EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México Iztapalapa

Ingeniería en Sistemas Computacionales

REPORTE DEL PROYECTO FINAL

Tema :

compuertas lógicas y la teoría de los circuitos(concurrente)

Asignatura:

Lenguajes y Automatas 1

Profesor:

Abiel Tomas Parra Hernández

Presenta:		PORCENTAJE
Castillo Hernández Airam Yelinda	181080142	33.33%
Pacheco Perez Christoper Arturo	181080141	33.33%
Romero Solis Marco Antoni	181080187	33.33%

Junio-2021

TOTAL DE PARTICIPACIÓN ENTRE TODOS ES DE 100%

INDICE

INDICE IMÁGENES

INTRODUCCIÓN

Se realizaron las comparaciones de sistemas cotidianos aplicados a las compuertas lógicas, esto, con el fin de demostrar el funcionamiento de primera mano con este tipo de sistemas. Canalizado a manera de automatización, sobre actividades que son realizadas de manera cotidiana, ya que la mayoría de información encontrada y explorada consta de circuitos simples no aplicados completamente a una situación real.

A través de diagramas vistos en prácticas y ejercicios simples, creamos algunos circuitos los cuales nos pueden solucionar la calidad de servicios que se puede implementar tanto en una tienda, como en un cine, la manera en que despertamos e incluso el uso que puede llegar a tener una casa.

En este reporte también se disertó acerca del funcionamiento de las compuertas lógicas aplicadas a los microprocesadores, para su funcionamiento y soluciones a problemas que son computables.

Además, se vio un atisbo de lo que podríamos llegar hacer con la nanotecnología y las compuertas, esto nos puede ayudar a realizar procesamientos de mayor magnitud y ahora tener más accesos a evoluciones tecnológicas, con unos límites más difíciles de alcanzar.

Aunado a lo anterior y para recalcar, el documento demuestra a través de automatizaciones sobre procedimientos que realizamos de manera cotidiana, explicar el funcionamiento de estas compuertas que hasta dónde puede llegar la complejidad de las mismas, como la toma de decisiones de una persona que comienza con dos alternativas, las compuertas pueden realizar soluciones de una magnitud demencial.

Los ejemplos constan principalmente en tres acciones que un ser humano puede experimentar: la espera en una fila, el despertar, entrar a algún sitio o simplemente entrar a su hogar, esto enfocado a que todos los registros y resultados se van a enfocar a computarizarse, para obtener ideas para desarrollos posteriores.

Continuando y cerrando, se vieron las compuertas lógicas digitales, las cuales son completamente enfocadas en el software y como ha influido en la actualidad y los avances de tecnología.

OBJETIVOS

Objetivo General:

El siguiente documento tiene como objetivo principal dar a conocer el tema de compuertas lógicas y teoría de circuitos implementados con una teoría de autómatas.

Para poder así poder juntar la teoría de los circuitos con las compuertas lógicas de una forma eficaz y práctica, la principal razón es hacer que se pueda aprender el tema, ya que se han visto con anterioridad las compuertas lógicas y ahora se debe tomar de la mano con la teoría de circuitos. Además, se ejemplificarán con temas cotidianos la explicación y fusión de los mismos.

Objetivos específicos:

1. Conocer los conceptos básicos de las compuertas lógicas además de la teoría de circuitos, para poder saber cuál es su aplicación.
2. Realizar ejemplos u/o ejercicios para poder volver a tener práctica con los temas ya realizados.
3. Conocer las leyes de Ley de Ohm, Ley de Conservación de carga, Ley Conservación de energía además saber sobre los análisis de malla y de nodos para comprender el tema de teoría de circuitos
4. Poder emplear correctamente la aplicación de Compuertas lógicas de un circuito de acuerdo a la teoría de circuitos además de conocer y manejar su simbología.
5. Enfocar todo a prácticas simples.

RESUMEN

Este documento va a hablar de las compuertas lógicas y la teoría de los circuitos. Ya que son de utilidad para el diseño de los circuitos lógicos.

La investigación sobre las compuertas lógicas y la teoría de los circuitos serán para ver cómo trabajan juntos.

Además de poder analizar el comportamiento de ambos en este caso ya sea combinados o por separado.

Lo primero que se hará será poder describir y poder analizar las compuertas lógicas para poder hacer los análisis sobre el comportamiento sobre las combinaciones que puede haber unas con otras concitándose en un circuito, de igual forma con la teoría de circuitos se va a tener que analizar y describir este tema para poder saber los fundamentos para los análisis de los circuitos eléctricos que esto permite determinar los niveles de tensión y corriente en cada punto del circuito en respuesta a un determinado voltaje.

PALABRAS CLAVES:

Compuertas, circuitos, comportamiento, combinaciones, concitándose, eléctricos, tensión, corriente, voltaje

ABSTRACT

This document is going to talk about logic gates and circuit theory. Since they are useful for the design of logic circuits.

Research on logic gates and circuit theory will be to see how they work together.

In addition to being able to analyze the behavior of both in this case, either combined or separately.

The first thing that will be done being able to describe and analyze the logic gates and be able to do the analysis on the behavior on the combinations that may exist with each other conceiting in a circuit, in the same way with circuit theory the document will have to be analyzed and described this topic to know the fundamentals for the analysis of electrical circuits that this allows to determine the levels of voltage and current at each point of the circuit in response to a certain voltage.

Keywords:

Gates, circuits, behavior, combinations, consisting of, electrical, voltage, current, voltage

JUSTIFICACIÓN

En internet no hay demasiada información aplicada de manera simple y concisa, simplemente se resume en dos caminos: algo muy complejo que los estudiantes novatos no entienden o algo tan simple que los estudiantes intermedios no lo sienten lo suficientemente desafiante y que mejor manera de hacerlo que automatizando y computarizando procesos que cualquier persona puede hacer en el día, aunque claro, como en todo hay excepciones y es por ello que nos abre nuevos límites de igual forma.

También se germinarán nuevas ideas que en un futuro puedan ser servibles tanto como en seguridad, como en control de accesos. Siendo así este documento tiene con una finalidad poder dar la información necesaria para poder definir y analizar los conceptos de las compuertas lógicas así como su aplicación y su combinación entre ellas dando así ejemplificaciones de distintitos circuitos digitales así como su teoría de los circuitos, esto será con la finalidad de poder resolver problemas que tengan que ver con las compuertas lógicas y la teoría de los circuitos, ya que la finalidad de las teoría de los circuitos es saber sobre las corrientes electromagnéticas y las compuertas lógicas están basadas en los circuitos que son los que ocupan para que puedan funcionar y depende de eso se verá si son paralelos o en serie.

Todo esto para que posterior mente se puedan llevar acabo y con facilidad ya explicados de forma simple las funciones, las leyes y todo lo que se pueda llegar a utilizar.

De la misma forma se brindara información para el entendimiento general al mundo de la electrónica siendo las telecomunicaciones, la informática, etc. Las cuales son áreas beneficiadas por la electrónica digital como lo es la formación base de los tecnólogos e ingenieros.

METODOLOGIA A UTILIZAR

¿Qué es el Extreme Programming? Extreme Programming o XP Programming es un marco de desarrollo de software ágil que tiene como objetivo producir un software de mayor calidad para mejorar la eficiencia del equipo de desarrollo. Se trata de una metodología de desarrollo cuyo objetivo es promover la aplicación de prácticas de ingeniería apropiadas para la creación de software. Esta metodología la formuló Kent Beck, autor del primer libro sobre este ámbito llamado «Extreme Programming Explained: Embrace Change», publicado en 1999. Extreme Programming fue la metodología dominante en el mundo ágil en los años 2.000. Luego, el framework Scrum le quitó el puesto en cuanto a popularidad. A pesar de ello, se trata de una de las metodologías ágiles de desarrollo de software más exitosas. La combinación de esta, junto con la Scrum, asegura un enorme control sobre los proyectos y una implementación mucho más efectiva. Para que nos hagamos una idea, Extreme Programming está diseñada para ofrecer el software que los usuarios necesitan en el momento adecuado. En este sentido, ayuda a los desarrolladores a ajustarse a los requerimientos cambiantes de los clientes. Este tipo de programación se diferencia de las metodologías tradicionales en que pone más énfasis en la adaptabilidad que en la previsibilidad. El Extreme Programming considera que los cambios de requisitos sobre la marcha son acciones naturales e inevitables en el desarrollo de un proyecto. Creen que ser capaces de adaptarse a los cambios que puedan surgir en cualquier punto del ciclo de vida de un proyecto es una mejor previsión y más realista que intentarlos definir todos en un principio y que no varíen más. Diseño y programación El diseño del programa suele ser simple y basado en la funcionalidad del sistema y se lleva a cabo durante todo el proyecto, tanto durante la planificación de la entrega como en el de la iteración. La programación del software se hace siempre en pareja, lo que se llama programar a dos manos. Se asegura con este método que al menos un programador conoce y controla la labor de otro y queda revisado. La ventaja es que se produce mejor código que en base a un programador aunque la dificultad de la misma sea mayor. El código es de todos, con el desarrollo de las pruebas automáticas y la programación a dos manos se incluye también la posibilidad de que cualquiera pueda añadir y retocar parte del código, aunque eso sí, deba ser un estilo común y cuyo resultado sea como si sólo lo hubiera hecho una persona. El Extreme Programming tiene como gran ventaja el de la programación organizada y planificada para que no haya errores durante todo el proceso. Los programadores suelen estar satisfechos con esta metodología. Es muy recomendable efectuarlo en proyectos a corto plazo.

El **Extreme Programming** tiene como gran ventaja el de la programación organizada y planificada para que no haya errores durante todo el proceso. Nos basamos en este método para la materia de autómatas si se presenta un trabajo de programación porque así nos basaríamos en un solo código y podríamos mejorar ese código entre todos los integrantes de un equipo, fusionando ideas y mejorándolas para beneficio del proyecto. Es muy recomendable efectuarlo en proyectos a corto plazo.

Ilustración 2

Ilustración 1

Nos agrada Lean ya que la eficacia del trabajo es buena con este método, fomenta la participación y el aprendizaje de cada integrante del equipo, para seguir este método todo va a enfocado a las exigencias de un cliente y así planteando diferentes soluciones llegar a está de la mejor manera, una solución eficaz. Por eso lean es el mejor método para mi equipo de trabajo, ya que todos podemos llevarnos una experiencia con aprendizaje y mejorar en nuestras relaciones laborales.

MARCO TEORICO

Teoría de la Computación.

La teoría de la computación comienza propiamente a principios del siglo XX, poco antes que las computadoras electrónicas fuesen inventadas. En esta época varios matemáticos se preguntaban si existía un método universal para resolver todos los problemas matemáticos. Para ello debían desarrollar la noción precisa de método para resolver problemas, es decir, la definición formal de algoritmo. La teoría de la computación o teoría de la informática es un conjunto de conocimientos racionales y sistematizados que se centran en el estudio de la abstracción de los procesos, con el fin de reproducirlos con ayuda de sistemas formales; es decir, a través de símbolos y reglas lógicas. La teoría de la computación permite modelar procesos dentro de las limitaciones de dispositivos que procesan información y que realizan cálculos; como, por ejemplo, el ordenador. Para ello, se apoya en la teoría de autómatas, a fin de simular y estandarizar dichos procesos, así como para formar los problemas y darles solución. Se va a dividir en subramas que son las siguientes:

Teoría de autómatas: Esta teoría da modelos matemáticos que formalizan el concepto de computadora o algoritmo de manera simplificada o general para que se puedan analizar sus capacidades y limitaciones. Algunos de estos modelos juegan un papel en varias aplicaciones de las ciencias de la computación, incluyendo procesamiento de texto, compiladores, diseño de hardware e inteligencia artificial. Esto da lugar a que se piense en la máquina de Turing como el modelo universal de computadora. Teoría de computabilidad: es la parte de la computación que estudia los problemas de decisión que se pueden resolver con un algoritmo o equivalentemente con una máquina de Turing.

Teoría de la complejidad computacional: estudia las necesidades de memoria, tiempo y otros recursos computacionales para resolver problemas; de esta manera es posible explicar por qué unos problemas son más difíciles de resolver que otros.

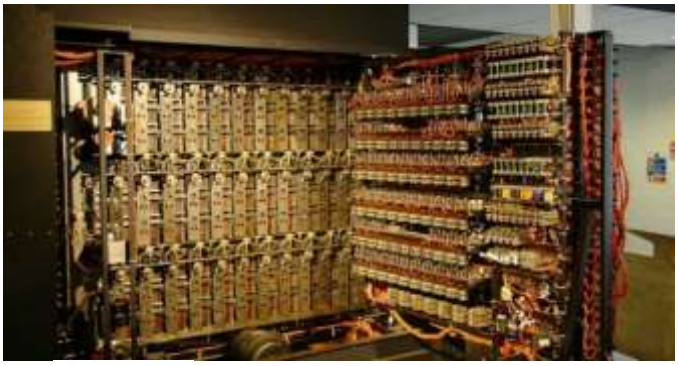


Ilustración 4

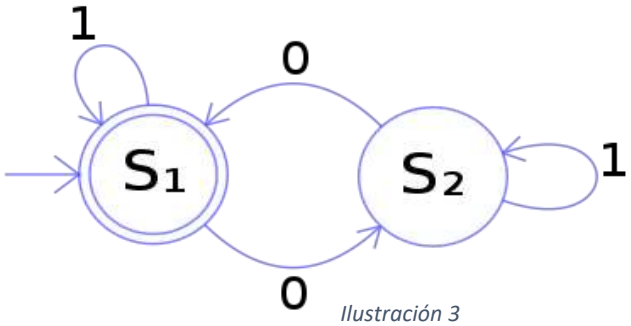


Ilustración 3

Teoría de computabilidad:

Es el estudio matemático de los modelos de computación. Se originó en la década de los años 30 con los trabajos de los lógicos Church, Gödel, Kleene, Post y Turing. La teoría de la computabilidad es la parte de la computación que estudia los problemas de decisión que se pueden resolver con un algoritmo o equivalentemente con una máquina de Turing. La teoría de la computabilidad, también conocida teoría de la recursión, es una de las cuatro partes que constituyen la lógica matemática, las otras tres son: la teoría de conjuntos, la teoría de modelos y la teoría de la demostración, se ocupa del estudio y clasificación de las relaciones y aplicaciones computables. Además, la teoría de la computabilidad, junto con la teoría de autómatas, lenguajes y máquinas, es el fundamento de la informática teórica y además también de los ordenadores. Se dividen en 3 partes que son los siguientes:
Los computables son aquellos para los cuales sí existe un algoritmo que siempre los resuelve cuando hay una solución y además es capaz de distinguir los casos que no la tienen. También se les conoce como decidibles, resolubles o recursivos.
Los semi-computables son aquellos para los cuales hay un algoritmo que es capaz de encontrar una solución si es que existe, pero ningún algoritmo que determine cuando la solución no existe (en cuyo caso el algoritmo para encontrar la solución entraría a un bucle infinito). El ejemplo clásico por excelencia es el problema de la parada. A estos problemas también se les conoce como listables, recursivamente nuemrables o reconocibles, porque si se enlistan todos los casos posibles del problema, es posible reconocer a aquellos que sí tienen solución.
Los incomputables son aquellos para los cuales no hay ningún algoritmo que los pueda resolver, no importando que tengan o no solución. El ejemplo clásico es el problema de la implicación lógica, que consiste en determinar cuándo una proposición lógica es un teorema; para este problema no hay ningún algoritmo que en todos los casos pueda distinguir si una proposición o su negación es un teorema.

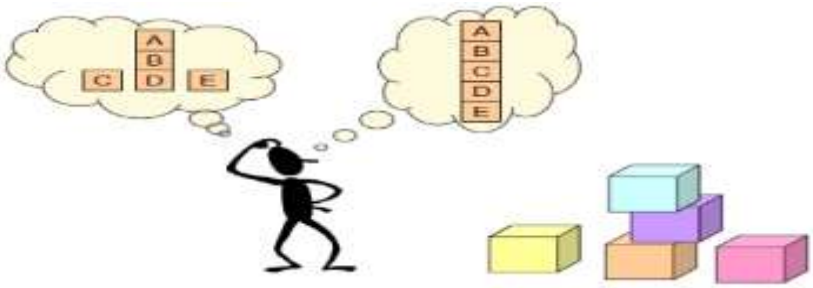
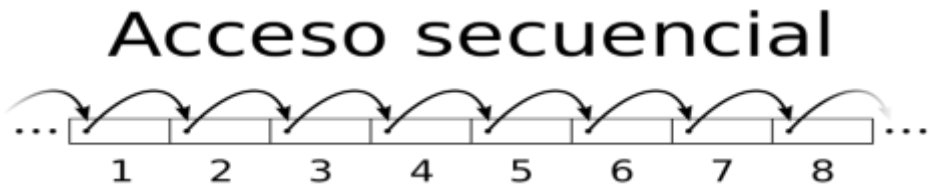


Ilustración 5

Modelos de computación:

Un modelo de computación es la definición un conjunto de operaciones permitidas a usar en el cómputo y sus respectivos costos. Asumiendo un cierto modelo de computación es posible analizar los recursos de cómputo requeridos, como el tiempo de ejecución o el espacio de memoria, o discutir las limitaciones de algoritmos o computadores.
El modelo de computación explica cómo el comportamiento del sistema entero es el resultado del comportamiento de cada uno de sus componentes.
Un modelo computacional en términos de operaciones primitivas permitidas que tengan un costo unitario, o simplemente operaciones costo unitario. Un ejemplo comúnmente usado es la máquina de acceso aleatorio, que tiene costo unitario para acceso de lectura y escritura para todas sus celdas de memoria. En este respecto, se diferencia del modelo de máquina de Turing.
Estos modelos se dividen en 3 tipos que son:
Secuencial: asume que solamente una operación puede ser ejecutada a la vez, y eso es cierto para una sola computadora con un solo procesador. Sin embargo, la mayoría de las computadoras modernas tienen procesadores multi-núcleo, donde cada núcleo puede ejecutar operaciones de forma independiente.

Ilustración 6



Funcional:
El modelo funcional especifica lo que sucede, el modelo dinámico cuándo sucede, y el modelo de objetos sobre qué entidades sucede.
Concurrente: La computación concurrente es una forma de cómputo en la cual varios cálculos se realizan concurrentemente, y no uno a la vez de forma secuencial. Es una característica propia de un sistema, ya sea un programa, una computadora o una red, en la cual existe un punto separado de ejecución o "hilo de control" para cada proceso. Un sistema concurrente es aquel donde un cálculo puede avanzar sin esperar a que el resto de los cálculos se completen.

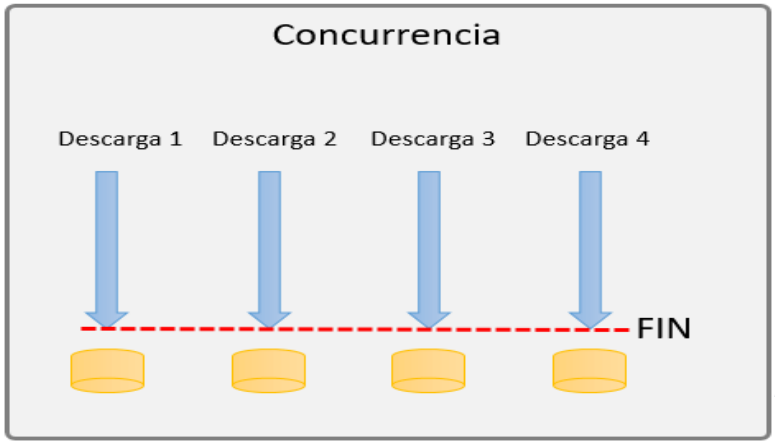


Ilustración 7

Compuertas lógicas y teoría de circuitos

Las computadoras digitales utilizan el sistema de números binarios, que tiene dos dígitos 0 y 1. Un dígito binario se denomina un bit.

La información está representada en las computadoras digitales en grupos de bits. Utilizando diversas técnicas de codificación los grupos de bits pueden hacerse que representen no solamente números binarios sino también otros símbolos discretos cualesquiera, tales como dígitos decimales o letras de alfabeto.

Utilizando arreglos binarios y diversas técnicas de codificación, los dígitos binarios o grupos de bits pueden utilizarse para desarrollar conjuntos completos de instrucciones para realizar diversos tipos de cálculos. La información binaria se representa en un sistema digital por cantidades físicas denominadas señales, Las señales eléctricas tales como voltajes existen a través del sistema digital en cualquiera de dos valores reconocibles y representan una variable binaria igual a 1 o 0. Por ejemplo, un sistema digital particular puede emplear una señal de 3 volts para representar el binario "1" y 0.5 volts para el binario "0". La región intermedia entre las dos regiones permitidas se cruza solamente durante la transición de estado. Los terminales de entrada de un circuito digital aceptan señales binarias dentro de las tolerancias permitidas y los circuitos responden en los terminales de salida con señales binarias que caen dentro de las tolerancias permitidas.

La lógica binaria tiene que ver con variables binarias y con operaciones que toman un sentido lógico. La manipulación de información binaria se hace por circuitos lógicos que se denominan Compuertas. La teoría de circuitos es aquella que comprende los fundamentos para el análisis de circuitos eléctricos y permite determinar los niveles de tensión y corriente en cada punto del circuito en respuesta a una determinada excitación. La teoría de circuitos es una simplificación de la teoría electromagnética de J. C. Maxwell, estas simplificaciones se basan en la consideración de corrientes lo que implica que sólo puede aplicarse cuando la longitud de onda de las señales ondas electromagnéticas presentes en el circuito es mucho mayor, que las dimensiones físicas de éste. Esto quiere decir que la propagación de las ondas en el circuito es instantánea. A estos circuitos a veces se les llama circuitos de parámetros concentrados.

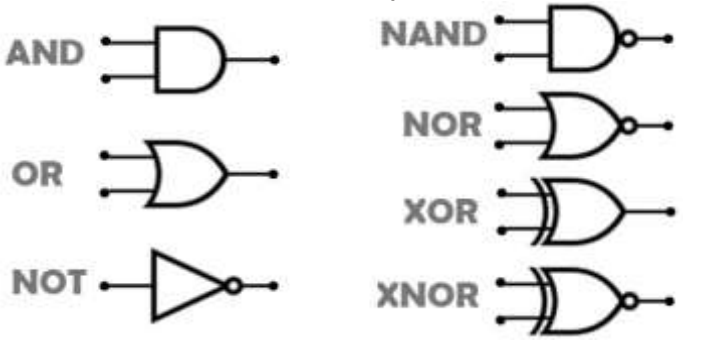


Ilustración 8

Ilustración 9

Tipos de compuertas lógicas y su funcionamiento

Ilustración 10

Compuerta AND Cada compuerta tiene dos variables de entrada designadas por A y B y una salida binaria designada por x. La compuerta AND produce la multiplicación lógica AND: esto es: la salida es 1 si la entrada A y la entrada B están ambas en el binario 1: de otra manera, la salida es 0.

Ilustración 11

La compuerta OR produce la función sumadora, esto es, la salida es 1 si la entrada A o la entrada B o ambas entradas son 1; de otra manera, la salida es 0.

Ilustración 12

La compuerta Not Si la variable binaria posee un valor 0, la compuerta NOT cambia su estado al valor 1 y viceversa.

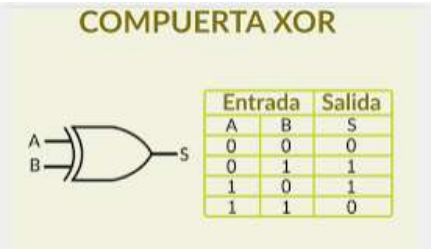
Compuertas separadas por un if, buffer o yes Un símbolo triángulo por sí mismo designa un circuito separador, el cual no produce ninguna función lógica particular puesto que el valor binario de la salida es el mismo de la entrada. Este circuito se utiliza simplemente para amplificación de la señal.

Ilustración 13

COMPUERTA NAND (NO Y) Es el complemento de la función AND, como se indica por el símbolo gráfico, que consiste en una compuerta AND seguida por un pequeño círculo (quiere decir que invierte la señal).

Ilustración 14

COMPUERTA NOR (NO O) La compuerta NOR es el complemento de la compuerta OR y utiliza el símbolo de la compuerta OR seguido de un círculo pequeño (quiere decir que invierte la señal).



COMPUERTA XOR (O EXCLUSIVO) La puerta lógica OR-exclusiva, más conocida por su nombre en inglés XOR, realiza la función booleana A'B+AB'. Su símbolo es (signo más "+" inscrito en un círculo).

Ilustración 15

COMPUERTA NXOR (NO O EXCLUSIVO) Una compuerta NXOR no es más que una XOR con su salida negada, por lo que su salida estará en estado alto solamente cuando sus entradas son iguales, y en estado bajo para las demás combinaciones posibles.

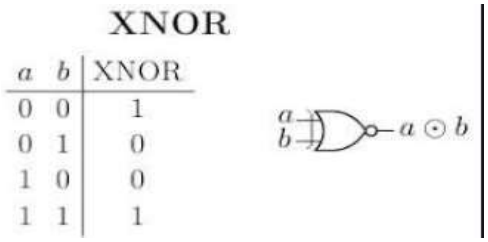


Ilustración 16

COMO SERIA LA APLICACION DEL METODO CON LA MAQUINA DE TURING

¿Qué es la máquina de Turing?

Una máquina de Turing es un dispositivo que manipula símbolos sobre una tira de cinta de acuerdo con una tabla de reglas. A pesar de su simplicidad, una máquina de Turing puede ser adaptada para simular la lógica de cualquier algoritmo de computador y es particularmente útil en la explicación de las funciones de una CPU dentro de un computador.

Nosotros fusionando con el proyecto antes mencionado, es posible aplicarlo a manera de conteo guardado en un arreglo (array), por lo cual, funcionara de tal modo que cada registro se almacene en el mismo, esto quiere decir, cada que el circuito termine su funcionamiento se registrara en el arreglo antes mencionado a manera de programar algún tipo de recompensa para limpiar el circuito. Por ejemplo: Si una persona con el despertador de tapete lo logra despertar a la hora que es 10 veces el sistema le permitirá cinco minutos más de descanso cada 10 veces completado el circuito, una vez cobrada la recompensa el sistema formateara el circuito para empezar de nuevo.

Ilustración 17

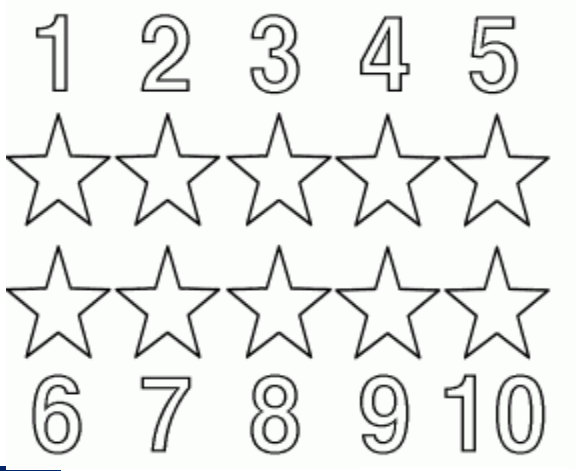


Ilustración 19

Ilustración 18

CÒDIGO FUENTE

```
<?xml version="1.0" encoding="UTF-8"?>
<classpath>
	<classpathentry kind="src" path="src"/>
	<classpathentry kind="con"
path="org.eclipse.jdt.launching.JRE_CONTAINER"/>
	<classpathentry kind="output" path="bin"/>
</classpath>
@@ -0,0 +1 @@
/bin/
<?xml version="1.0" encoding="UTF-8"?>
<projectDescription>
	<name>MyFrame</name>
	<comment></comment>
	<projects>
	</projects>
	<buildSpec>
		<buildCommand>
			<name>org.eclipse.jdt.core.javabuilder</name>
			<arguments>
			</arguments>
		</buildCommand>
	</buildSpec>
	<natures>
		<nature>org.eclipse.jdt.core.javanature</nature>
	</natures>
</projectDescription>
import java.awt.Graphics;
import java.util.ArrayList;
```

```
public abstract class Compuertas{
    /**
    *Almacena un entero consecutivo que se encarga de identificar a dicha compuerta.
    */
   protected int identificador;
   /**
    *Punto que se encarga de guardar la posicion especifica en la que se debe
    * dibujar la compuerta en una ventana grafica.
    * Almacena la posicion de la esquina superior izquierda.
    */
   public Punto pos;
   /**
    *ArrayList de Pin  que se encarga de almacenar los pines que corresponden
    * a dicha compuenrta.
    */
   public ArrayList <Pin> terminales= new ArrayList();
   public Compuertas(){}
   /**
    *Constructor de Compuertas
    * @param _identificador Entero que contiene el codigo consecutivo con el que se identifica a la compuerta.
    * @param ent Entero que se encarga de contar la cantidad de entradas que tiene la compuerta.
    * @param sal Entero que se encarga de contar la cantidad de entradas que tiene la compuerta.
    * @param _pos Punto que se encarga de dar la posicion en la que se encuentra la compuerta.
    */
   public Compuertas(int _identificador, int ent, int sal, Punto _pos){
```

```
        identificador = _identificador;
        pos=_pos;
        for (int i = 0; i < ent; i++) {
            Pin pin = new Pin(true);
            terminales.add(pin);
        }
        for (int i = 0; i < sal; i++) {
            Pin pin = new Pin(false);
            terminales.add(pin);
        }
        pos_pin();
        MyPanel.ocupa_com(this);
    }
    /**
     * Posiciona los pines de cada compuerta en su respectiva posicion respecto a su ubicacion especifica.
     */
    public  void pos_pin(){
        int ent=0;
        for (Pin pin: terminales) {
            if (pin.flujo==true) {
                if (ent==0) {
                    pin.pos.x=this.pos.x;
                    pin.pos.y=this.pos.y+10;
                    ent=1;
                } else {
                    pin.pos.x=this.pos.x;
                    pin.pos.y=this.pos.y+30;
                }
```

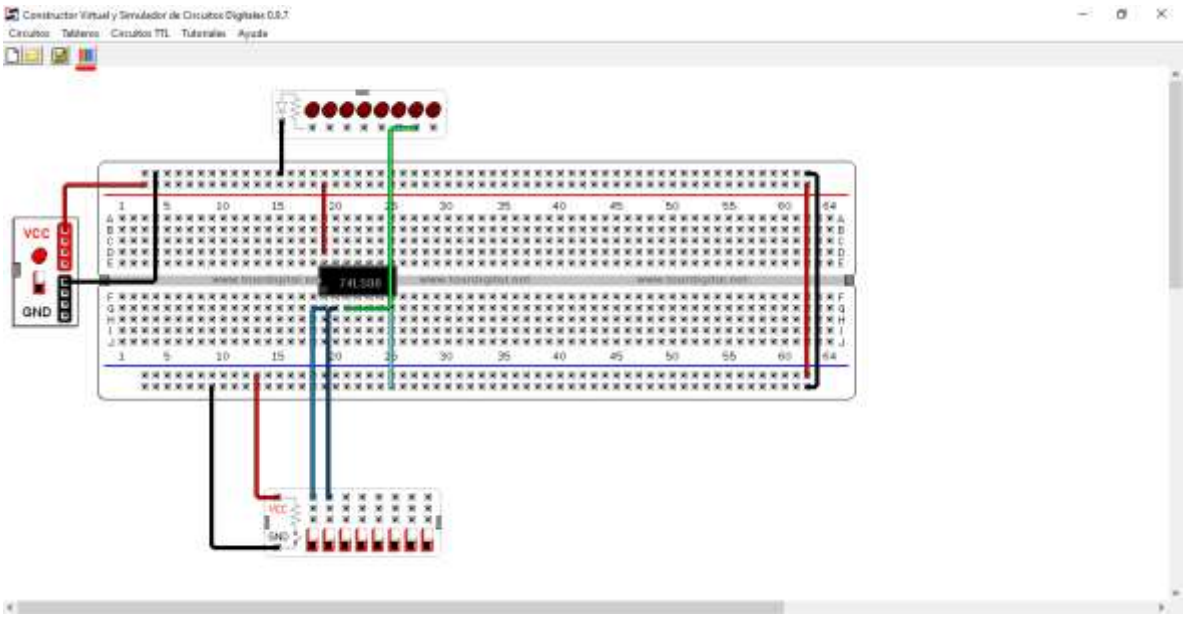
```
        } else {
            pin.pos.x=this.pos.x+90;
            pin.pos.y=this.pos.y+20;
        }
    }
};
/**
 *Se encarga de revisar los valores almacenados en los pines, solicitar
 * los faltantes y evaluar dicha compuerta de acuerdo a su respectiva logica.
 * @return Valor resultante despues de la evaluacion logica de la compuerta.
 */
public boolean evaluar(){
    boolean a = false, b = false, cont = false, sal = false;
    revisar_entradas();
    for (Pin p:terminales) {
        if (p.flujo==true) {
            if (cont==false) {
                a=p.valor;
                cont=true;
            } else {
                b=p.valor;
            }
        }
    }
    sal=logica(a,b);
    for (Pin p:terminales) {
        if (p.flujo==false) {
            p.valor=sal;
```

```
            p.evaluado=true;
        }
    }
    return sal;
};
/**
 *Se encarga de revisar cada uno de los pines y sus valores, si algun valor
 * no esta evaluado se encarga de evaluar a su respectiva compuerta y
 * asignarle el valor resultante.
 */
public void revisar_entradas(){
    for (Pin p:this.terminales) {
        if (p.flujo==true) {
            if (p.evaluado==false) {
                for (Compuertas c : MyPanel.circuito){
                    if (p.identificador==c.identificador) {
                        p.valor=c.evaluar();
                        p.evaluado=true;
                    }
                }
            }
        }
    }
}
/**
 *Se encarga de dibujar graficamente a cada una de las compuertas de su
 * respectiva forma.
 * @param g Parametro grafico de java.
```

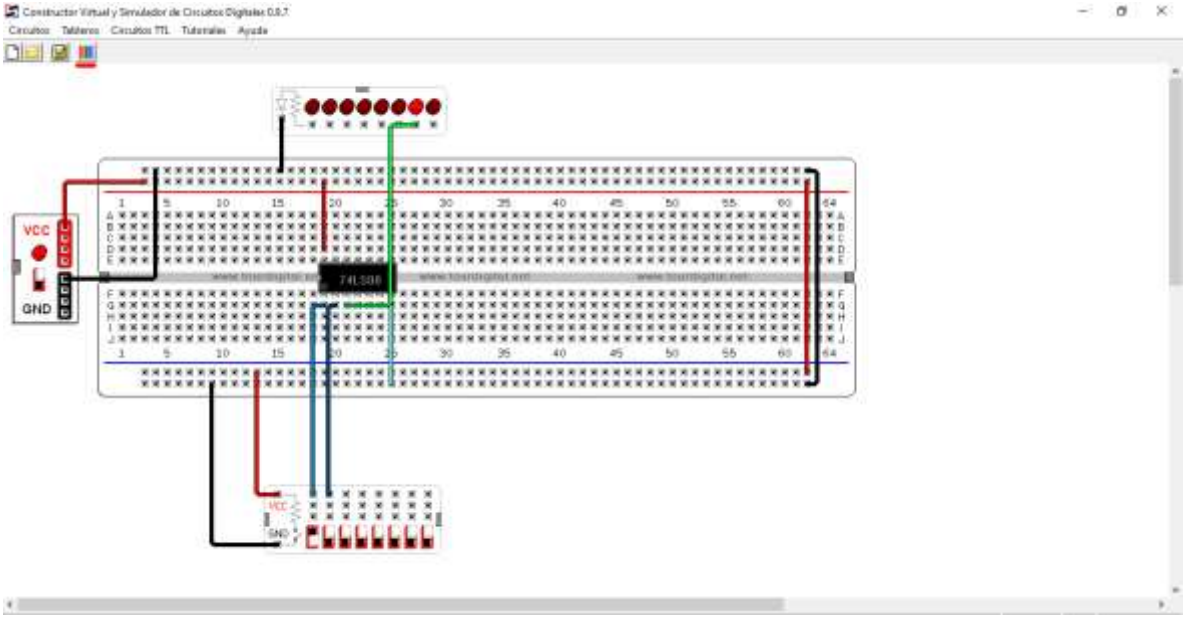
```
     */
    public abstract void dibujar(Graphics g);
    /**
     *Se encarga de realizar el proceso de logica de cada una de las compuertas
     * que usan dos parametros logicos.
     * @param a Primer paramero logico para la evaluacion.
     * @param b Primer paramero logico para la evaluacion.
     * @return resultado de la evaluacion logica de los parametros de acuerdo
     * a cada condicion logica.
     */
    public abstract boolean logica(boolean a, boolean b);
public class MyFrame {
        //Sera el main del programa donde se ejecutara el entorno de este.
}
```

RESULTADOS

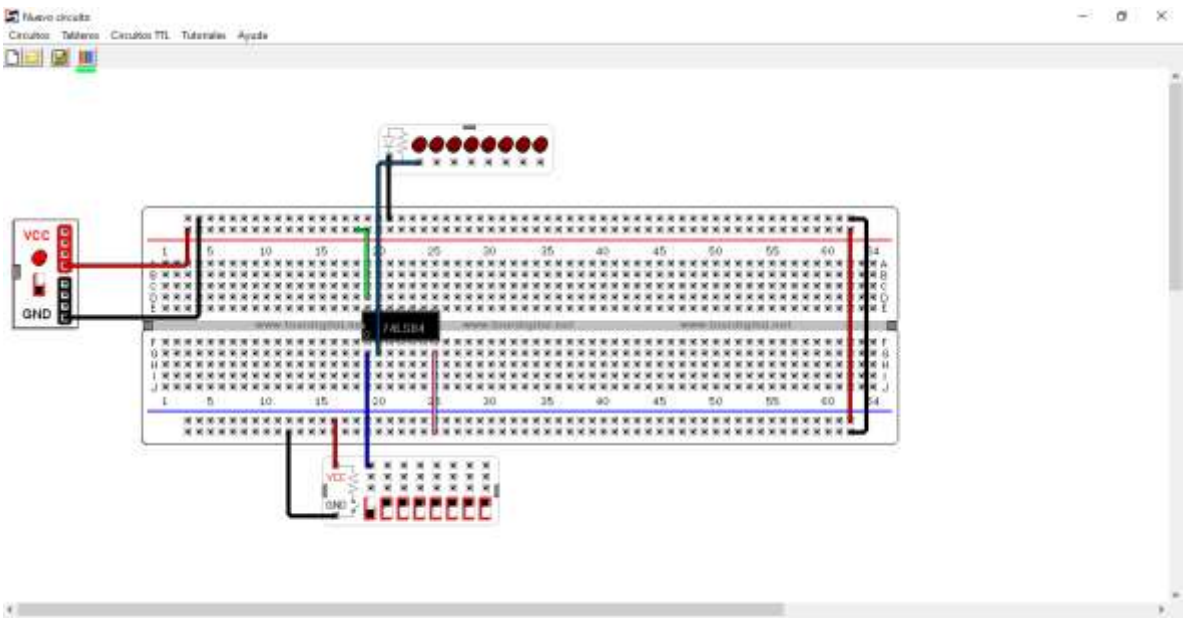
Circuito integrado apagado compuerta NAND



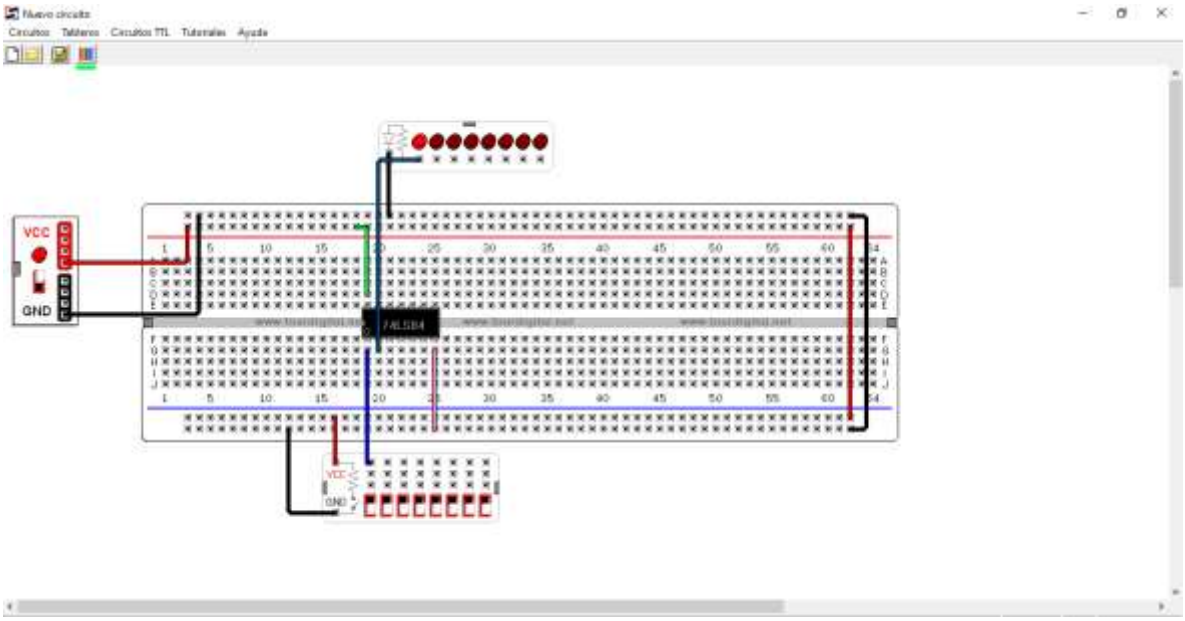
Circuito integrado encendido compuerta NAND

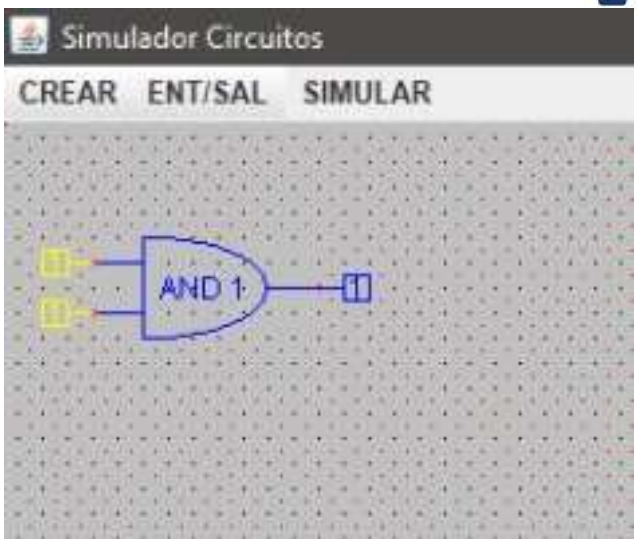


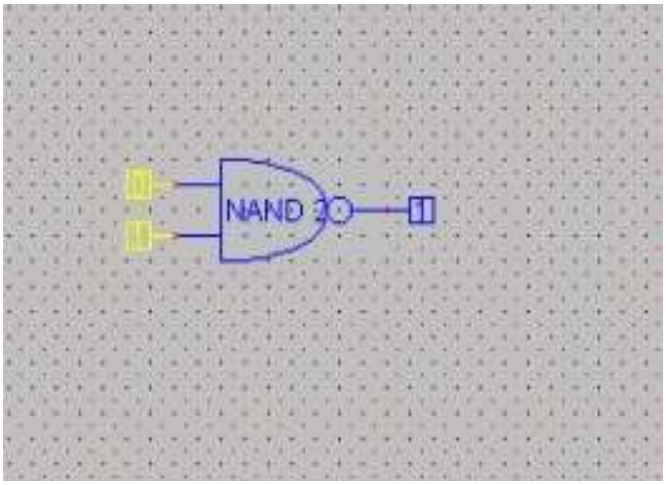
Circuito integrado NOT apagado

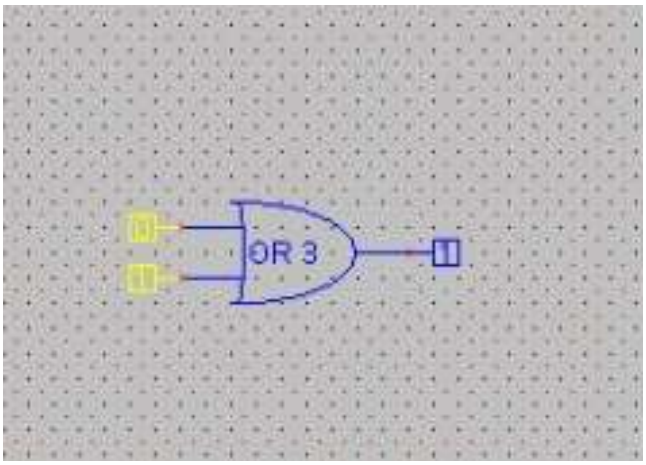


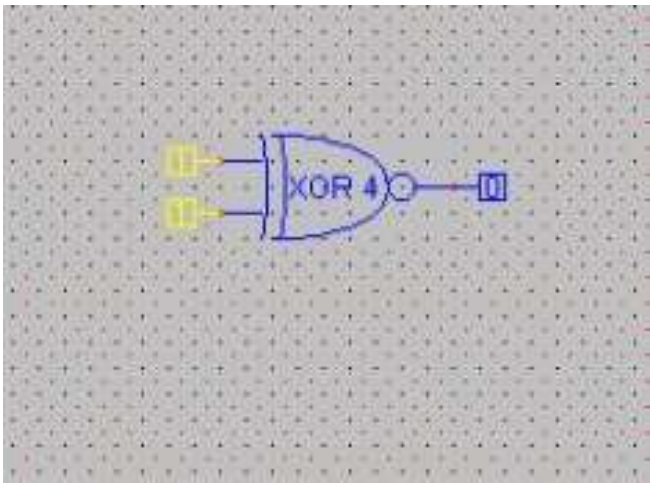
Circuito integrado NOT encendido

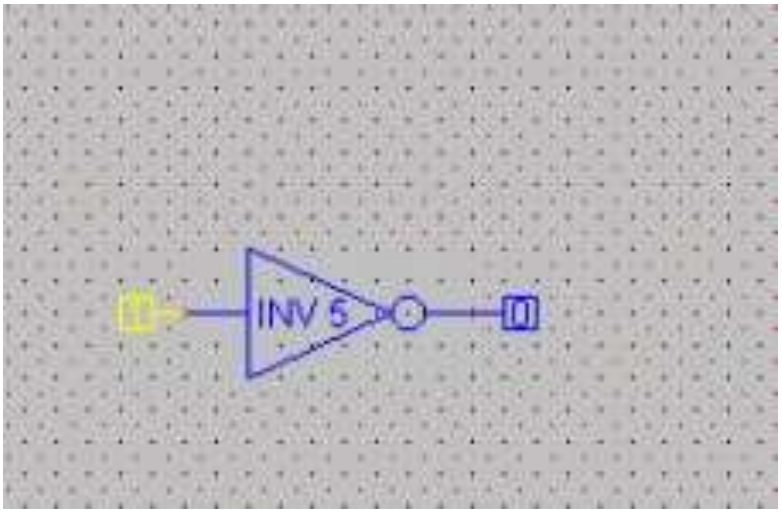



Simulador Circuitos
CREAR
ENT/SAL
SIMULAR
AND 1


NAND 2


OR 3


XOR 4


INV 5

CONCLUSIÓN

La mayoría de procesos y acontecimientos en nuestra vida diaria pueden ser representados de manera matemática, pero, en distintas ocasiones, el ser humano tiene muchas limitantes para ello, por ende, construye e innova de tal manera en que el mismo ser humano pueda resolver problemas de mayor magnitud, esto nos ha permitido conocer nuevas y sustanciales consecuencias.

Parte de estas consecuencias es computarizar además de automatizar diferentes procesos, abriéndonos paso a otro tipo de pensamiento y soluciones en el tramo de tiempo en que algo existe, todos estos procesos necesitan llevar un control, por la misa situación que son computarizados es necesario registrar y comprobar los resultados de los mismos.

Aunado a lo anterior, estos procesos pueden ser computarizados, ya que las computadoras y la computación, aunque parezcan que la manera en que se comunican es muy simple, nos ha permitido crear tantas ramas y tantos conocimientos que tuvieron que ser clasificados como aparte, este trabajo demostró que es posible hacerlos con procesos tan simple y burdos, que se pueden ir haciendo tan complejos, lo suficiente para logar simular situaciones y mundos complejos y completos.

Aún hay mucho más por descubrir e innovar y es trabajo de todos compartir la información y conocimientos a los que busquen respuestas.

REFERENCIAS

A. (2019, 5 noviembre). *Compuerta AND, funcionalidad y ejemplos – ProLogic*. Prologic. https://www.germanmadrid.com/2019/11/05/compuerta-and/

circuitos electricos. (s. f.). recursostic. Recuperado 4 de junio de 2021, de http://recursostic.educacion.es/secundaria/edad/4esotecnologia/quincena6/pdf/quincena6.pdf

colaboradores de Wikipedia. (2019, 21 octubre). *Teoría de circuitos*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Teor%C3%ADa_de_circuitos#:%7E:text=En%20ingenier%C3%ADa%20el%C3%A9ctrica%2C%20la%20teor%C3%ADa,respuesta%20a%20una%20determinada%20excitaci%C3%B3n.

compuertas logicas. (s. f.). logicbus. Recuperado 4 de junio de 2021, de https://www.logicbus.com.mx/blog/compuertas-logicas/index.php

Electronica digital. (s. f.). electronica. Recuperado 4 de junio de 2021, de https://www.uv.mx/instru/files/2013/11/S-ELECTRONICA_DIGITAL.pdf

electronica digital. (s. f.). uv instrufiles. Recuperado 4 de junio de 2021, de https://www.uv.mx/instru/files/2013/11/S-ELECTRONICA_DIGITAL.pdf

practicas. (s. f.). practicas4. Recuperado 4 de junio de 2021, de http://weblidi.info.unlp.edu.ar/catedras/organiza/descargas/Practica4.pdf

T. (2020, 1 agosto). *Compuertas lógicas: ¿Qué son?, ¿Cómo funcionan?, ¿Para qué sirven?* Actualidad Tecnologica. https://actualidadtecnologica.com/compuertas-logicas/#Que_es_una_compuerta_logica

Castaño, L. (2017, 30 marzo). *Compuertas logicas*. Compuertas Logicas. https://es.slideshare.net/carolinacastano1044/compuertas-logicas-73931324

colaboradores de Wikipedia. (2019, 15 julio). *Modelo de computación*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Modelo_de_computaci%C3%B3n

colaboradores de Wikipedia. (2020, 1 diciembre). *Computación concurrente*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Computaci%C3%B3n_concurrente

colaboradores de Wikipedia. (2021, 13 junio). *Teoría de la computabilidad*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Teor%C3%ADa_de_la_computabilidad

colaboradores de Wikipedia. (2019, 21 octubre). *Teoría de circuitos*. Wikipedia, la enciclopedia libre. https://es.wikipedia.org/wiki/Teor%C3%ADa_de_circuitos