa (app) en el cual se pueda realizar la detección de errores en números binarios sea cualquier longitud que se nos sea presentada, a través de la paridad par o impar, en este programa debe ser implementado un algoritmo propio para realizar la detección del error de cualquier número binario, para realizar este programa no se admite el uso de librerías que ya se encuentren preestablecidas, todo esto con el uso del software online propuesto que es MIT APP INVENTOR 2.

2. OBJETIVOS

2.1.OBJETIVO GENERAL

Diseñar e implementar circuitos lógicos aplicando todo lo relacionado a compuertas lógicas, las cuales nos ayudan a través de sus condiciones a solucionar problemas o las necesidades que estos mismos presenten. Todo esto mediante softwares de simulación de los diagramas lógicos y además de un laboratorio virtual.

2.2.OBJETIVOS ESPECÍFICOS

- Investigar sobre compuertas lógicas y el uso de estas mismas, en especial las compuertas universales NOR las cuales son necesarias para armar los circuitos pedidos en el 3er literal del enunciado, además de la indagación sobre mapas de Karnaugh y algebra de Boole que son necesarios para realizar la correcta simplificación del circuito, sus propiedades y teoremas.

- Buscar artículos relacionados con el tema de producto de unidad y comparar estos cuáles su relación con el proyecto además de las compuertas lógicas que serán usadas para la implementación dentro de los circuitos a construir
- Realizar 3 circuitos cada uno con sus respectivas indicaciones y los requisitos que se piden para cada uno de estos.

3. ESTADO DEL ARTE

Una familia de compuertas lógicas de transistores de un solo electrón y su aplicación.

Parte I: componentes básicos para lógica binaria, de valores múltiples y de modo mixto.

K. Degawa, T. Aoki, T. Higuchi, H. Inokawa y A. Takahashi, "*Una familia de compuertas lógicas de transistores de un solo electrón y su aplicación - Parte I: componentes básicos para lógica binaria, de valores múltiples y de modo mixto* " Actas. 34° Simposio internacional sobre lógica de valores múltiples, Toronto, Ontario, Canadá, 2004.

En este artículo nos presenta un estudio basado en modelos de una familia de compuertas lógicas SET para sintetizar los circuitos lógicos que son de forma binaria y MV que son valores múltiples. El uso de estas compuertas lógicas SET al combinarse con transistores MOS el cual nos va a permitir una realización compacta de las funciones básicas que conocemos las cuales exhiben características de transferencias periódicas.

Estas compuertas lógicas básicas SET son útiles para implementar circuitos lógicos binarios, circuitos lógicos MV y los circuitos lógicos mixtos (Binarios-MV) de una manera más flexible un ejemplo es que es este documento que describe el diseño con varios contadores paralelo para la aritmética libre de programación de transporte, donde las señales MV se utilizan de una manera más efectiva y lograr una mayor funcionalidad y un hardware no tan complejo

Un diseño novedoso de compuerta lógica óptica AND, OR y NOT con tubo nano de carbono de pared múltiple de cloruro de polivinilo

M. Desani, A. Bavarva y V. Sorathiya, "*Un diseño novedoso de compuerta lógica óptica AND, OR y NOT with Polyvinyl-Chloride Multiwalled Carbon Nano Tube*", Conferencia Internacional 2015 sobre Avances en Informática, Comunicaciones e Informática (ICACCI) , Kochi, 2015.

En este artículo se nos muestra que para realizar un cómputo más rápido que sea de mayor velocidad tanto en el cómputo como en la velocidad de los datos en circuitos digitales los circuitos fotonicos son una de las soluciones a esto.

En este se representara un diseño de ND,OR y NO compuerta lógica optima pero con una guía de onda de tubo de nano carbono multicapa de cloruro de polivinilo abreviado PVC-MWCNT, este material fue elegido para una respuesta más óptica constante en longitudes de onda que sean en un rango de 800nm a 1000nm.

Aquí las puertas lógicas se investigan con el método de dominio de tiempo de diferenciación finita (FDTD además que también se informan sobre las comparaciones con los diseños de compuertas lógicas

Todas las compuertas lógicas son compatibles para una mayor expansión de los circuitos lógicos, una de las funciones booleanas está formateada por las puertas lógicas propuestas y analiza el trabajo de investigación

Puertas lógicas experimentales AND y OR con MZI y SOA mediante modulación PAM

DNS Cavalcante "*Puertas lógicas experimentales AND y OR con MZI y SOA utilizando modulación PAM*", en IEEE Photonics Technology Letters , vol. 31, no. 1. 12 de noviembre de 2018

En este artículo primero se nos presenta una de las principales limitaciones de los sistemas de comunicación óptica la cual es la necesidad actual de conversiones optoelectrónicas en

dispositivos intermedios las cuales no alcanzan las velocidades de transmisión como la de las fibras ópticas de terabits por segundo. En este trabajo, investigamos la obtención de puertas lógicas a través de una configuración de un interferómetro Mach-Zehnder con un amplificador óptico de semiconductores (SOA) como elemento desequilibrante del interferómetro. Estudiamos experimentalmente tres escenarios diferentes de potencia de entrada para un régimen de onda continua y un escenario para el régimen PAM, ambos con niveles variables de ganancia de SOA. Como resultado, obtuvimos las compuertas lógicas AND y OR en todos los escenarios en un amplio rango de ganancia de SOA, además de las funciones lógicas 0 y 1 en algunos de los casos.

Confirmamos la calidad de estas puertas lógicas por la tasa de precisión con respecto al umbral de 0.4 dB por encima y por debajo del límite central, de acuerdo con las recomendaciones de la literatura. Destacamos que la técnica utilizada en este experimento es mucho menos compleja que la mayoría de las descritas en la literatura. El uso de la modulación de amplitud de pulso (PAM) sugiere una forma más simple de controlar las puertas lógicas, una vez que podamos obtenerlas simplemente cambiando la corriente de inyección aplicada por el SOA. La obtención experimental de tales puertas lógicas es relevante para los circuitos lógicos totalmente ópticos que se proponen para optimizar los sistemas actuales de comunicación óptica. Destacamos que la técnica utilizada en este experimento es mucho menos compleja que la mayoría de las descritas en la literatura. El uso de la modulación de amplitud de pulso (PAM) sugiere una forma más simple de controlar las puertas lógicas, una vez que podamos obtenerlas simplemente cambiando la corriente de inyección aplicada por el SOA. La obtención experimental de tales puertas lógicas es relevante para los circuitos lógicos totalmente ópticos que se proponen para optimizar los sistemas actuales de comunicación óptica. Destacamos que la técnica utilizada en este experimento es mucho menos compleja que la mayoría de las descritas en la literatura. El uso de la modulación de amplitud de pulso (PAM) sugiere una

forma más simple de controlar las puertas lógicas, una vez que podamos obtenerlas simplemente cambiando la corriente de inyección aplicada por el SOA. La obtención experimental de tales puertas lógicas es relevante para los circuitos lógicos totalmente ópticos que se proponen para optimizar los sistemas actuales de comunicación óptica.

4. MARCO TEÓRICO

Compuertas lógicas

Compuertas lógicas Las Compuertas Lógicas son circuitos electrónicos conformados internamente por transistores que se encuentran con arreglos especiales con los que otorgan señales de voltaje como resultado o una salida de forma booleana, están obtenidos por operaciones lógicas binarias (suma, multiplicación). También niegan, afirman, incluyen o excluyen según sus propiedades lógicas. Estas compuertas se pueden aplicar en otras áreas de la ciencia como mecánica, hidráulica o neumática.

Existen diferentes tipos de compuertas y algunas de estas son más complejas, con la posibilidad de ser simuladas por compuertas más sencillas. Todas estas tienen tablas de verdad que explican los comportamientos en los resultados que otorga, dependiendo del valor booleano que tenga en cada una de sus entradas.

Trabajan en dos estado, «1» o «0», los cuales pueden asignarse a la lógica positiva o lógica negativa.

El estado 1 tiene un valor de 5v como máximo y el estado 0 tiene un valor de 0v como mínimo y existiendo un umbral entre estos dos estados donde el resultado puede variar sin saber con exactitud la salida que nos entregara. Las lógicas se explican a continuación:

La lógica positiva es aquella que con una señal en alto se acciona, representando un 1 binario y con una señal en bajo se desactiva. representado un 0 binario.

La lógica negativa proporciona los resultados inversamente, una señal en alto se representa con un 0 binario y una señal en bajo se representa con un 1 binario.

A continuación, vamos a analizar las diferentes operaciones lógicas una por una comenzando por la más simple:

Compuerta AND

Esta compuerta es representada por una multiplicación en el Algebra de Boole. Indica que es necesario que en todas sus entradas se tenga un estado binario 1 para que la salida otorgue un 1 binario. En caso contrario de que falte alguna de sus entradas con este estado o no tenga si quiera una accionada, la salida no podrá cambiar de estado y permanecerá en 0. Esta puede ser simbolizada por dos o más interruptores en serie de los cuales todos deben estar activos para que esta permita el flujo de la corriente.

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1



Compuerta OR

En el Algebra de Boole esta es una suma. Esta compuerta permite que con cualquiera de sus entradas que este en estado binario 1, su salida pasara a un estado 1 también. No es necesario que todas sus entradas estén accionadas para conseguir un estado 1 a la salida pero tampoco causa algún inconveniente. Para lograr un estado 0 a la salida, todas sus entradas deben estar en el mismo valor de 0. Se puede interpretar como dos interruptores en paralelo, que sin importar cual se accione, será posible el paso de la corriente.

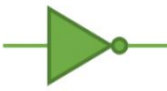
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1



Compuerta NOT

En este caso esta compuerta solo tiene una entrada y una salida y esta actúa como un inversor. Para esta situación en la entrada se colocara un 1 y en la salida lo que nos dara como resultado un 0 y en el caso contrario esta recibirá un 0 y mostrara un 1. Por lo cual todo bit que entre su salida será el inverso

Q	Q'
0	1
1	0

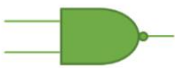


$$Q = \bar{Q}$$

Compuerta NAND

También denominada como AND negada, esta compuerta trabaja al contrario de una AND ya que al no tener entradas en 1 o solamente alguna de ellas, esta concede un 1 en su salida, pero si esta tiene todas sus entradas en 1 la salida se presenta con un 0.

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0




$$Q = \overline{A * B}$$

Compuerta NOR

Así como vimos anteriormente, la compuerta OR también tiene su versión inversa. Esta compuerta cuando tiene sus entradas en estado 0 su salida estará en 1, pero si alguna de sus entradas pasa a un estado 1 sin importar en qué posición, su salida será un estado 0.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0



$$Q = \overline{A + B}$$

Compuerta XOR

También llamada OR exclusiva, esta actúa como una suma binaria de un dígito cada uno y el resultado de la suma sería la salida. Otra manera de verlo es que con valores de entrada igual el estado de salida es 0 y con valores de entrada diferente, la salida será 1.

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	0



$$Q = A * \bar{B} + \bar{A} * B$$

Compuerta XNOR

Esta es todo lo contrario a la compuerta XOR, ya que cuando las entradas sean iguales se presentara una salida en estado 1 y si son diferentes la salida será un estado 0.

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	1



$$Q = A * B + \bar{A} + \bar{B}$$

Compuerta IF

Esta compuerta no es una muy utilizada o reconocida ya que su funcionamiento en estados lógicos es parecido a si solo hubiera un cable conectado porque exactamente lo que se le coloque en la entrada, se encontrara en la salida. Pero también es conocido como un buffer, en la práctica se utiliza como amplificador de corriente o como seguidor de tensión para adaptar impedancias.

Q	Q'
0	0
1	1



$$Q = Q$$

Algebra de Boole

En 1815 George Boole propuso una herramienta matemática llamada Algebra de Boole .

Luego en 1938 Claude Shannon propuso que con esta álgebra es posible modelar los llamados Sistemas Digitales.

El Algebra de Boole ´ es un sistema matemático que utiliza variables y operadores lógicos. Las variables pueden valer 0o 1. Y las operaciones básicas son OR(+) y AND(·).

Luego se definen las expresiones de conmutación como un número finito de variables y constantes, relacionadas mediante los operadores (AND y OR).

En la ausencia de paréntesis, se utilizan las mismas reglas de precedencia, que tienen los operadores suma (OR) y multiplicación (AND) en el álgebra normal

En el álgebra de Boole se cumplen las siguientes Leyes:

1) Conmutatividad:

$$X + Y = Y + X$$

$$X \cdot Y = Y \cdot X$$

2) Asociatividad:

$$X + (Y + Z) = (X + Y) + Z$$

$$X \cdot (Y \cdot Z) = (X \cdot Y) \cdot Z$$

3) Distributividad:

$$X + (Y \cdot Z) = (X + Y) \cdot (X + Z)$$

$$X \cdot (Y + Z) = (X \cdot Y) + (X \cdot Z)$$

4) Elementos Neutros (Identidad):

$$X + 0 = X$$

$$X \cdot 1 = X$$

5) Complemento:

$$X + X' = 1$$

$$X \cdot X' = 0$$

6) Dominación:

$$X + 1 = 1 \quad X \cdot 0 = 0$$

Demostración:

$$X + 1 = (X + 1) \cdot 1 = (X + 1) \cdot (X + X')$$

$$(X + 1) \cdot (X + X') = X + (1 \cdot X') = 1$$

7) Idempotencia:

$$X + X = X$$

$$X \cdot X = X$$

8) Doble complemento:

$$X'' = X$$

9) Absorción:

$$X + X \cdot Y = X$$

$$X \cdot (Y + X) = X$$

Demostración:

$$X + X \cdot Y = (X \cdot 1) + (X \cdot Y) = X \cdot (1 + Y) = X$$

10) DeMorgan:

$$(A \cdot B)' = A' + B'$$

$$(A + B)' = A' \cdot B'$$

Teorema del complemento único

Suponemos 2 complementos para A (A1 y A2)

$$A + A1 = 1 \quad A + A2 = 1$$

$$A \cdot A1 = 0 \quad A \cdot A2 = 0$$

Luego,

$$A1 = A1 \cdot 1 = A1 \cdot (A + A2) = A1 \cdot A + A1 \cdot A2$$

$$A1 = 0 + A2 \cdot A1$$

$$A1 = A \cdot A2 + A1 \cdot A2 = (A + A1) \cdot A2$$

$$A1 = 1 \cdot A2 = A2$$

5. DIAGRAMAS

5.1. DIAGRAMA DE BLOQUES

Circuito 1

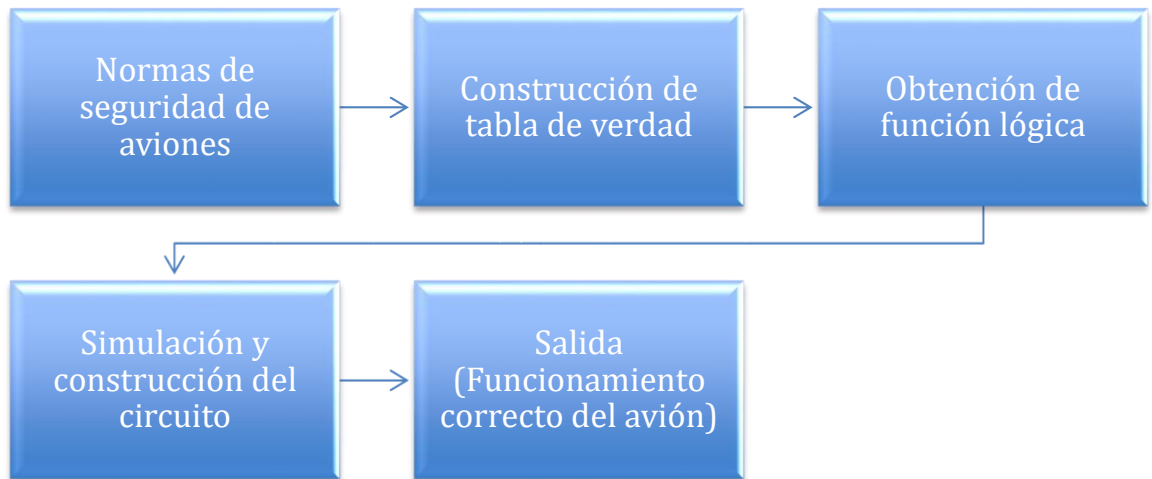


Diagrama 1.1 – Diagrama de bloques Circuito 1.



Diagrama 1.2 – Diagrama de entradas y salidas Circuito 1.

Circuito 2

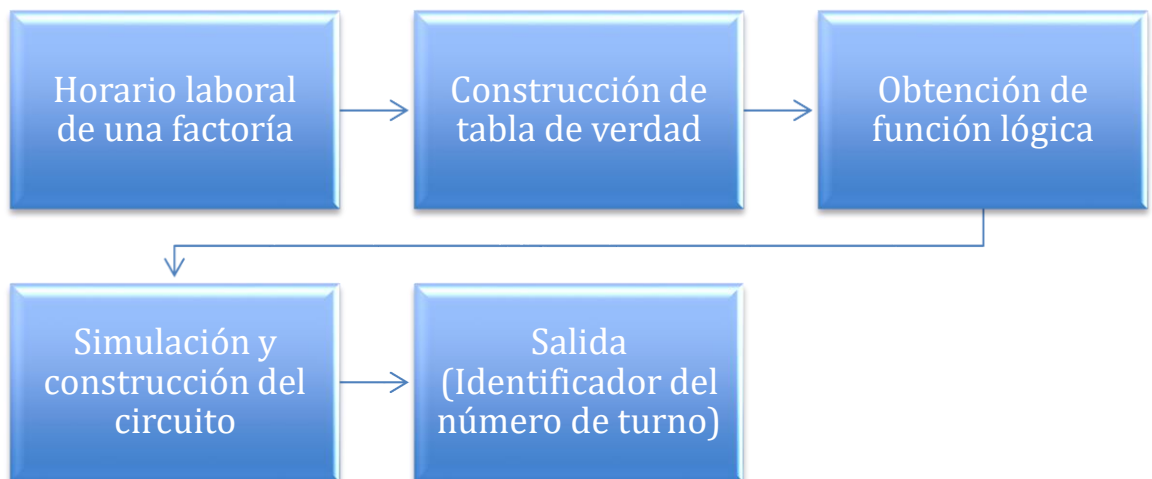


Diagrama 1.3 – Diagrama de bloques Circuito 2.

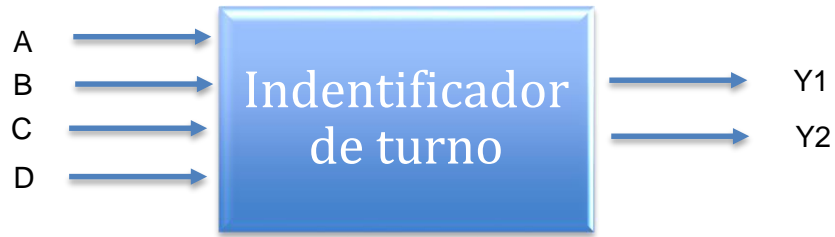


Diagrama 1.4 – Diagrama de entradas y salidas Circuito 2



Diagrama 1.5 – Diagrama de bloques Circuito 3



Diagrama 1.6 – Diagrama de entradas y salidas Circuito 3

6. LISTA DE COMPONENTES

Componentes utilizados	
Tinkercad	Software en línea utilizado para realizar la simulación de un laboratorio virtual, cuya experiencia es muy similar a lo que se realizaría en un laboratorio físico.
Proteus	Software utilizado para realizar la simulación lógica de los circuitos electrónicos digitales.
Compuerta lógica AND	Compuerta lógica, cuyas entradas binarias deben ser verdaderas para que su salida sea verdadera
Compuerta lógica NOT	Compuerta lógica, cuya salida es la negación de su entrada

Compuerta lógica OR	Compuerta lógica, cuyas entradas binarias pueden ser verdaderas o falsa para que su salida sea verdadera
Compuerta lógica NOR	Negación de la compuerta lógica OR,
Diodo LED	Componente utilizado para comprobar y simular el correcto funcionamiento del circuito lógico
Resistencias	Componente utilizado para proteger a los demás elementos que conforman el circuito, oponiéndose al paso de la corriente
Protoboard	Placa de pruebas para realizar la conexión de todos los elementos que conforman un circuito
Dipswitch	Componente utilizado para modificar todas las combinaciones posibles de las entradas lógicas de un circuito
Software online MIT APP INVENTOR 2	Software utilizado para crear la app, cuya programación se la realiza mediante un entorno gráfico (programación por bloques).
Compilador móvil AI Companion	Después de realizar la programación en el software MIT APP INVENTOR 2, es necesario realizar las respectivas pruebas mediante el compilador móvil AI Companion para comprobar que el programa se esté ejecutando de manera correcta
Software Java NetBeans (Apoyo)	Software utilizado como método de apoyo para la elaboración del mismo programa en un entorno diferente (No gráfico).

Tabla 1.1 – Lista de componentes utilizados.

7. MAPA DE VARIABLES

	Circuito “Votador”	Circuito “Número de turno”	Circuito “Comparados de dos números”
Variables de entrada	A → <i>Circuito 1 avión</i> Bit más significativo	A → <i>Representación de horas</i> (Bit más significativo)	a1 → <i>Representación de número A en binario</i> (Bit más significativo)
	B → <i>Circuito 2 avión</i>	B → <i>Representación de horas</i>	a0 → <i>Representación de número A en binario</i>

			(Bit menos significativo)
	<i>C</i> → <i>Circuito 3 avión</i> Bit menos significativo	<i>C</i> → <i>Representación de horas</i>	b1 → <i>Representación de número B en binario (Bit más significativo)</i>
		<i>D</i> → <i>Representación de horas</i> (Bit menos significativo)	b0 → <i>Representación de número B en binario (Bit menos significativo)</i>
Variables de salida	<i>Y</i> Variable de salida única, la cual hace referencia a la representación en valores lógicos (1 y 0) a que el avión está a salvo siempre y cuando dos o más circuitos estén funcionando correctamente	<i>Y1</i> → <i>Número de turno en binario</i> (Bit más significativo)	M → <i>1 Si $A > B$</i>
		<i>Y2</i> → <i>Número de turno en binario</i> (Bit menos significativo)	I → <i>1 Si $A = B$</i>
			m → <i>si $A < B$</i>

Tabla 1.2 – Variables utilizadas para cada circuito.

Circuito 1

Las normas de seguridad de los modernos aviones exigen que, para señales de vital importancia para la seguridad del aparato, los circuitos deben estar triplicados para que el fallo de uno de ellos no produzca una catástrofe. En caso de que los tres circuitos no produzcan la misma salida, ésta se escogerá mediante votación. Diseñe el circuito "votador" que ha de utilizarse para obtener como resultado el valor mayoritario de las tres entradas.

➤ **Tabla de verdad**

Decimal	Entrada 3 bits			Salida
	A	B	C	
0	0	0	0	0
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Tabla 1.3 – Tabla de verdad Circuito 1.

➤ Simplificación de la función lógica (Álgebra de Boole)

Por minterminos:

$$Y = A'BC + AB'C + ABC' + ABC$$

$$Y = A'BC + AB'C + AB$$

$$Y = A'BC + A(B'C + B)$$

$$Y = A'BC + A(B + C)$$

$$Y = A'BC + AB + AC$$

$$Y = B(A'C + A) + AC$$

$$Y = B(A + C) + AC$$

➤ Simulación (Proteus)

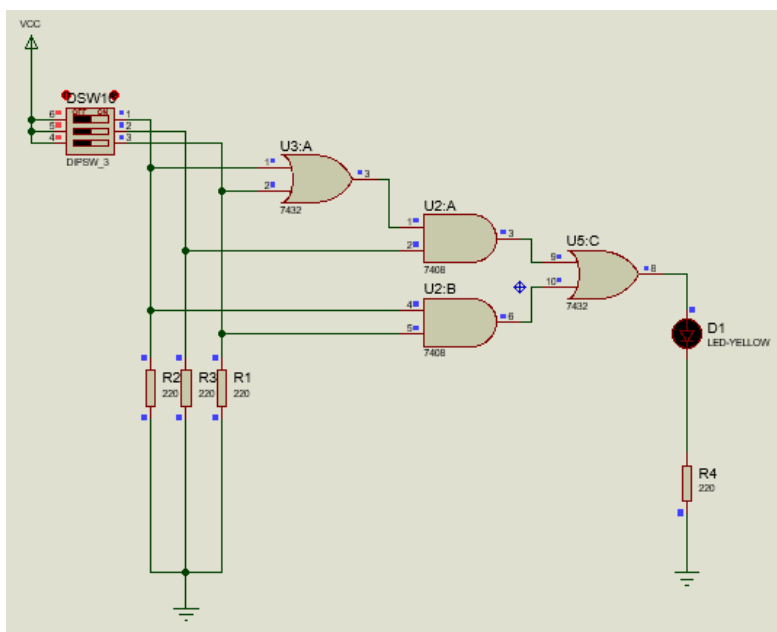


Figura 1.1 – Simulación Circuito 1.

➤ Implementación en Tinkercad (Laboratorio virtual)

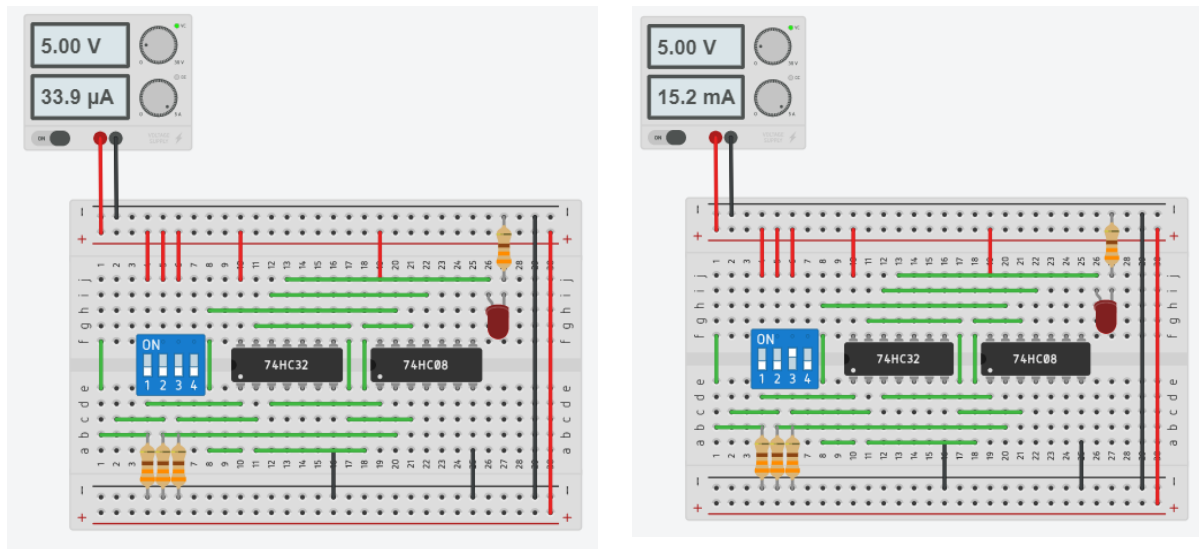


Figura 1.2 – [Inputs (0/0/0) Output 0] [Inputs (0/0/1) Output 0] .

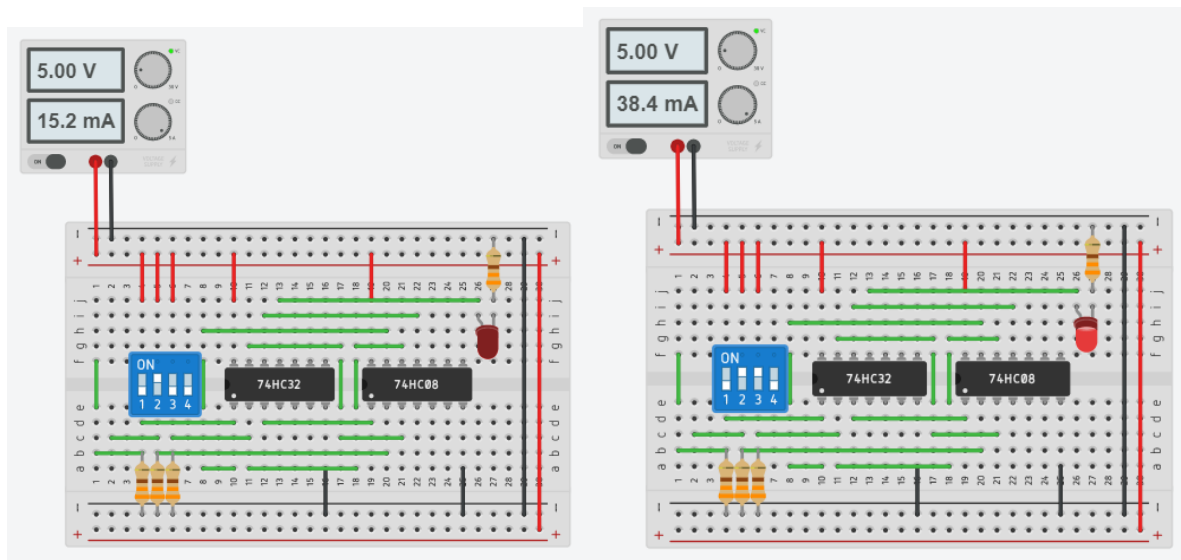


Figura 1.3 – [Inputs (0/1/0) Output 0] [Inputs (0/1/1) Output 1] .

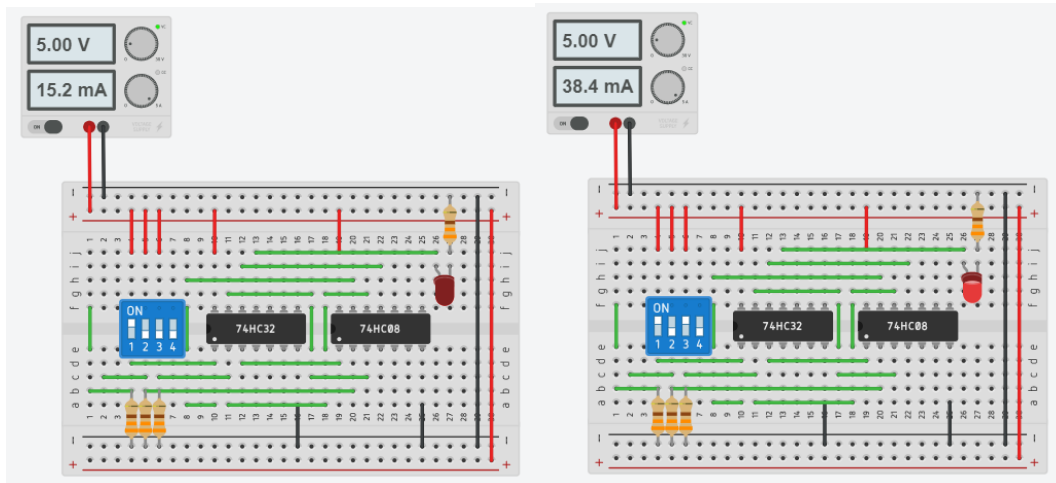


Figura 1.4 – [Inputs (1/0/0) Output 0] [Inputs (1/0/1) Output 1] .

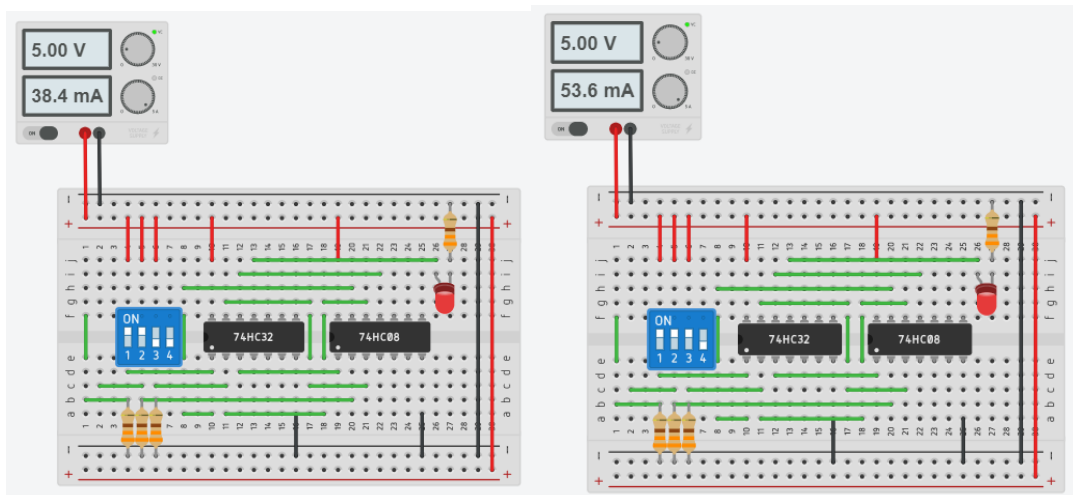


Figura 1.5 – [Inputs (1/1/0) Output 1] [Inputs (1/1/1) Output 1] .

Circuito 2

El horario laboral de una factoría es de 8 horas diarias, divididas en tres turnos: de 8 a 11 (primer turno), de 11 a 13 (segundo turno), de 13 a 16 (descanso) y de 16 a 19 (tercer turno). Se pretende diseñar un circuito que tenga como entradas la representación binaria de la hora actual menos ocho y que proporcione a la salida el número de turno que está trabajando (si procede) o "0" si es hora de descanso.

➤ Tabla de verdad

Decimal	Hora	Entrada 4 bits				Salida	
		A	B	C	D	Y1	Y2
0	8	0	0	0	0	0	1
1	9	0	0	0	1	0	1
2	10	0	0	1	0	0	1
3	11	0	0	1	1	1	0
4	12	0	1	0	0	1	0
5	13	0	1	0	1	0	0
6	14	0	1	1	0	0	0
7	15	0	1	1	1	0	0
8	16	1	0	0	0	1	1
9	17	1	0	0	1	1	1
10	18	1	0	1	0	1	1

Tabla 1.4 – Tabla de verdad Circuito 2.

➤ Simplificación de la función lógica (Álgebra de Boole)

Por maxtérminos:

- $$Y_1 = (A + B + C + D)(A + B + C + D')(A + B' + C + D')(A + B' + C' + D)(A + B' + C' + D')(A + B + C' + D')$$

$$Y_1 = (A + B + C)(A + B' + C')(A + B' + C + D')(A + B + C' + D)$$

$$Y_1 = (A + B + C)(A + B' + C'D')(A + B + C' + D)$$

$$Y_1 = (A + B + CD)(A + B' + C'D')$$

$$Y_1 = A + AB' + AC'D' + AB + BC'D' + ACD + B'CD$$

$$Y_1 = A + BC'D' + B'CD$$

Por mintérminos

- $$Y_2 = A'B'C'D' + A'B'C'D + A'B'CD' + AB'C'D' + AB'C'D + AB'CD'$$

$$Y_2 = A'B'C' + A'B'CD' + AB'C' + AB'CD'$$

$$Y_2 = B'C' + B'CD'$$

$$Y_2 = B'(C' + CD')$$

$$Y_2 = B'(C' + D')$$

➤ Simulación (Proteus)

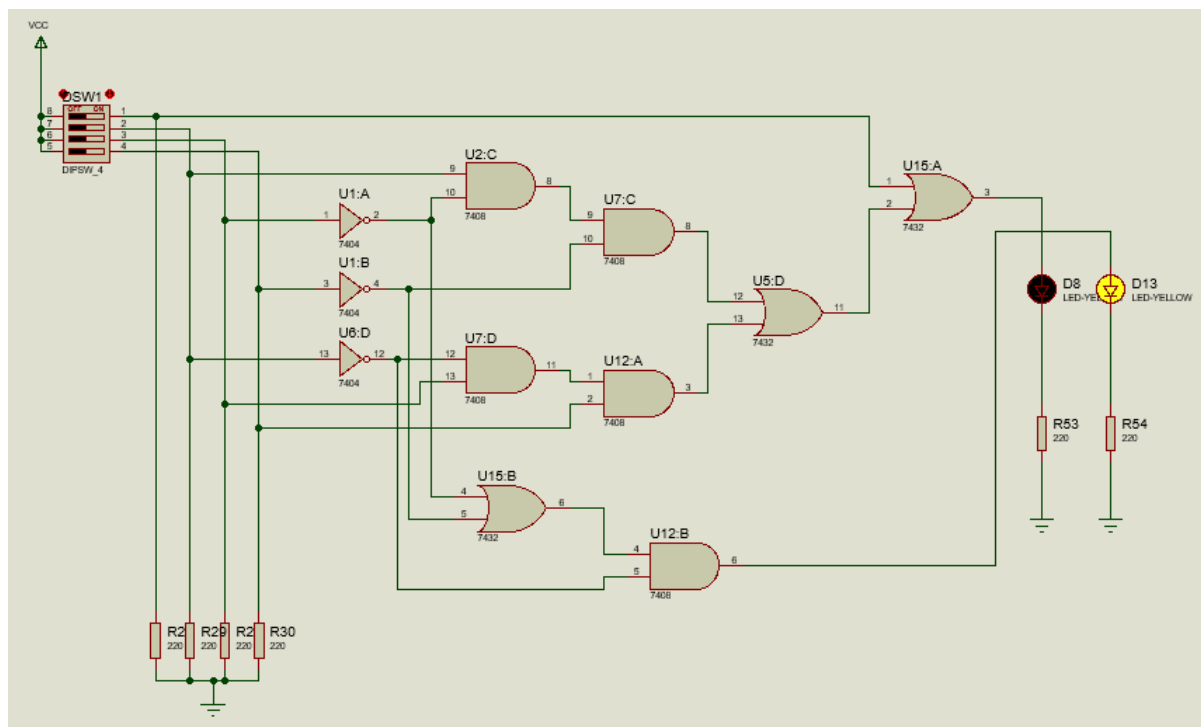


Figura 1.6 – Simulación Circuito 2.

➤ **Implementación en Tinkercad (Laboratorio virtual)**

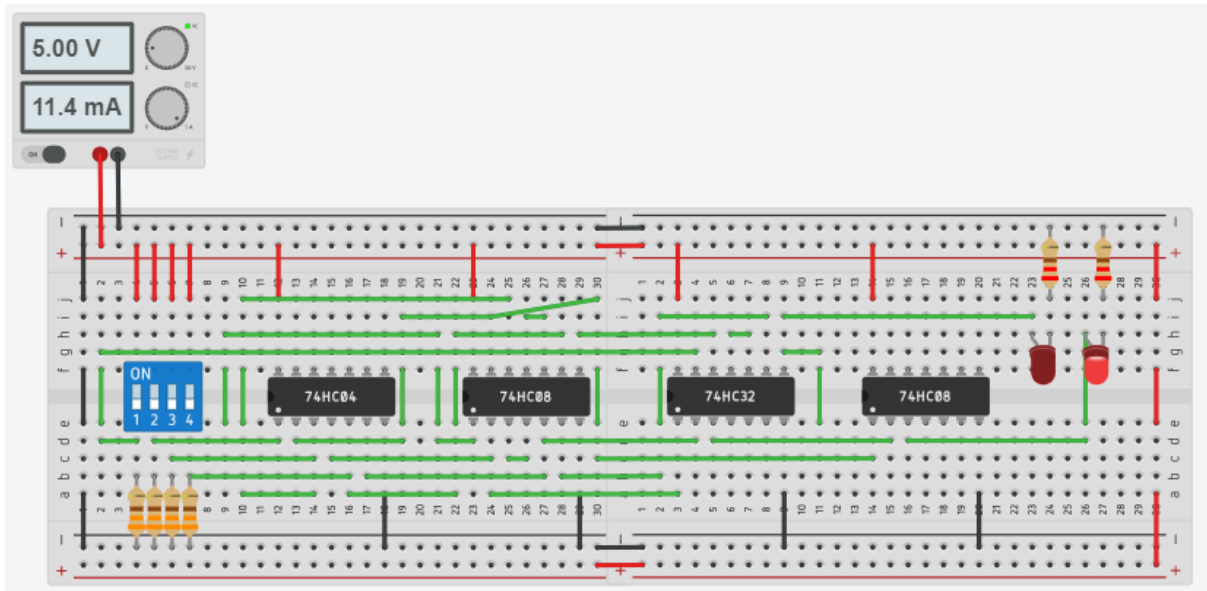


Figura 1.7 – [Inputs (0/0/0/0) Outputs (0/1)].

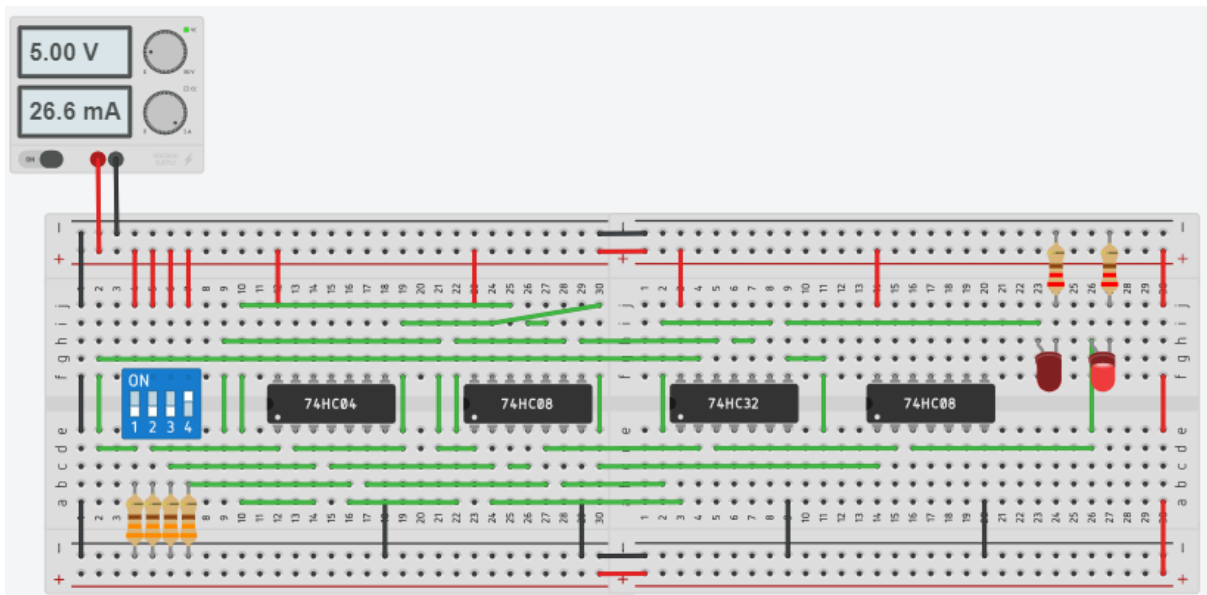


Figura 1.8 – [Inputs (0/0/0/1) Outputs (0/1)].

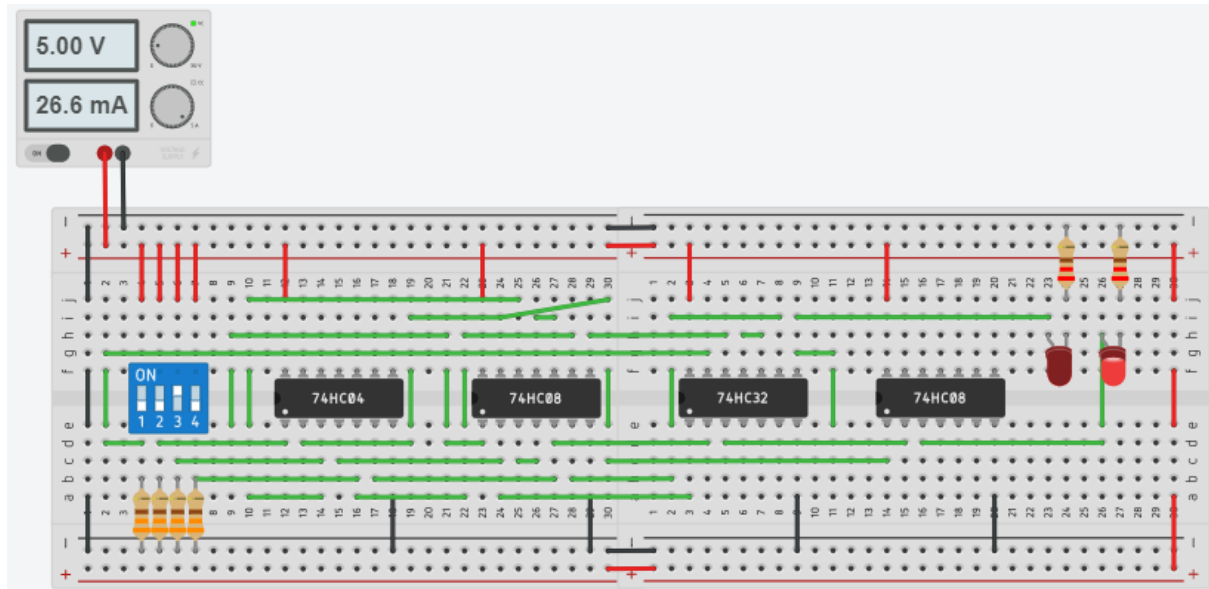


Figura 1.9 – [Inputs (0/0/1/0) Outputs (0/1)].

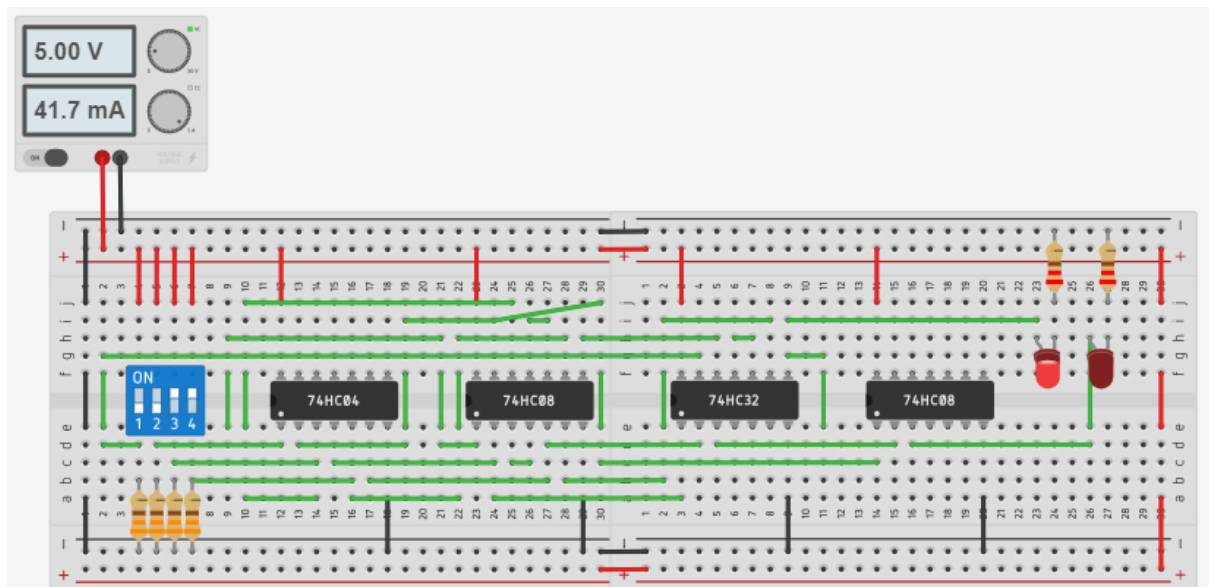


Figura 2.1 – [Inputs (0/0/1/1) Outputs (1/0)].

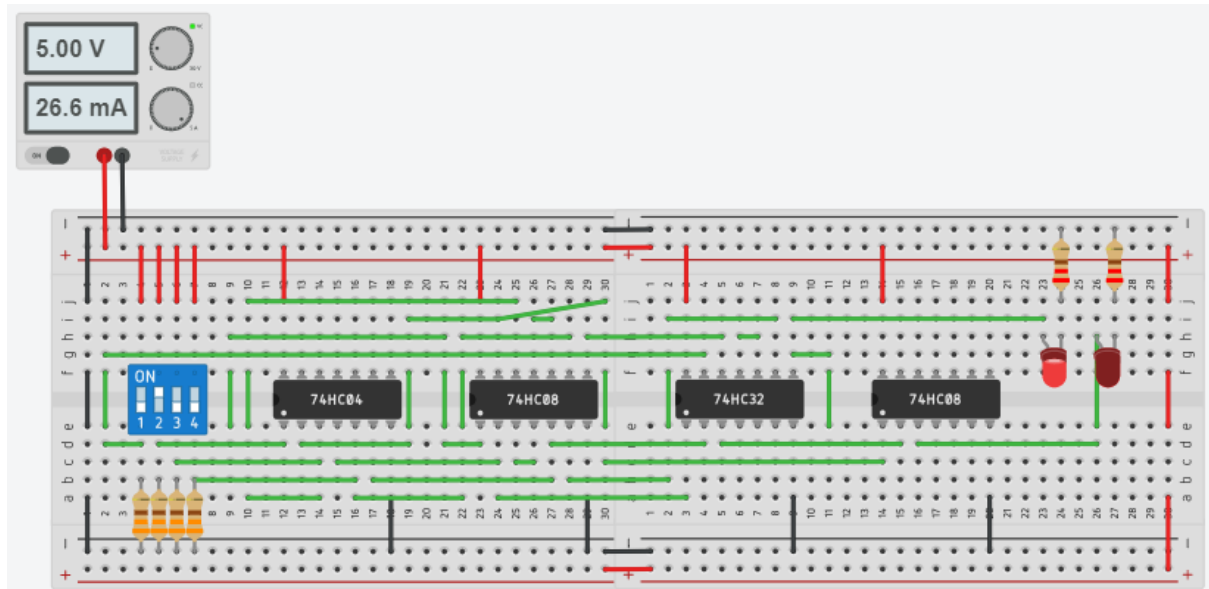


Figura 2.2 – [Inputs (0/1/0/0) Outputs (1/0)].

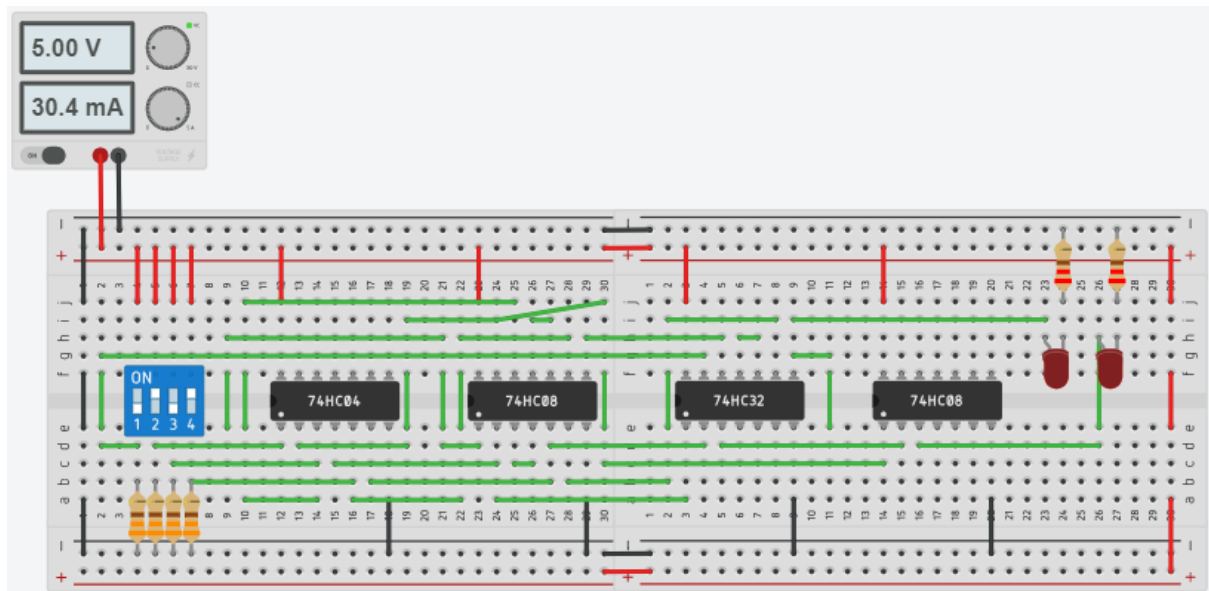


Figura 2.3 – [Inputs (0/1/0/1) Outputs (0/0)].

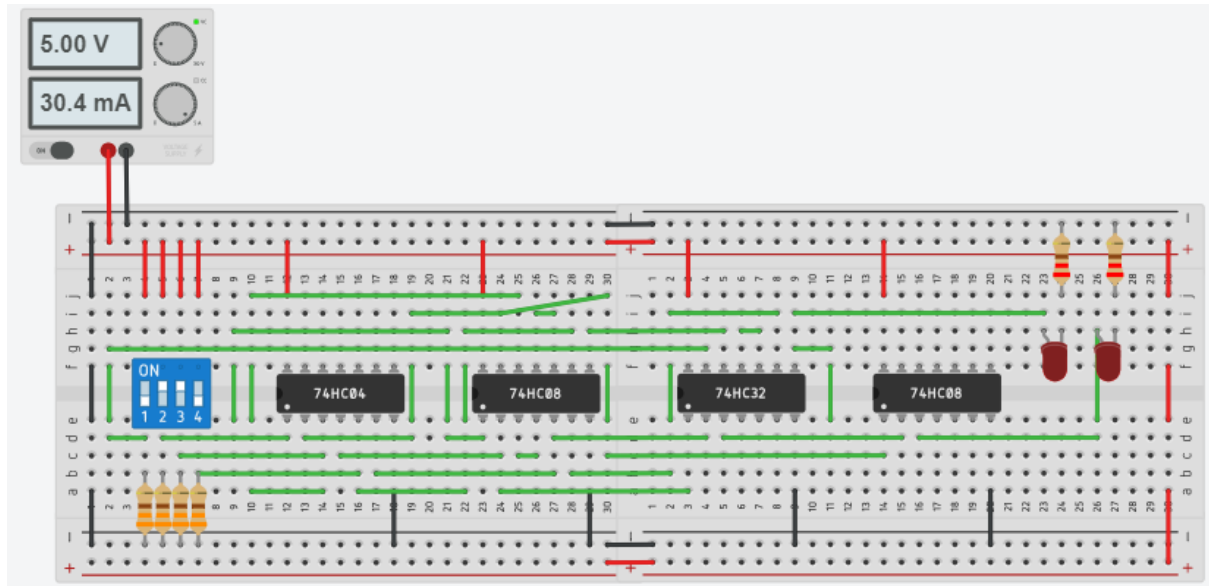


Figura 2.4 – [Inputs (0/1/1/0) Outputs (0/0)].

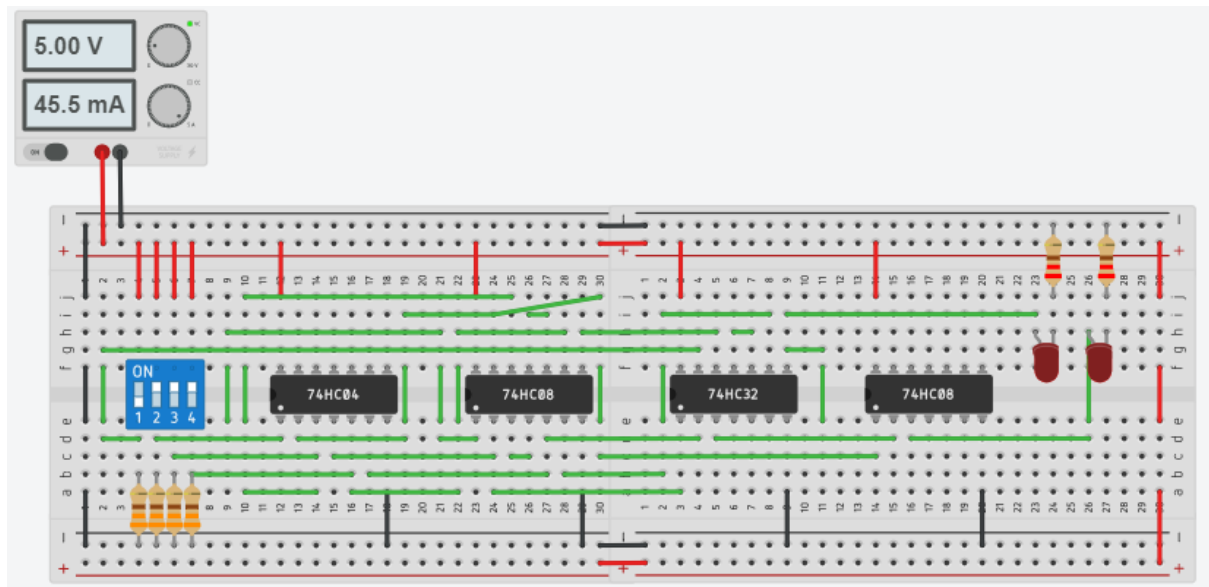


Figura 2.5 – [Inputs (0/1/1/1) Outputs (0/0)].

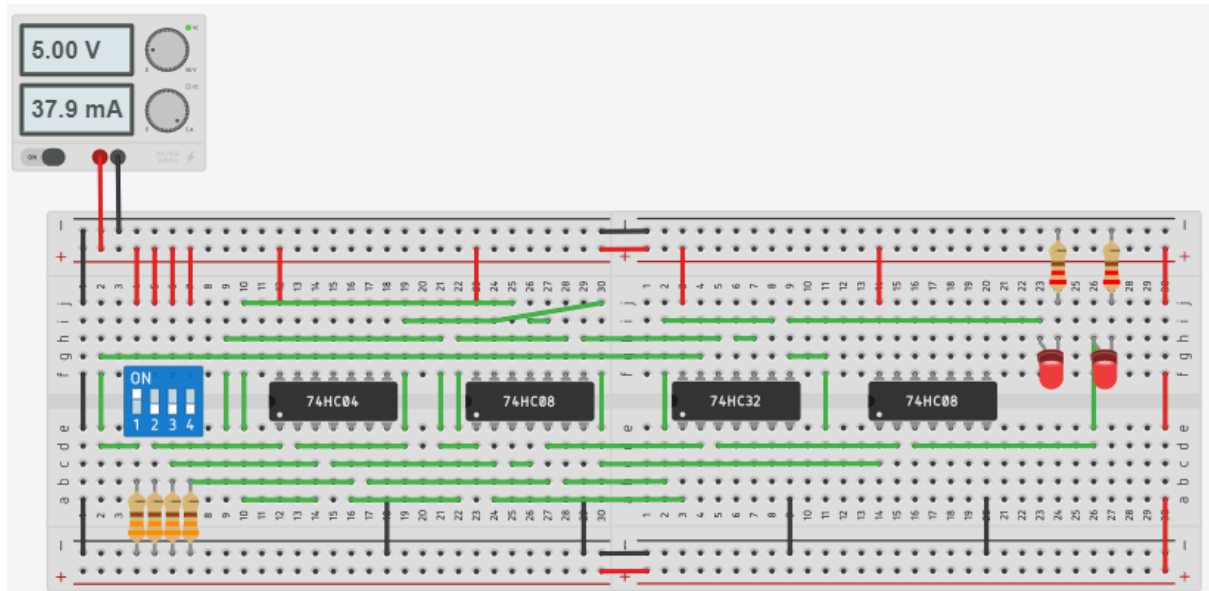


Figura 2.6 – [Inputs (1/0/0/0) Outputs (1/1)].

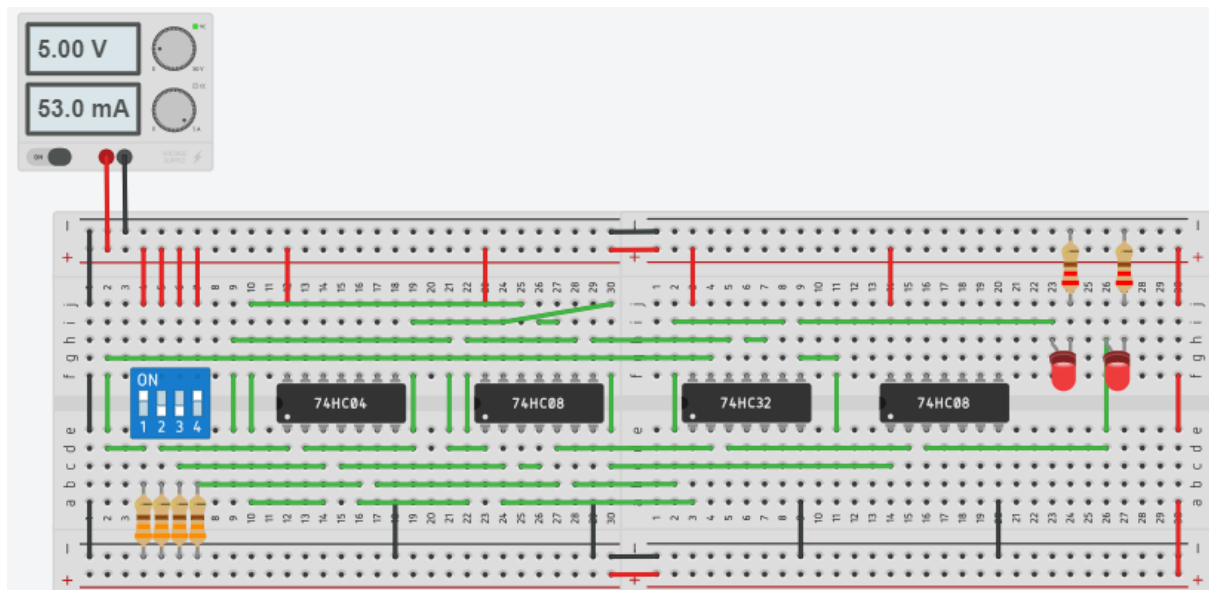


Figura 2.7 – [Inputs (1/0/0/1) Outputs (1/1)].

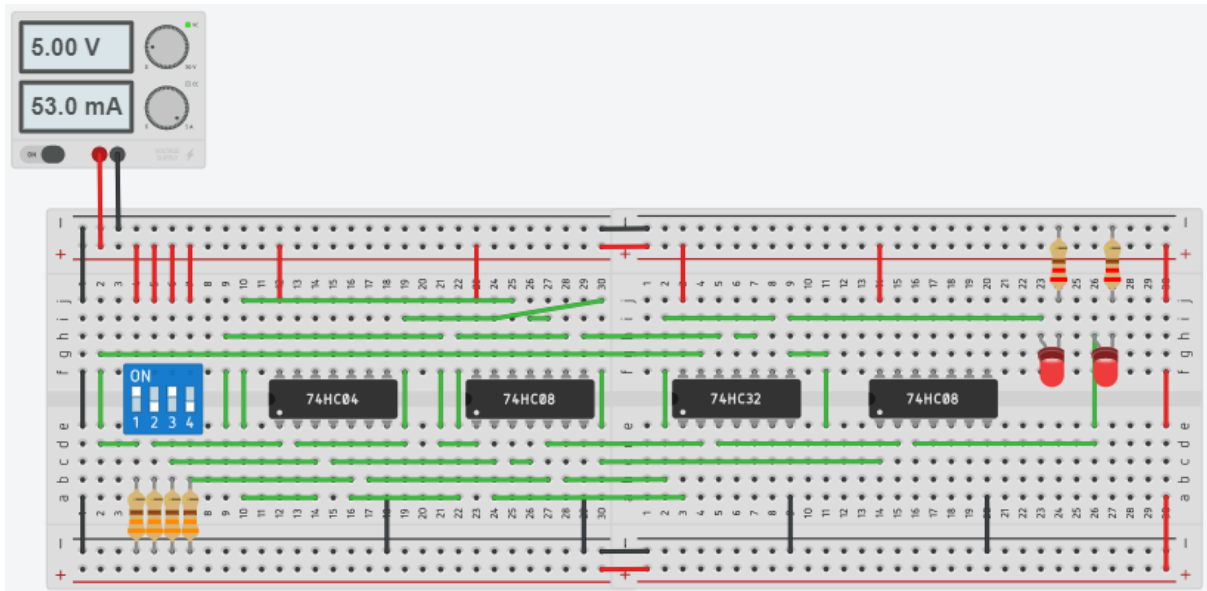


Figura 2.8 – [Inputs (1/0/1/0) Outputs (1/1)].

➤ Implementación del circuito con display de 7 segmentos

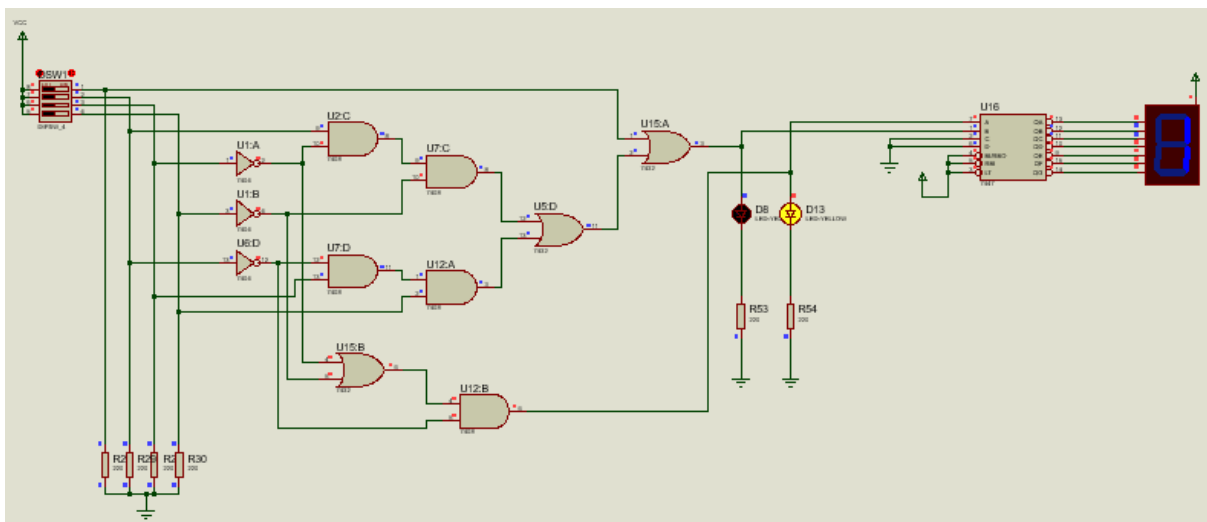


Figura 2.9 – Turno número 1, [Inputs (0/0/0/0) Outputs (0/1)].

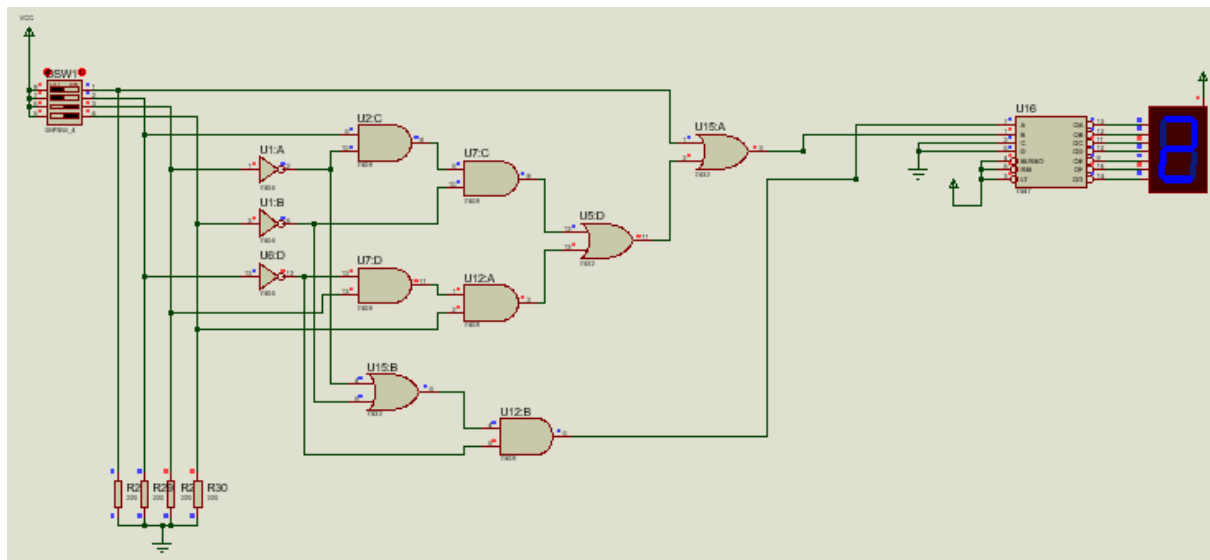


Figura 2.9 – Turno número 2, [Inputs (0/0/1/1) Outputs (1/0)].

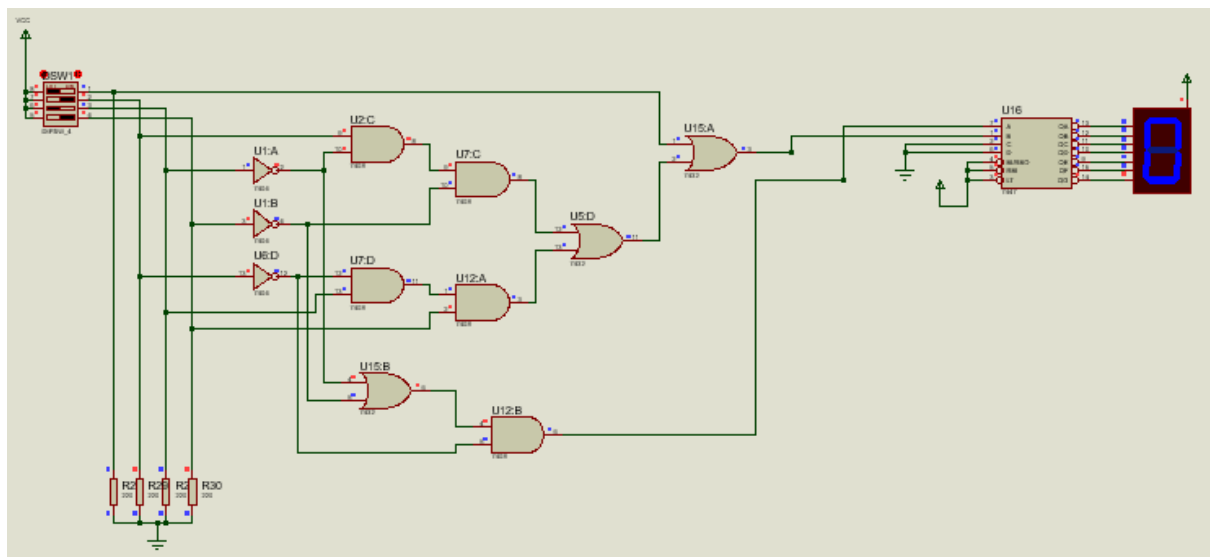


Figura 3.1 – Turno número 0 (Descanso), [Inputs (0/1/0/1) Outputs (0/0)].

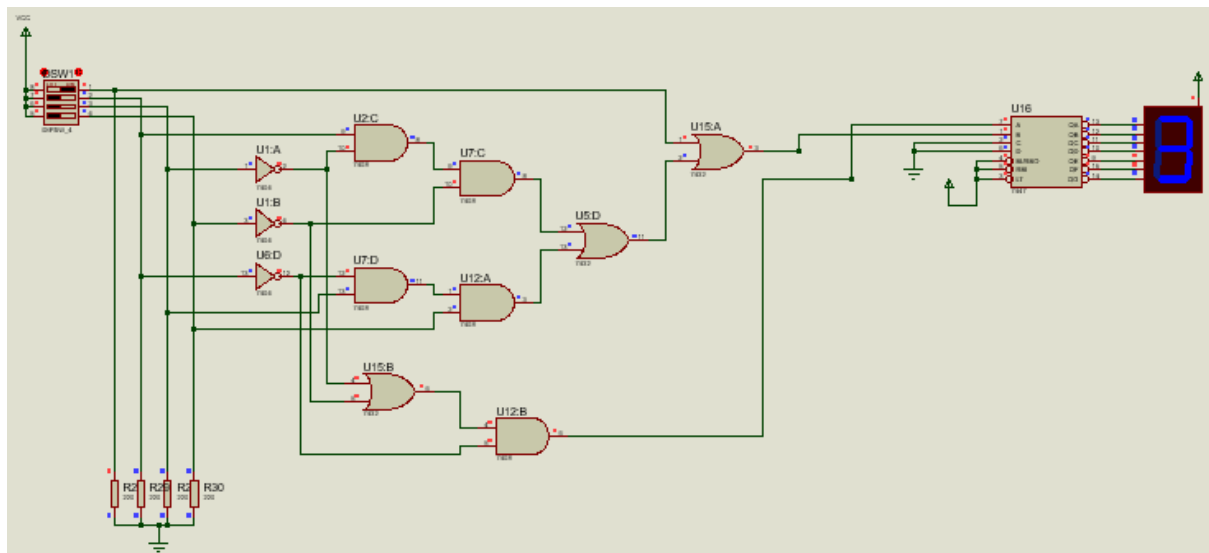


Figura 3.2 – Turno número 3, [Inputs (1/0/0/0) Outputs (1/1)].

Circuito 3

Se pretende diseñar un circuito comparador de 2 números de 2 bits, $A = (a_1, a_0)$ y $B = (b_1, b_0)$.

Dicho circuito deberá tener tres salidas M , I , m , de tal forma que:

$M = 1$ si $A > B$

$I = 1$ si $A = B$

$m = 1$ si $A < B$

➤ **Tabla de verdad**

Decimal	Entrada 4 bits				M	Salida	
	a1	a0	b1	b0		1	m
0	0	0	0	0	0	1	0
1	0	0	0	1	0	0	1
2	0	0	1	0	0	0	1
3	0	0	1	1	0	0	1
4	0	1	0	0	1	0	0
5	0	1	0	1	0	1	0
6	0	1	1	0	0	0	1
7	0	1	1	1	0	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	0
10	1	0	1	0	0	1	0
11	1	0	1	1	0	0	1
12	1	1	0	0	1	0	0
13	1	1	0	1	1	0	0
14	1	1	1	0	1	0	0
15	1	1	1	1	0	1	0

Tabla 1.5 – Tabla de verdad Circuito 3.

➤ **Simplificación de la función lógica (Mapas de Karnaugh)**

Salida M (1 si A>B)

$a_1 a_0$ $b_1 b_0$	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	0	0	0	0
10	0	0	1	0

Tabla 1.6 – Tabla de simplificación para M, mediante mapas de Karnaugh

Entonces, la función simplificada nos queda de la siguiente manera:

$$M = a_0 \bar{b}_1 \bar{b}_0 + a_1 a_0 \bar{b}_0 + a_1 \bar{b}_1$$

Salida l (1 si A=B)

$a_1 a_0$ $b_1 b_0$	00	01	11	10
00	1	0	0	0
01	0	1	0	0
11	0	0	1	0
10	0	0	0	1

Tabla 1.7 – Tabla de simplificación para l, mediante mapas de Karnaugh

En este caso no podemos simplificar mucho más, ya que no se pueden agrupar los términos “1”. Entonces la función l nos quedaría:

$$l = \overline{a_1} \overline{a_0} \overline{b_1} \overline{b_0} + \overline{a_1} a_0 \overline{b_1} b_0 + a_1 a_0 b_1 b_0 + a_1 \overline{a_0} b_1 \overline{b_0}$$

Salida m (1 si A<B)

$\begin{matrix} a_1 a_0 \\ b_1 b_0 \end{matrix}$	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	0	1
10	1	1	0	0

Tabla 1.8 – Tabla de simplificación para m, mediante mapas de Karnaugh

$$m = \overline{a_1} b_1 + \overline{a_1} \overline{a_0} b_0 + \overline{a_0} b_1 b_0$$

➤ Simulación (Proteus)

Salida M

Para realizar el diseño del circuito exclusivamente con compuertas universales NOR, primero se ha procedido a realizarlo con las compuertas básicas AND, NOT y OR, y posterior a ello se llevó a cabo la transformación del circuito mencionado, por medio del álgebra de Boole y específicamente aplicando las leyes de Morgan para llevar de una compuerta AND a NOR. Para ilustrar de mejor manera esta transformación podemos observar la tabla 1.6.

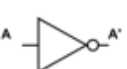




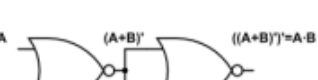

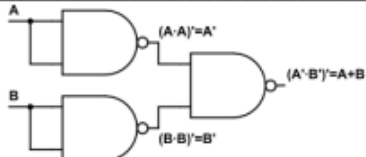
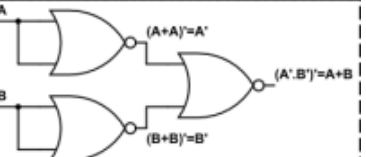
	Con NAND	Con NOR
	 $(A \cdot A)' = A'$	 $(A + A)' = A'$
	 $((A \cdot B)')' = A \cdot B$	 $((A + B)')' = A \cdot B$
	 $(A' \cdot B')' = A + B$	 $(A' \cdot B')' = A + B$

Tabla 1.6 – Transformación a compuertas NOR.

Aplicando todo este concepto a nuestro circuito. Tenemos primeramente el circuito con las compuertas básicas AND, NOT y OR (Figura 3.3) y posterior a ello el mismo circuito con únicamente compuertas NOR (Figura 3.4).

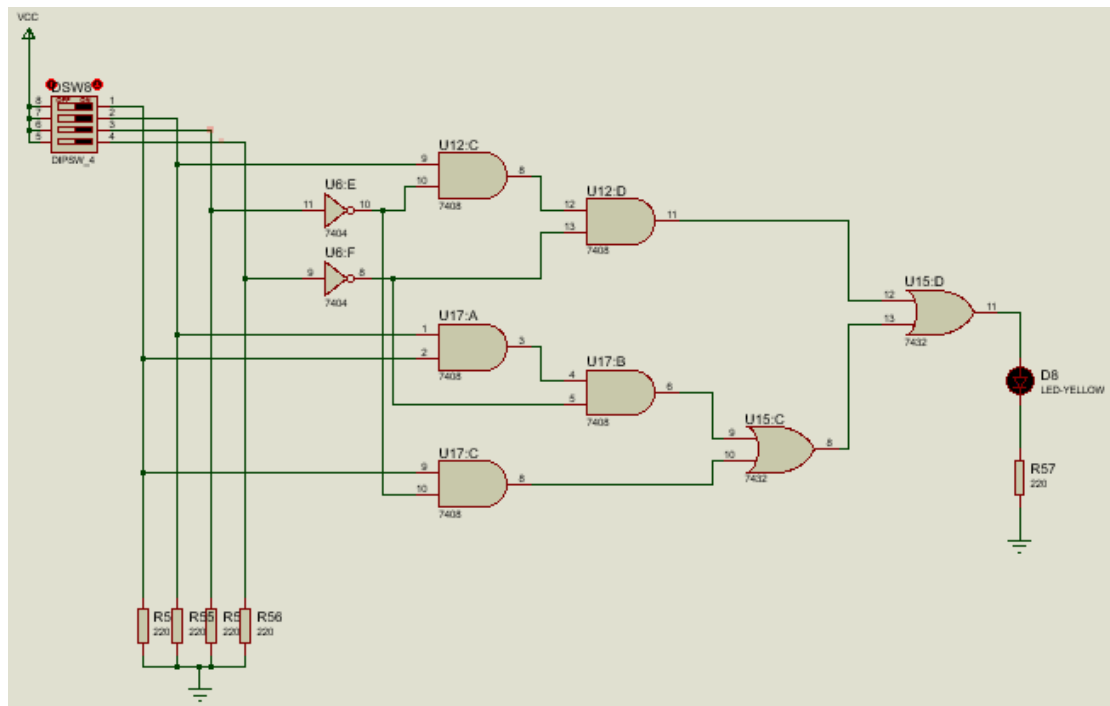


Figura 3.3 – Circuito con compuertas básicas.

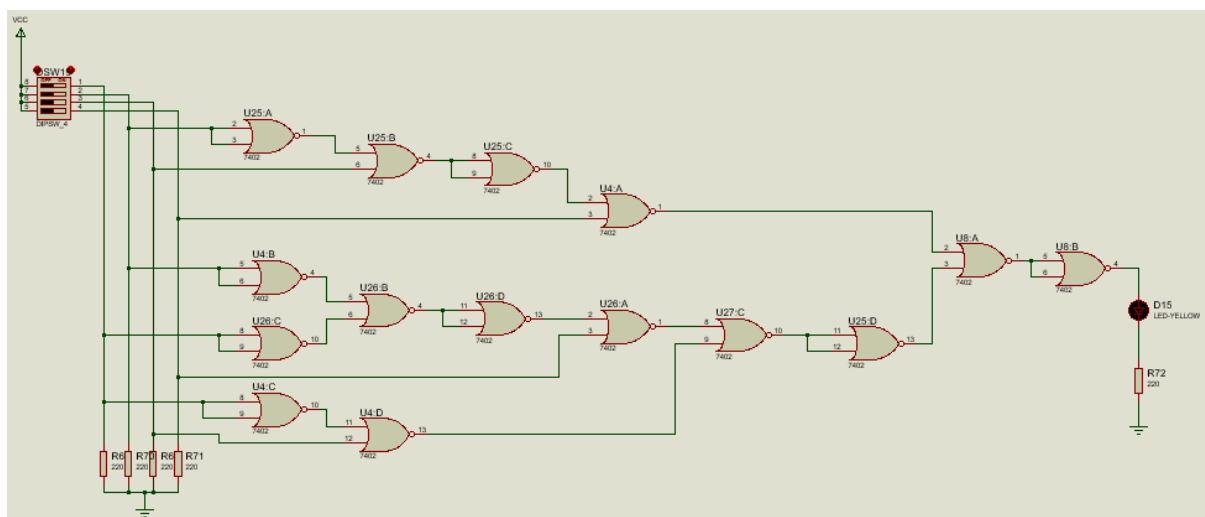


Figura 3.4 – Circuito con compuertas universales NOR.

Salida I

De la misma manera realizamos esta transformación con las demás salidas. Entonces nos queda de la siguiente manera:

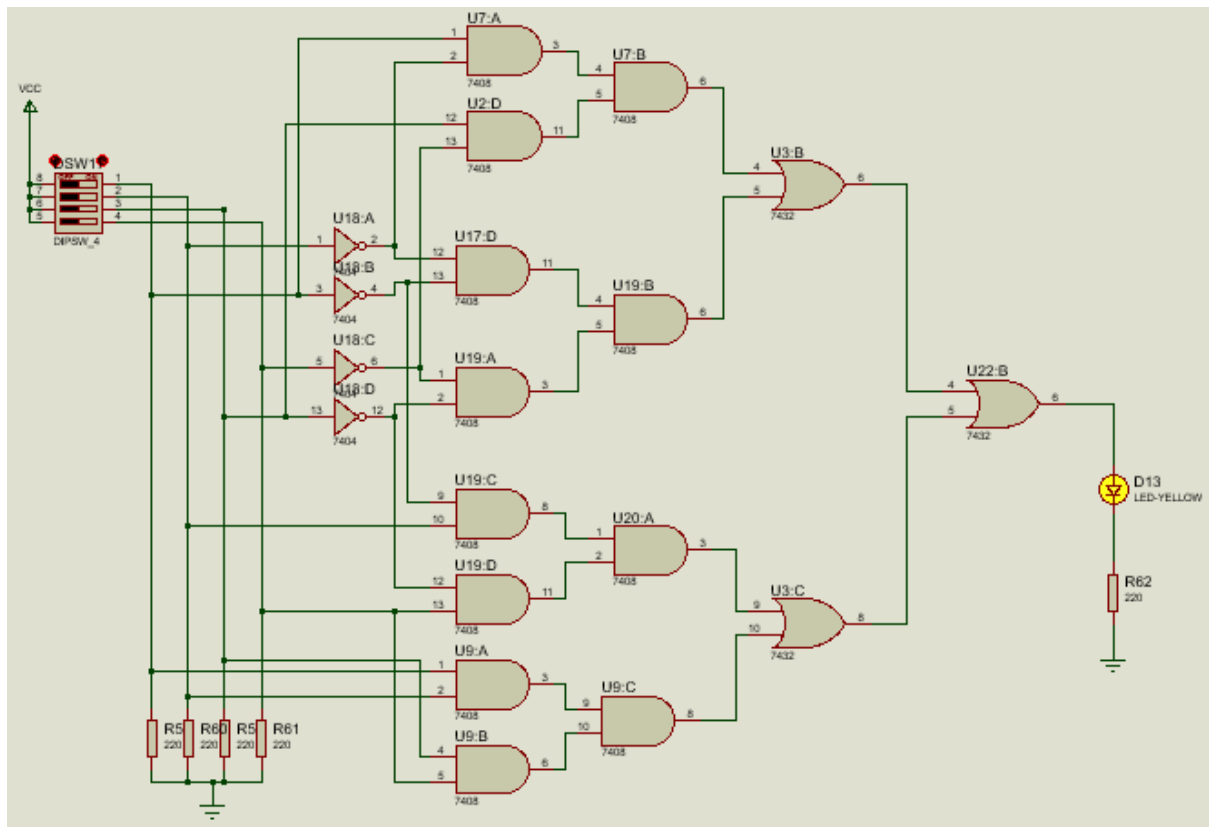


Figura 3.5 – Circuito con compuertas básicas.

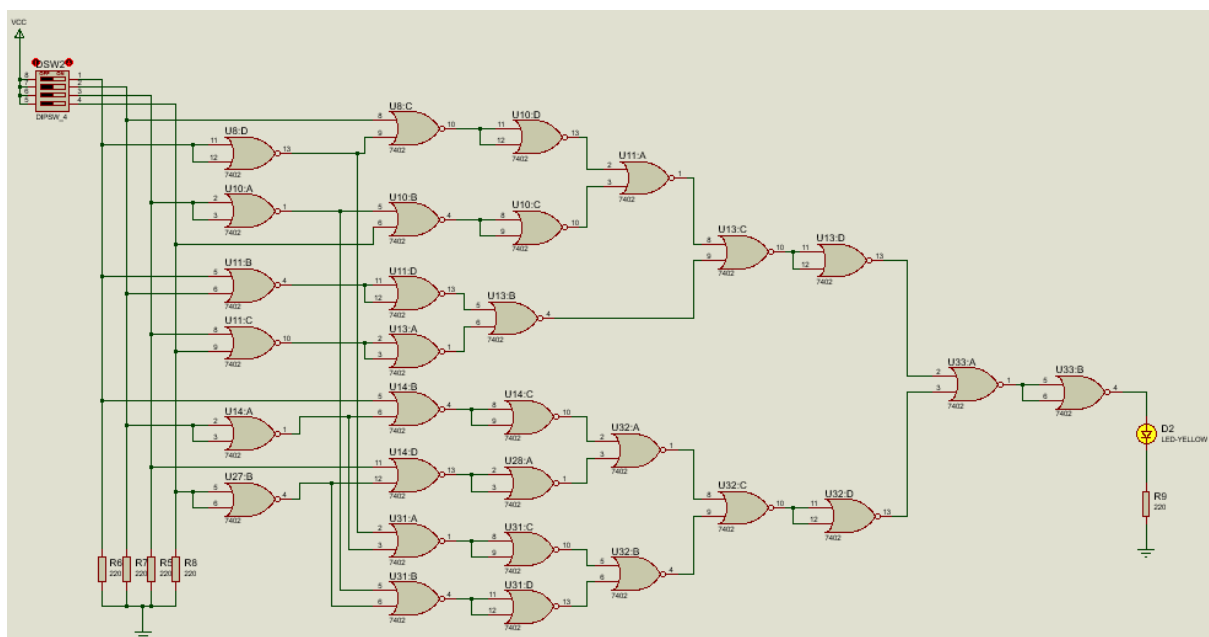


Figura 3.6 – Circuito con compuertas universales NOR.

Salida m

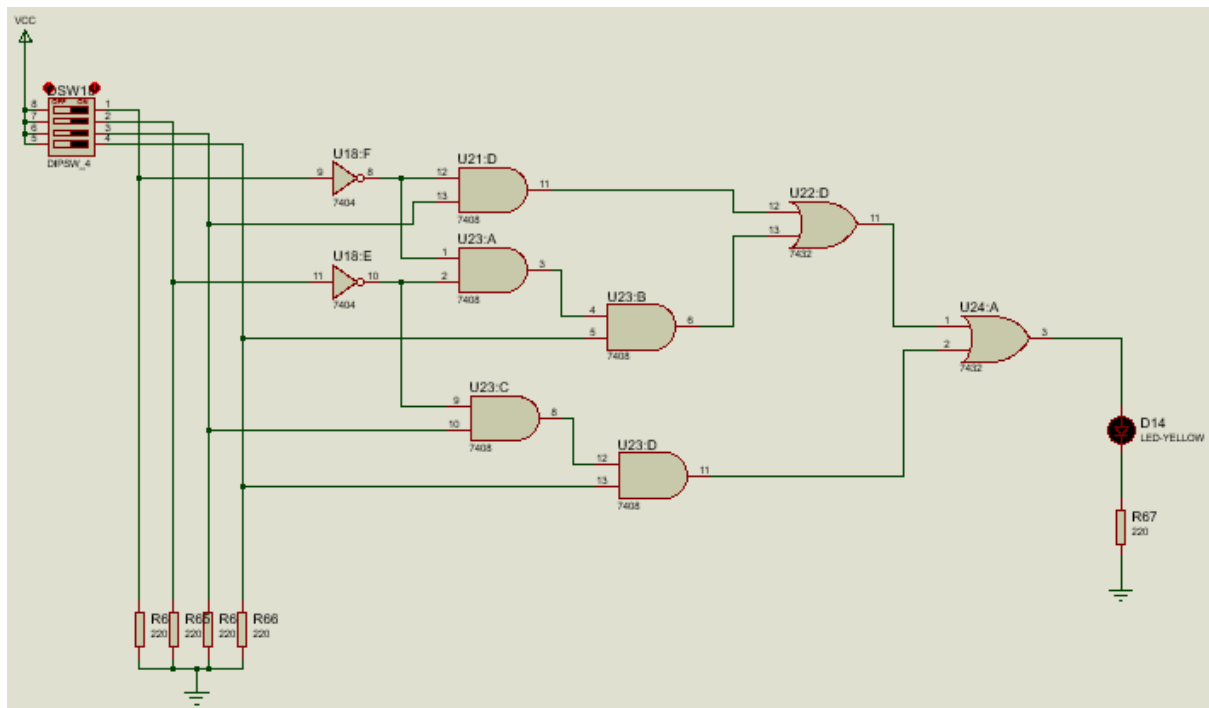


Figura 3.7 – Circuito con compuertas básicas.

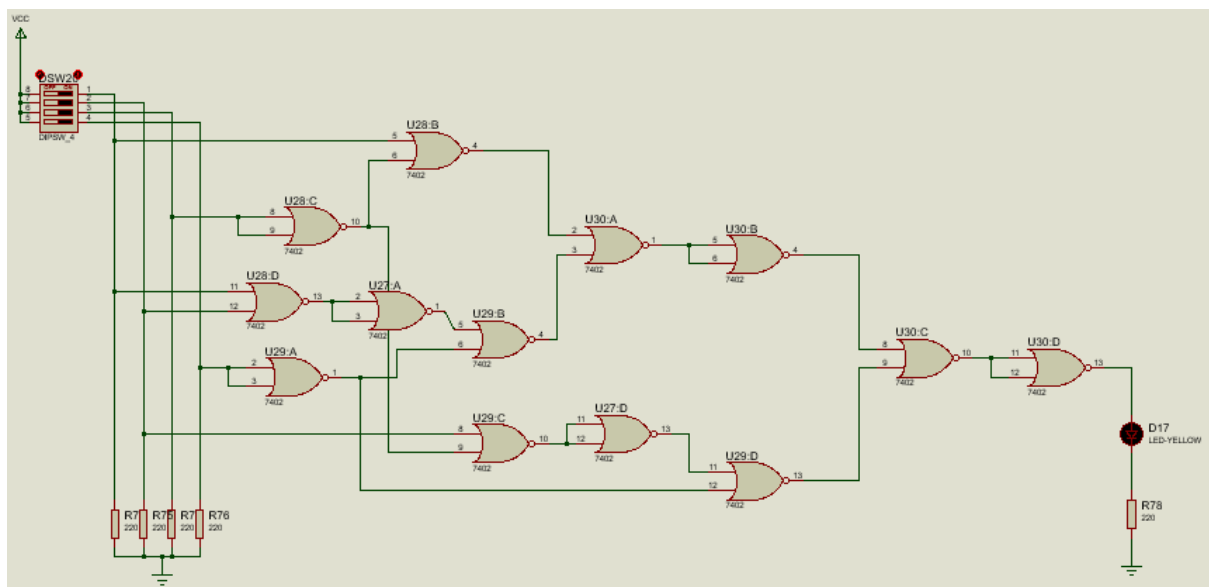


Figura 3.8 – Circuito con compuertas universales NOR.

➤ Implementación en Tinkercad (Laboratorio virtual)

Salida M

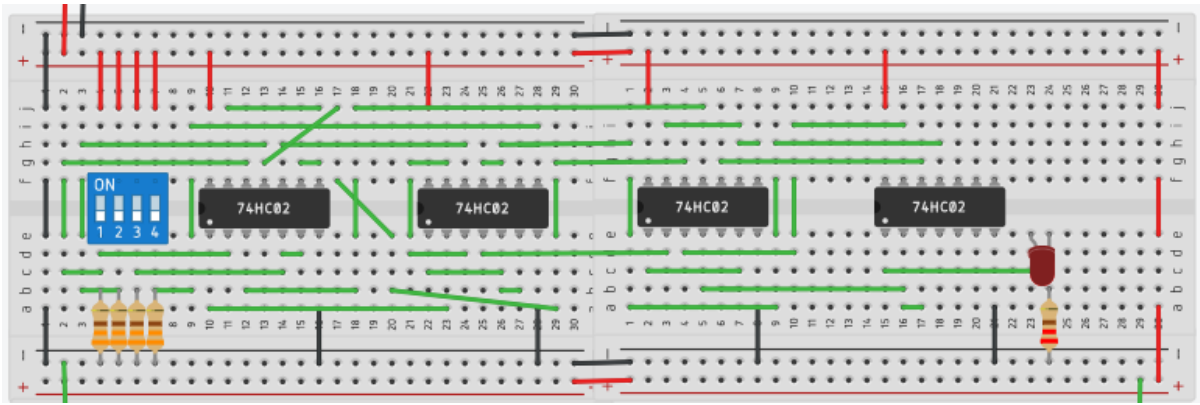


Figura 3.9 – Implementación en Lab. Virtual [Inputs (0/0/0/0) Output M (0)].

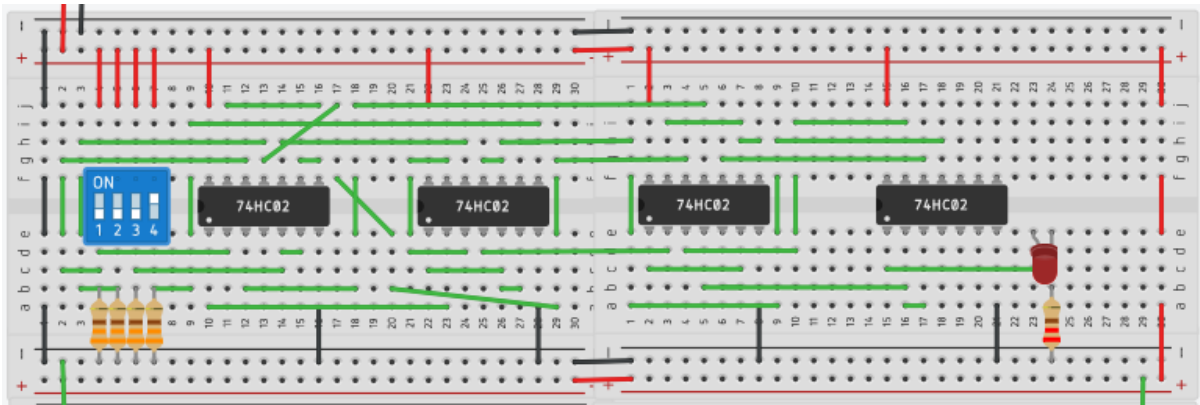


Figura 4.1 – Implementación en Lab. Virtual [Inputs (0/0/0/1) Output M (0)].

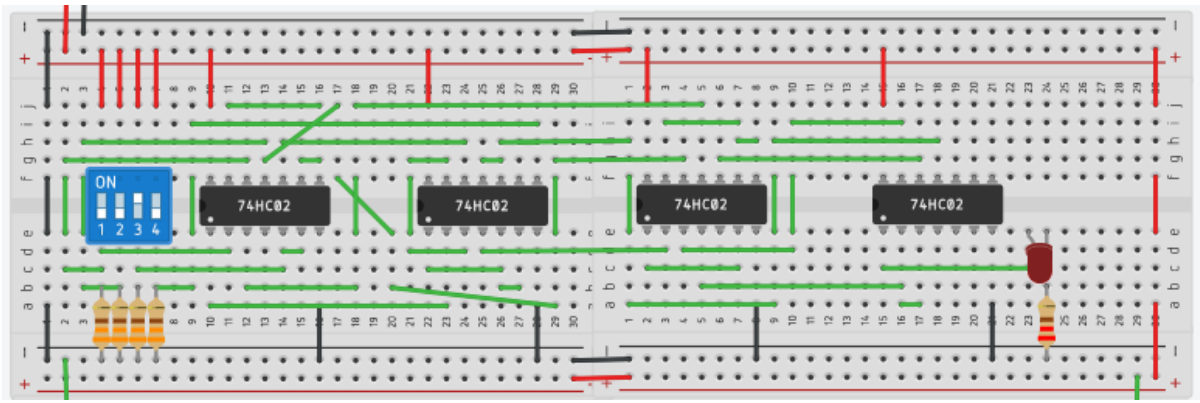


Figura 4.2 – Implementación en Lab. Virtual [Inputs (0/0/1/0) Output M (0)].

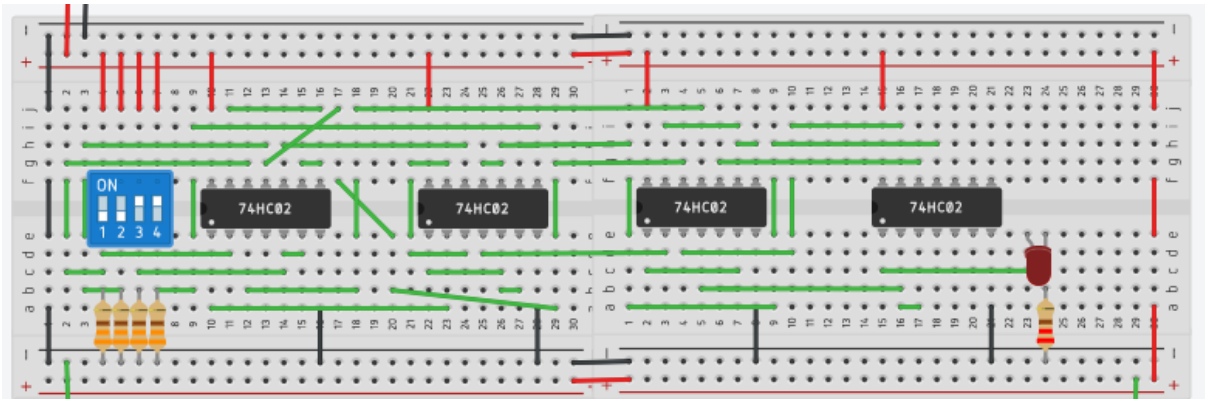


Figura 4.3 – Implementación en Lab. Virtual [Inputs (0/0/1/1) Output M (0)].

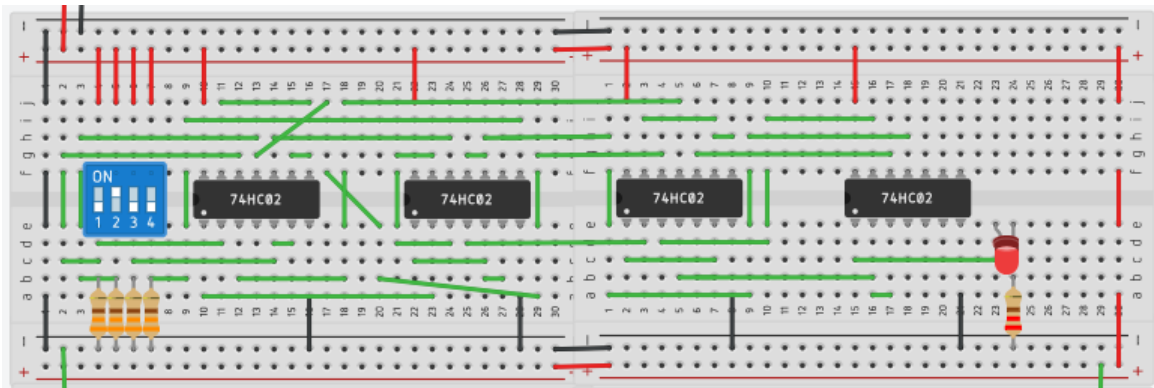


Figura 4.4 – Implementación en Lab. Virtual [Inputs (0/1/0/0) Output M (1)].

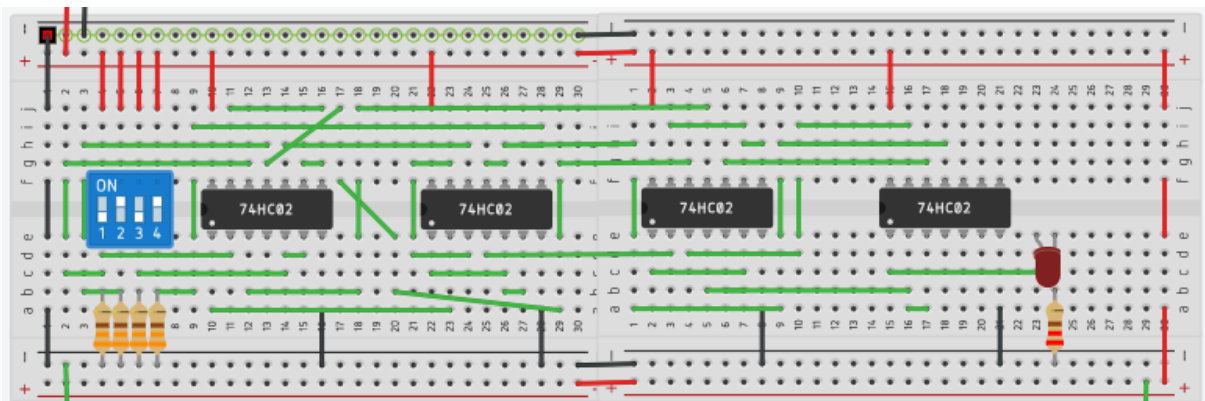


Figura 4.5 – Implementación en Lab. Virtual [Inputs (0/1/0/1) Output M (0)].

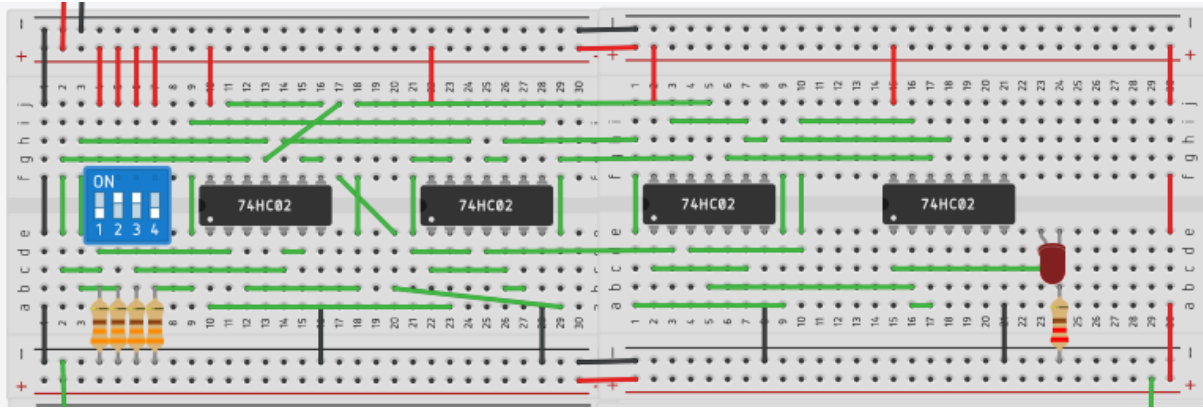


Figura 4.6 – Implementación en Lab. Virtual [Inputs (0/1/1/0) Output M (0)].

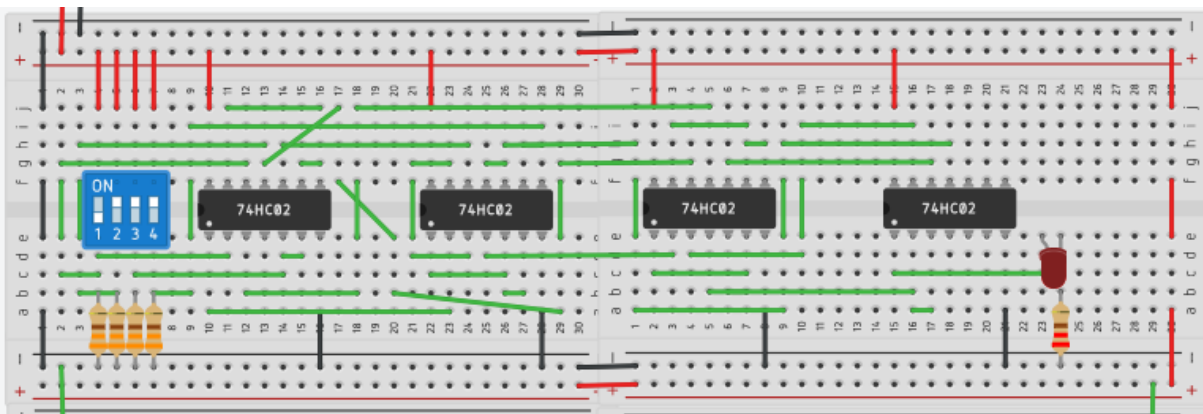


Figura 4.7 – Implementación en Lab. Virtual [Inputs (0/1/1/1) Output M (0)].

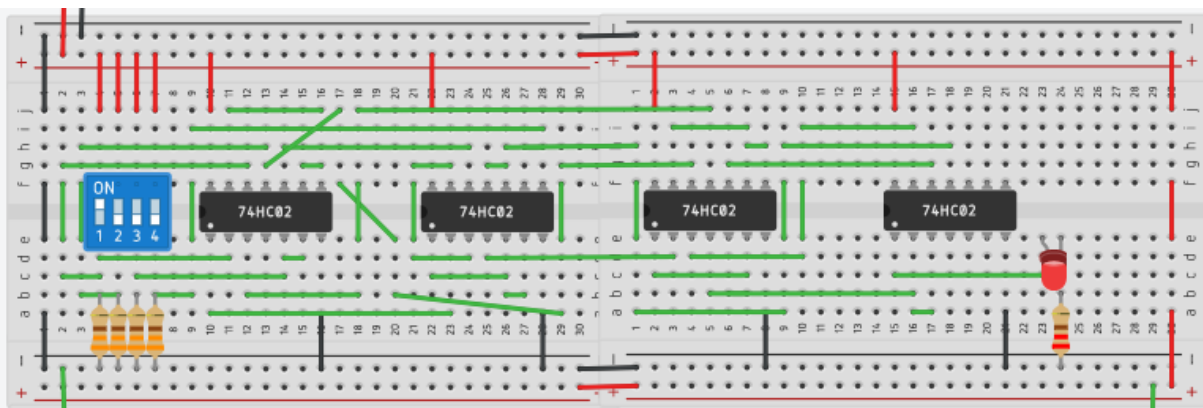


Figura 4.8 – Implementación en Lab. Virtual [Inputs (1/0/0/0) Output M (1)].

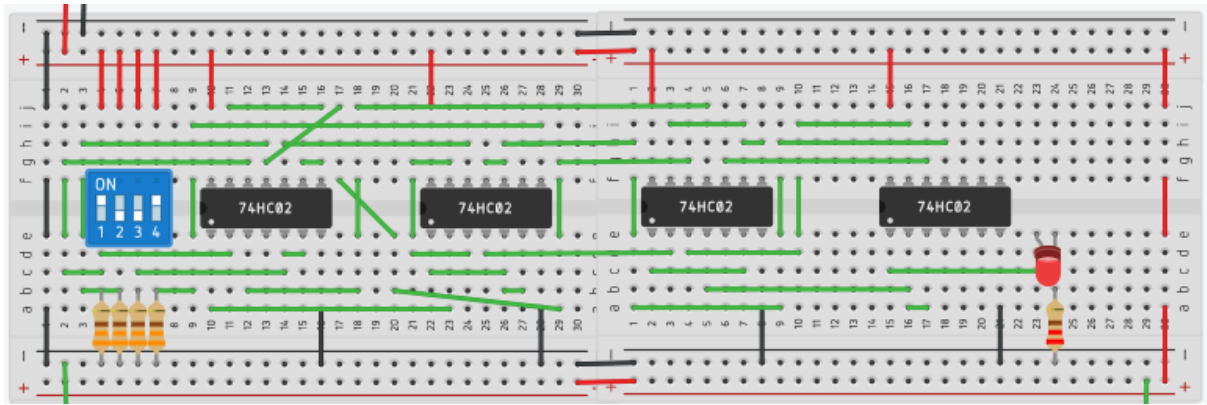


Figura 4.9 – Implementación en Lab. Virtual [Inputs (1/0/0/1) Output M (1)].

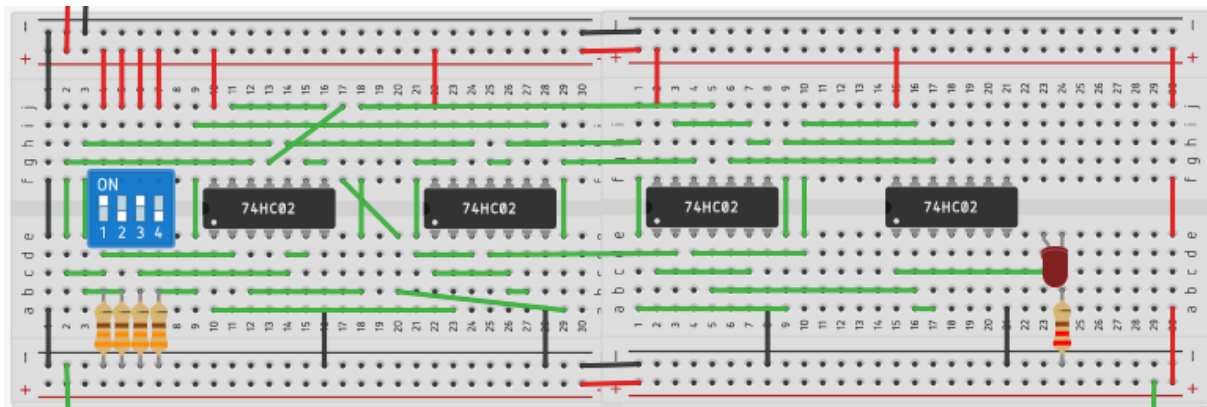


Figura 5.1 – Implementación en Lab. Virtual [Inputs (1/0/1/0) Output M (0)].

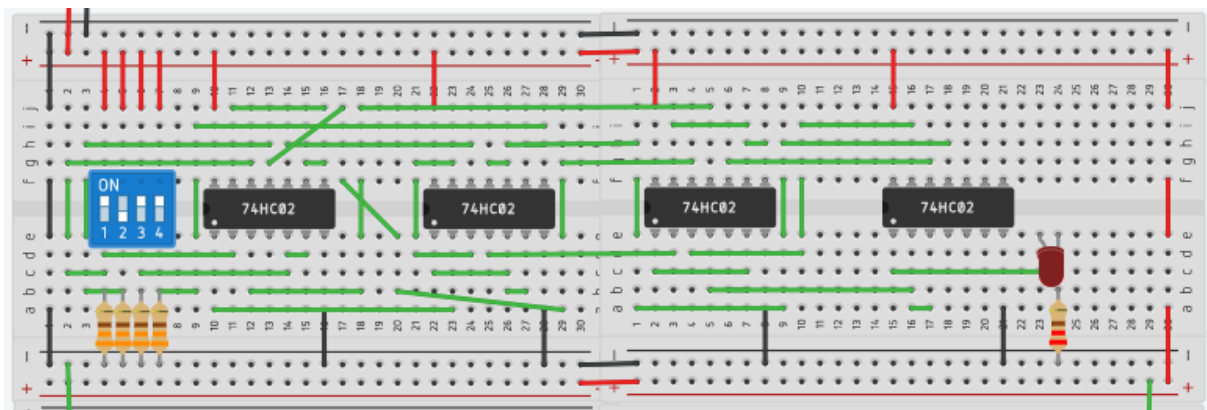


Figura 5.2 – Implementación en Lab. Virtual [Inputs (1/0/1/1) Output M (0)].

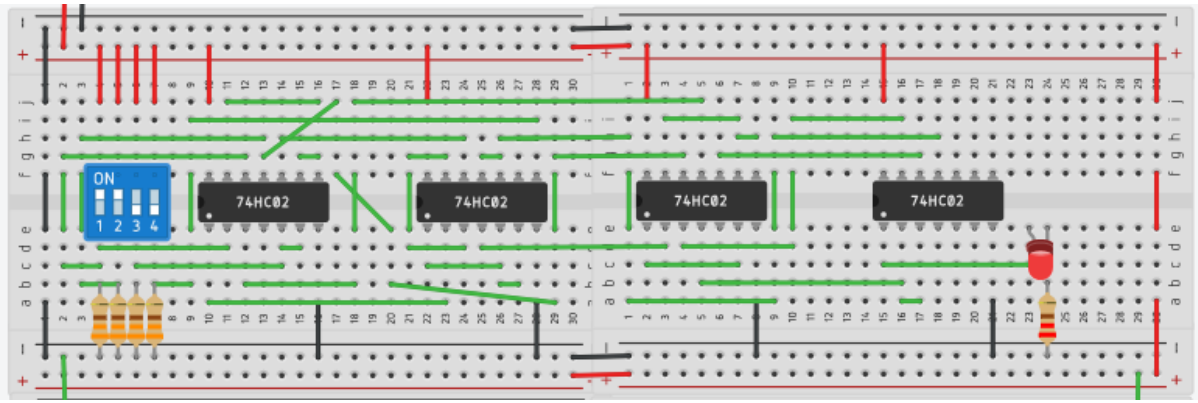


Figura 5.3 – Implementación en Lab. Virtual [Inputs (1/1/0/0) Output M (1)].

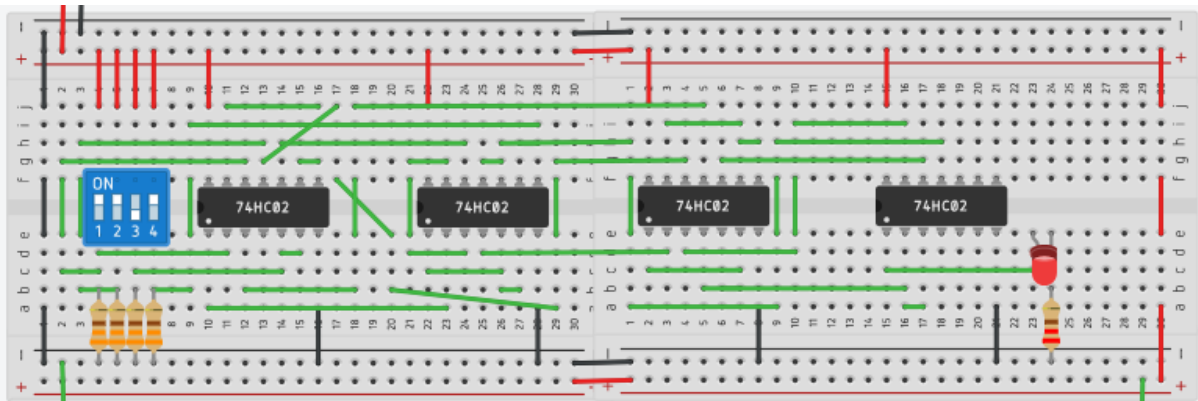


Figura 5.4 – Implementación en Lab. Virtual [Inputs (1/1/0/1) Output M (1)].

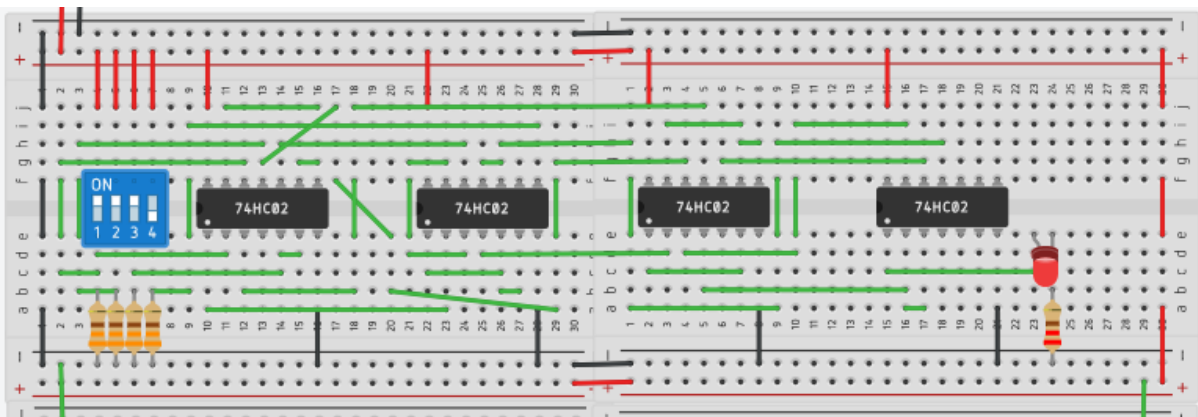


Figura 5.5 – Implementación en Lab. Virtual [Inputs (1/1/1/0) Output M (1)].

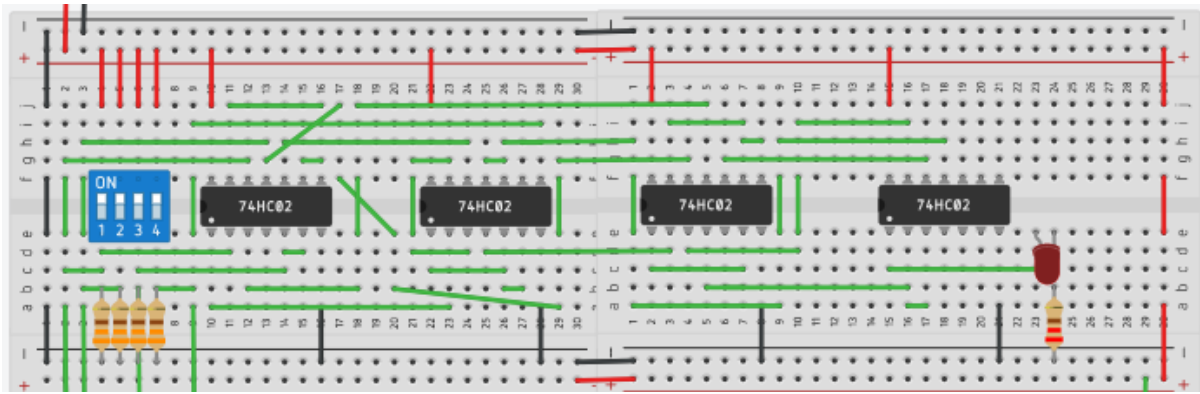


Figura 5.6 – Implementación en Lab. Virtual [Inputs (1/1/1/1) Output M (0)].

Salida I

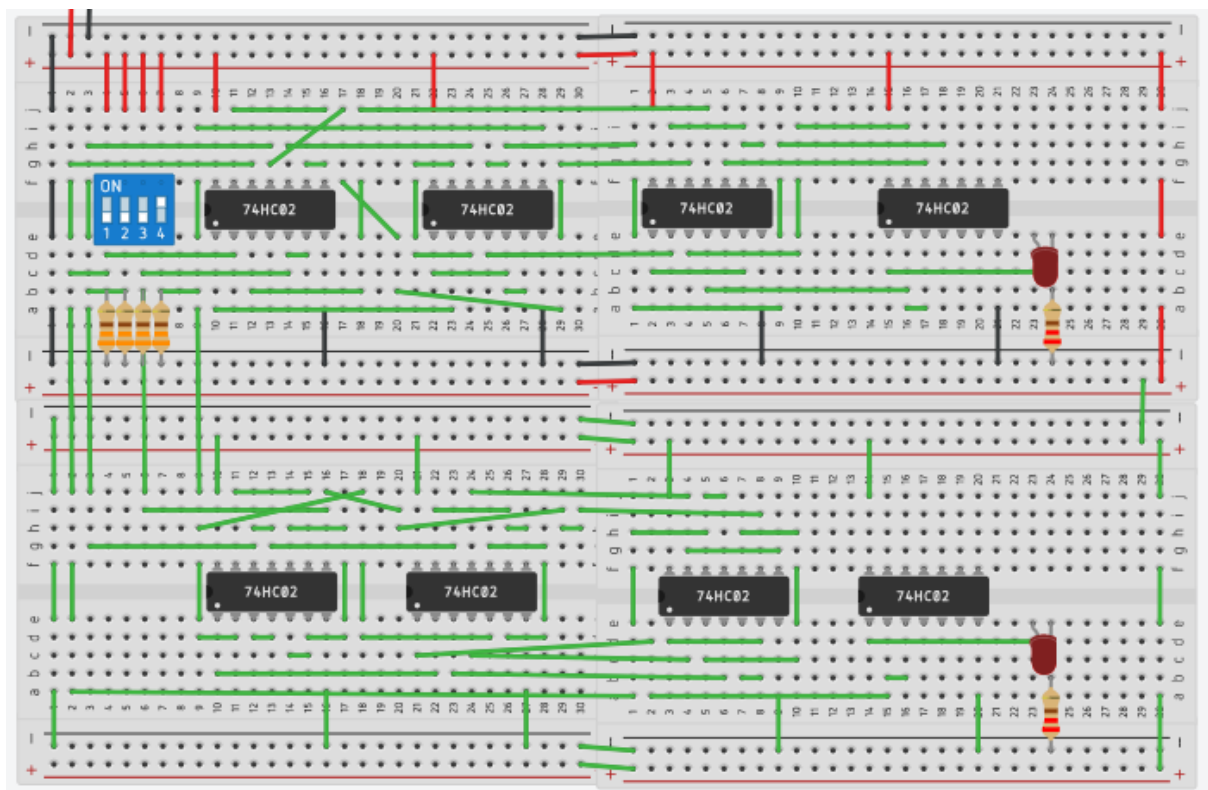


Figura 5.7 – Implementación en Lab. Virtual [Inputs (0/0/0/1) Output I (0)].

8. APORTACIONES

Existen varios ejemplos que nos dan a conocer cuán importante es la electrónica digital. A continuación, mencionaremos algunos que nos hará tener una idea más clara de lo que significa trabajar con compuertas lógicas

Electrónica Digital Restadores:

Implícitamente, a la operación de la resta va ligada la necesidad de representación de números positivos y negativos. En el caso del binario, el sistema de representación más sencillo, a priori, es el módulo-signo.

En la diferencia, cada bit del sustraendo se resta de su correspondiente bit del minuendo para formar el bit de la diferencia. El préstamo ocurre cuando el bit del minuendo es menor al bit del sustraendo, de tal forma que se presta un 1 de la siguiente posición significativa.

La resta se implementa mediante un sumador. El método consiste en llevar al minuendo a una de las entradas y el sustraendo en complemento 2 a la otra entrada.

Un medio restador es un circuito combinacional que sustrae dos bits y produce su diferencia. También tiene la salida para especificar si se ha tomado un 1. Se designa el bit minuendo por x y el bit sustraendo mediante y . Para llevar a cabo $x - y$, tienen que verificarse las magnitudes relativas de x y y . Si $x \geq y$, se tienen tres posibilidades; $0 - 0 = 0$, $1 - 0 = 1$ y, $1 - 1 = 0$. El resultado se denomina bit de diferencia. Si $x < y$, tenemos $0 - 1$ y es necesario tomar un 1 de la siguiente etapa más alta. El 1 que se toma de la siguiente etapa más alta añade dos al bit minuendo, de la misma forma que en el sistema decimal lo que se toma añade 10 a un dígito minuendo. Con el minuendo igual a 2, la diferencia llega a ser $2 - 1 = 1$. El medio restador requiere dos salidas. Una salida genera la diferencia y se denotará por el símbolo D . La segunda salida, denotada B para lo que se toma, genera la señal binaria que informa a la siguiente etapa que se ha tomado un 1. La tabla de verdad para las relaciones de entrada-salida de un medio restador ahora puede derivarse como sigue:

x	y	B	D
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

La salida que toma B es un 0 en tanto que $x \geq y$. Es un 1 para $x = 0$ y $y = 1$. La salida D es el resultado de la operación aritmética $2B + x - y$.

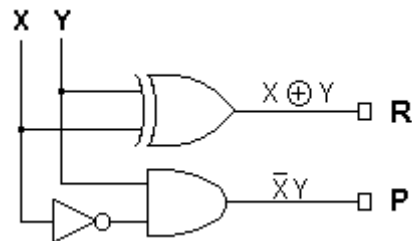
Las funciones booleanas para las dos salidas del medio restador se derivan de manera directa de la tabla de verdad:

$$D = x'y + xy'$$

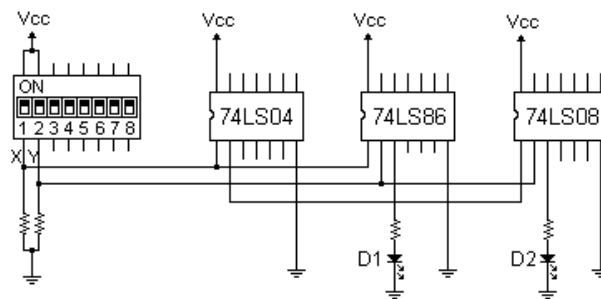
$$B = x'y$$

Es interesante observar que la lógica para D es exactamente la misma que la lógica para la salida S en el medio sumador.

El logigrama del restador es el siguiente:



Su circuito topológico es el siguiente:



Restador Electrónico completo

RESTADOR COMPLETO

Un restador completo es un circuito combinacional que lleva a cabo una sustracción entre dos bits, tomando en cuenta en un 1 se ha tomado por una etapa significativa más baja. Este circuito tiene tres entrada y dos salidas. Las tres entradas x, y y z, denotan al minuendo, sustraendo y a la toma previa, respectivamente. Las dos salidas, D y B, representan la diferencia y la salida tomada, respectivamente. La tabla de verdad para el circuito es como sigue:

x y z | B D

0 0 0 | 0 0

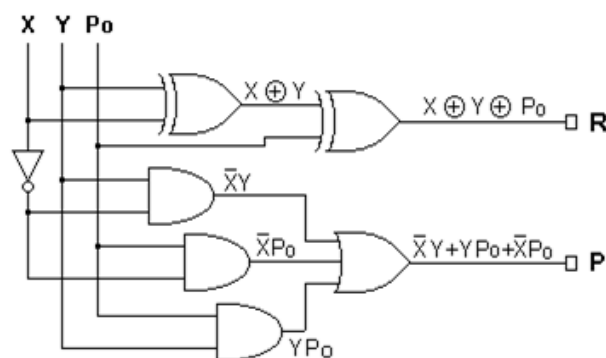
0 0 1 | 1 1

0 1 0 | 1 1

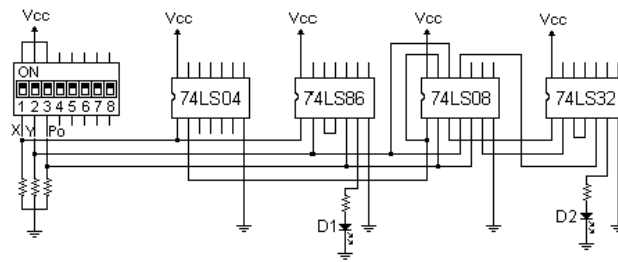
0	1	1		1	0
1	0	0		0	1
1	0	1		0	0
1	1	0		0	0
1	1	1		1	1

Los ocho renglones bajo las variables de entrada designan todas las combinaciones posibles de 1 y 0 que pueden tomar las variables binarias. Los 1 y 0 para las variables de salida están determinados por la sustracción de $x - y - z$. Las combinaciones que tienen salida de toma $z = 0$ se reducen a las mismas cuatro condiciones del medio sumador. Para $x = 0$, $y = 0$ y $z = 1$, tiene que tomarse un 1 de la siguiente etapa, lo cual hace $B = 1$ y añade 2 a x . Ya que $2 - 0 - 1$, $D = 1$. Para $x = 0$ y $yz = 11$, necesita tomarse otra vez, haciendo $B = 1$ y $x = 2$. Ya que $2 - 1 - 1 = 0$, $D = 0$. Para $x = 1$ y $yz = 01$, se tiene $x - y - z = 0$, lo cual hace $B = 0$ y $D = 0$. Por último, para $x = 1$, $y = 1$, $z = 1$, tiene que tomarse 1, haciendo $B = 1$ y $x = 3$ y, $3 - 1 - 1 = 1$, haciendo $D = 1$.

El circuito lógico implementado con compuertas es el siguiente:



El circuito topológico del restador completo es el siguiente:



9. CONCLUSIONES

- El diseño e implementación de los tres circuitos, se lo ha realizado de manera que cumpla con el objetivo de cada uno de ellos. Todo este proceso se llevó a cabo con la simplificación de las funciones lógicas mediante álgebra de Boole en los circuitos 1 y 2. Y en el tercero mediante mapas de Karnaugh, esto nos permitió realizar la correcta simulación e implementación de cada circuito en los softwares Proteus y Tinkercad. Los cuales fueron muy útiles para realizar y tener una mejor idea de cómo es el funcionamiento de cada uno de ellos. Además, que podemos, de esta manera, simular como si estuviésemos en un laboratorio físico, ya que sus similitudes son muy amplias.
- La realización del programa se la realizó en base a todo lo investigado. Prácticamente necesitamos obtener la tabla de verdad y a partir de ello, realizar el diseño de cada uno de los circuitos mediante las compuertas lógicas pertinentes. En el caso del tercer circuito, en el que se nos solicitaba realizar el diseño a partir de únicamente compuertas NOR. Se lo realizó en base al diseño original con compuertas AND, OR y NOT. De esta manera comprobamos de que elaboración del circuito con compuertas universales NOR es equivalente al circuito realizado originalmente. Sin embargo, al realizar el diseño y simulación de las funciones lógicas en Proteus, nos dimos cuenta que si comparamos los dos circuitos resultantes, el primero, con compuertas AND, OR y NOT, y el segundo, con compuertas NOR. Obtuvimos mayor cantidad de compuertas NOR. Lo que nos lleva a concluir, que, en cierta parte, es mejor utilizar un solo tipo de

compuertas lógicas, pero esto no significa que siempre, el número de compuertas disminuya.

- El diseño del primer circuito se basó en realizar una tabla de verdad en la que deben existir al menos dos entradas con un “1 lógico” para que, en la salida obtengamos un 1 lógico. En otras palabras, el avión no tendría ningún inconveniente y la seguridad que brinda a los pasajeros es óptima. En el caso del segundo circuito, se ha simulado a partir de 4 bits de entrada un “reloj”, el cual nos permitió determinar el número de turno en el que un trabajador se encontraba, en binario, mediante la representación de diodos LED, y en decimal mostrando su valor en un display de 7 segmentos. Esto se pudo lograr a partir de un codificador 7447, el cual nos permite realizar la conversión de un valor binario a decimal. Y, por último, en el tercer ejercicio, se procedió, de igual manera, a realizar la tabla de verdad, la cual constaba de 3 salidas, en las que se comparaba el valor del número A con respecto a B. Cabe recalcar que al momento de realizar la investigación con los artículos de los autores investigados, podemos darnos cuenta de la gran similitud en cuanto a aplicaciones que encontramos con respecto a nuestro trabajo realizado en esta práctica, e incluso muy interesantes, todo esto se lo ha detallado en la sección de Estado del arte.

10. RECOMENDACIONES

- Se recomienda realizar la investigación única y exclusivamente de los temas a ser tratados durante la ejecución y realización del informe que a su vez sustenten los fundamentos para la elaboración y diseño de los circuitos a realizar.
- Se recomienda analizar correctamente un circuito para aplicar el uso de compuertas universales, pues lo ideal sería aplicar estas compuertas en otro tipo de circuitos. En los que se pueda aprovechar las cuatro compuertas que contiene cada circuito integrado. Y así tratar de simplificar lo máximo posible el circuito.

- Comparar bien los Datasheet de cada componente que usemos, puesto que, si no lo verificamos, los datos a comparar pueden ser erróneos con los de nuestro circuito y esto causará que el circuito no funcione correctamente, además de esta manera podemos saber cuáles son los pines de entrada, salida y de energía, con su respectiva numeración.

11. CRONOGRAMA

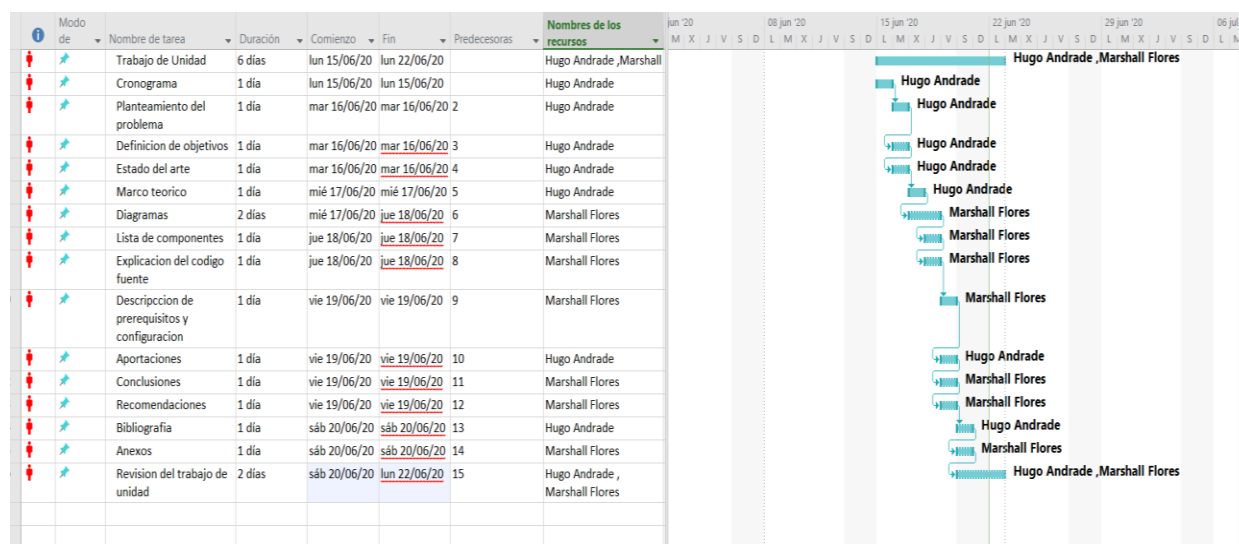


Diagrama 1.4 – Diagrama de Gantt.

12. BIBLIOGRAFÍA

Logicbus S.(1 JULIO, 2019)Logicubus.Obtenido de logicubus:

<https://www.logicbus.com.mx/blog/compuertas-logicas/>

Prof. Rodrigo Araya E. (2006)uchile. Obenido de :

https://users.dcc.uchile.cl/~clgutier/Capitulo_3.pdf

Artículos:

[1] K. Degawa, T. Aoki, T. Higuchi, H. Inokawa y A. Takahashi, (09 de agosto de 2004)

IEEEExplore.Obtenido de:

<https://sci-hub.tw/https://ieeexplore.ieee.org/document/1319952>

[2] Cavalcante (12 de noviembre de 2018) IEEEExplore.Obtenido de:

<https://sci-hub.tw/https://ieeexplore.ieee.org/document/8531697>

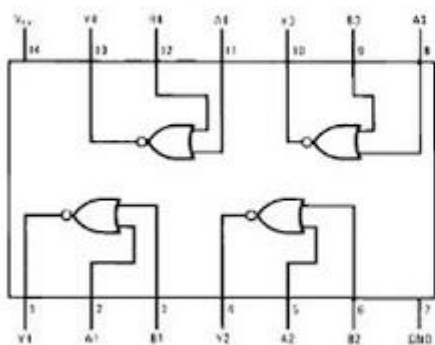
[3] M. Desani, A. Bavarva y V. Sorathiya (28 de septiembre de 2015) IEEEExplore.Obtenido de: <https://sci-hub.tw/https://ieeexplore.ieee.org/document/7275660>

13. ANEXOS

13.1. HOJAS TÉCNICAS

NOR

Connection Diagram



Function Table

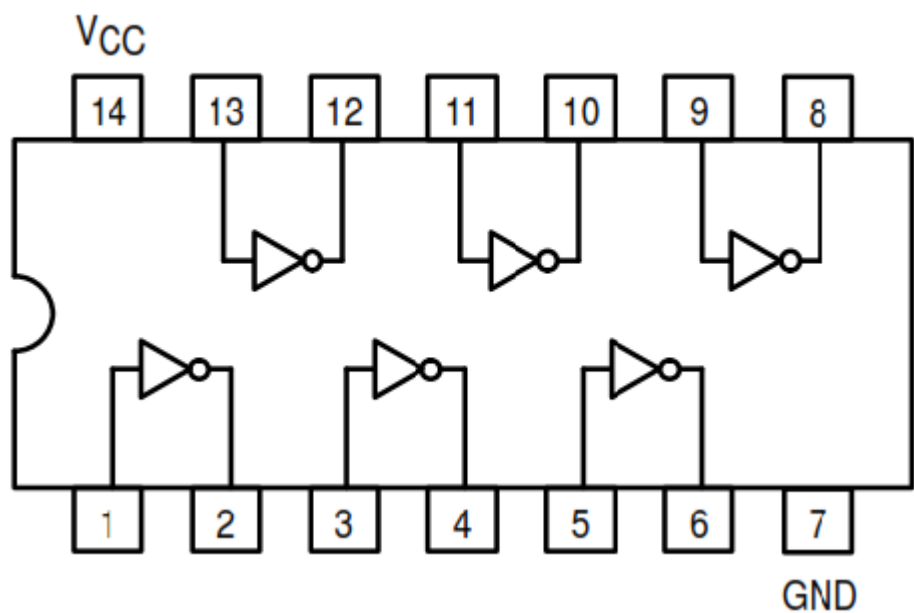
$$Y = \overline{A + B}$$

Inputs		Output
A	B	Y
L	L	H
L	H	L
H	L	L
H	H	L

H = HIGH Logic Level
L = LOW Logic Level

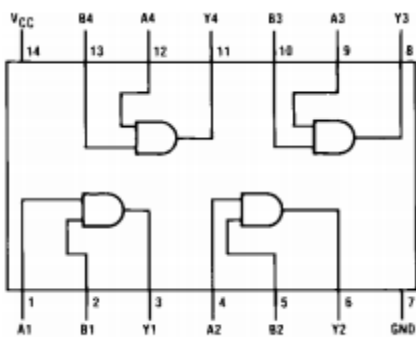
NOT

7404



OR

7432



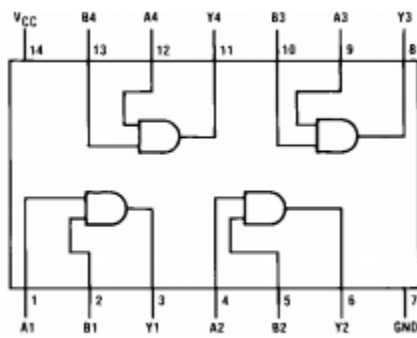
$$Y = AB$$

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = HIGH Logic Level
L = LOW Logic Level

AND

7408



$Y = AB$

Inputs		Output
A	B	Y
L	L	L
L	H	L
H	L	L
H	H	H

H = HIGH Logic Level
L = LOW Logic Level

Codificador 7447

