INGENIERÍA EN TECNOLOGÍAS DE INFORMACIÓN E INGENIERIA INDUSTRIAL

ASIGNATURA:

**SISTEMAS DIGITALES**

|  |  |  |  |
| --- | --- | --- | --- |
| NOMBRE DEL TUTOR | Fausto Noé Jiménez | CUATRIMESTRE | Mayo – Agosto 2025 |
| CONTENIDO PROGRAMÁTICO | 1.3. Algebra booleana.  1.4. Compuertas lógicas.  1.5. Familias lógicas de circuitos integrados | SEMANA DEL CURSO | 02 |
| NOMBRE DE ACTIVIDAD | Actividad 2. Implementación de compuertas lógicas. | | |
| NOMBRE DEL ALUMNO |  | | |

|  |
| --- |
| OBJETIVO |
| * **Implementar las compuertas lógicas en un protoboard** para **demostrar su funcionamiento** con base en los **fundamentos de los sistemas digitales.** |

|  |
| --- |
| MATERIAL Y/O EQUIPO NECESARIO |
| Componentes  **1 CI 7404 (NOT)**  **1 CI 7408 (AND)**  **1 CI 7432 (OR)**  **1 conjunto de interruptores tipo DIP**  **3 LED**  **1 Protoboard**  Instrumentos de medición, Multímetro digital  **Fuente de poder regulada de 0-5V**  **Cables**  **Resistencias de 1KOhm** |

|  |
| --- |
| INTRODUCCIÓN |
| En esta práctica usted combinará las compuertas lógicas básicas para formar circuitos lógicos, estudiará su comportamiento, y aprenderá a identificar y resolver problemas comunes en estos circuitos. Estas habilidades serán utilizadas en las siguientes prácticas, donde se utilizarán circuitos lógicos más complejos, llamados bloques combinacionales. Los bloques combinacionales se encuentran comúnmente encapsulados en circuitos integrados, y forman parte fundamental de los sistemas digitales.  **Una tabla de verdad, o tabla de valores de verdades, es una tabla que muestra el valor de verdad de una proposición compuesta, para cada combinación de verdad que se pueda asignar.​**  **En otras palabras, la tabla de verdad nos dice en qué situaciones el enunciado es verdadero y en cuales es falso.** |

|  |
| --- |
| DESARROLLO DE LA PRÁCTICA |
| **Objetivo**: Desarrollar la capacidad de analizar, diseñar e implementar circuitos lógicos combinacionales mediante la aplicación del álgebra booleana, la utilización de compuertas lógicas básicas y el conocimiento de las características de las principales familias lógicas de circuitos integrados, empleando herramientas de simulación como Tinkercad para validar su funcionamiento.  **INVESTIGACIÓN**  **ALGEBRA BOOLEANA:** El álgebra booleana es una rama de las matemáticas que se centra en operaciones lógicas sobre variables que toman valores binarios: 1 (verdadero) y 0 (falso). Fue introducida por el matemático inglés George Boole en el siglo XIX como una forma de representar y analizar proposiciones lógicas mediante métodos algebraicos. (Whitesitt, 1971)  **I.I Preguntas de investigación sobre Álgebra Booleana**   1. ¿Qué es el álgebra booleana y cuál es su origen histórico? 2. ¿Qué valores pueden tomar las variables booleanas y cómo se representan? 3. ¿En qué consiste la operación lógica AND y cuál es su tabla de verdad? 4. ¿En qué consiste la operación lógica OR y cuál es su tabla de verdad? 5. ¿Cómo funciona la operación lógica NOT y qué representa en una tabla de verdad? 6. ¿Cómo se construye una tabla de verdad para una función lógica compuesta? 7. ¿Cuáles son las leyes fundamentales del álgebra booleana y cómo se aplican?    * Ley de identidad    * Ley de anulación    * Ley de idempotencia    * Ley conmutativa, asociativa y distributiva    * Ley de absorción    * Ley de involución    * Leyes de De Morgan 8. ¿Cómo se pueden simplificar expresiones booleanas utilizando las leyes algebraicas? 9. ¿Qué son las funciones booleanas y cómo se expresan simbólicamente? 10. ¿Cómo se aplican los principios del álgebra booleana en el diseño de circuitos lógicos digitales? 11. ¿Qué ventajas tiene simplificar una función lógica antes de implementarla con compuertas? 12. ¿Qué herramientas gráficas (como mapas de Karnaugh) se pueden utilizar para simplificar funciones booleanas? *(opcional según nivel)*   **II Investigación sobre COMPUERTAS LÓGICAS**  **1. Compuerta NOT**   * **Definición y función lógica**: Descripción de cómo la compuerta NOT invierte el estado lógico de su entrada. *| Describir |* * **Símbolo y tabla de verdad**: Representación gráfica y tabla que muestra las posibles entradas y salidas. *| Describir |* * **Circuito integrado (CI) común**: Identificación de modelos como el 74LS04 que contienen compuertas NOT. *| Describir |* * **Aplicaciones típicas**: Ejemplos de uso en circuitos digitales. *| Describir |*   **2. Compuerta AND**   * **Definición y función lógica**: Explicación de cómo la compuerta AND produce una salida alta solo cuando todas sus entradas son altas. *| Describir |* * **Símbolo y tabla de verdad**: Representación gráfica y tabla correspondiente. *| Describir |* * **Circuito integrado (CI) común**: Modelos como el 74LS08 que contienen compuertas AND. *| Describir |* * **Aplicaciones típicas**: Usos comunes en sistemas de control y lógica digital. *| Describir |* * **Diagrama compuerta:** *| Dibujo técnico, símbolo |* * **Diagrama conjunto comercial de compuertas:** *| Diagrama técnico |*   **3. Compuerta OR**   * **Definición y función lógica**: Descripción de cómo la compuerta OR produce una salida alta si al menos una entrada es alta. *| Describir |* * **Símbolo y tabla de verdad**: Representación gráfica y tabla correspondiente. *| Describir |* * **Circuito integrado (CI) común**: Modelos como el 74LS32 que contienen compuertas OR. *| Describir |* * **Aplicaciones típicas**: Ejemplos de uso en sistemas de decisión y lógica combinacional. *| Describir |* * **Diagrama compuerta:** *| Dibujo técnico, símbolo |* * **Diagrama** **conjunto comercial de compuertas:** *| Diagrama técnico |*   **4. Interruptores DIP**   * **Definición y función**: Explicación de qué son los interruptores DIP y cómo se utilizan para configurar circuitos. *| Describir |* * **Características técnicas**: Número de posiciones, corriente y voltaje máximos, dimensiones, etc. *| Describir |* * **Aplicaciones típicas**: Usos en configuración de dispositivos y selección de modos de operación. *| Describir |* * **Diagrama compuerta:** *| Dibujo técnico, símbolo |* * **Diagrama** **conjunto comercial de compuertas:** *| Diagrama técnico |*   **5. Resistencias en circuitos con compuertas lógicas**   * **Función en circuitos digitales**: Descripción de cómo las resistencias se utilizan para limitar corriente y establecer niveles lógicos. *| Describir |* * **Cálculo de valores**: Métodos para determinar el valor adecuado de resistencia según la aplicación. *| Describir |* * **Aplicaciones típicas**: Ejemplos de uso en protección de componentes y establecimiento de niveles lógicos. *| Describir |* * **Diagrama:** *| Dibujo técnico, símbolo |*   **6. Implementación en Tinkercad**   * **Plataforma**: Breve descripción de Tinkercad como herramienta de simulación de circuitos. * **Pasos para la implementación**:   1. Crear una cuenta en Tinkercad.   2. Iniciar un nuevo proyecto de circuito.   3. Agregar los componentes necesarios: compuertas lógicas, interruptores DIP, resistencias, etc.   4. Conectar los componentes según el diseño del circuito combinacional.   5. Simular el circuito y verificar su funcionamiento. |

|  |
| --- |
| PROCEDIMIENTO DE SOLUCIÓN Y COMENTARIOS |
| //El alumno describe paso a paso la realización de la práctica, además de incluir comentarios generales  **PUESTA EN PRÁCTICA**   1. Realizar el siguiente circuito en la plataforma de Tinkercad, en el classroom y espacio asignado en: *https://www.tinkercad.com/classrooms/e1v5DJPj4qA/activities*   **Dispositivos necesarios:**  ***Dispositivos electrónicos***   1. 1 CI 7404 (NOT) 2. 1 CI 7408 (AND) 3. 1 CI 7432 (OR) 4. 1 Conjunto de interruptores tipo DIP 5. 6 LED, de cualquier color 6. 1 Tablilla Protoboard   ***Instrumentos de medición***   1. Multímetro digital 2. Fuente de poder regulada de 0 - 5V 3. Cable telefónico, Alambre para protoboard (No cable dupont) 4. Imagen del circuito propuesto:     **Organiza los colores del cable** a utilizar para la lectura del circuito, la imagen del circuito se adjunta para revisar conexiones, pero debes organizar los cables con los colores ***de acuerdo a las convenciones***.  **✅ Convenciones comunes de colores de cables en protoboard**  ***Color Uso común recomendado***  🔴 Rojo Voltaje positivo (Vcc, +5V, +3.3V, etc.)  ⚫ Negro Tierra (GND)  🔵 Azul Salida lógica o señales digitales  🟢 Verde Entrada lógica o señales de control  🟡 Amarillo Señales intermedias entre compuertas  🟣 Morado Señales auxiliares o salidas hacia LED, buzzer  🟠 Naranja Conexiones a sensores o botones  **✅ Recomendación de agrupación de colores por tipo de función**   | **Función** | **Color sugerido** | | --- | --- | | Voltaje positivo (+5V) | 🔴 Rojo | | Tierra (GND) | ⚫ Negro | | Entrada desde interruptores (DIP) | 🟢 Verde | | Señales entre compuertas | 🔵 Azul / 🟣 Morado / 🟡 Amarillo | | Salida a LED | 🟠 Naranja / 🟡 Amarillo |   **Sugerencias:**   * 1. Salida intermedia hacia entrada de segunda compuerta   **Color sugerido:** Amarillo o Azul  **Justificación:** 🔁 Usar colores distintos para diferenciar señales   * 1. Cable de señal, pero mismo color que GND   **Color sugerido:** ❌ Mejor usar **Azul** o **Naranja**  **Justificación:** Evitar confusión con tierra   * 1. Salida lógica a 74HC32   **Color sugerido:** Amarillo  **Justificación:** 💡 Conviene separar color de entradas y salidas  ***Fundamento de la convención***   **Horowitz, P., & Hill, W. (2015). *The Art of Electronics* (3rd ed.). Cambridge University Press.**   * En esta obra clásica se sugieren convenciones de colores en protoboard para facilitar el diseño, depuración y documentación de circuitos. * **Rerefencia APA**: Horowitz, P., & Hill, W. (2015). *The Art of Electronics* (3rd ed.). Cambridge University Press.    **SparkFun Electronics – Wiring Color Codes**   * SparkFun, una empresa reconocida en electrónica educativa, recomienda el uso de colores específicos para alimentación (rojo), tierra (negro), señales (verde, azul, amarillo, etc.).   🔗 https://learn.sparkfun.com/tutorials/wire-gauge-and-stranding   * **Rerefencia APA**:   SparkFun. (s.f.). *Wire Gauge and Stranding*. Recuperado de https://learn.sparkfun.com/tutorials/wire-gauge-and-stranding   **Adafruit – Best Practices for Breadboarding**   * Recomendaciones educativas para mantener orden y seguridad visual en protoboards.   🔗 https://learn.adafruit.com/adafruit-arduino-lesson-0-getting-started/breadboard-layout   * **Rerefencia APA**: Adafruit. (s.f.). *Best Practices for Breadboarding*. Recuperado de https://learn.adafruit.com/adafruit-arduino-lesson-0-getting-started/breadboard-layout    **Normas industriales (uso profesional)**   * **IEC 60204-1**: Equipos eléctricos industriales – codificación de conductores. * **NEC (NFPA 70)** – National Electrical Code (USA), que establece códigos de color en AC y DC para instalaciones.   Estas no son obligatorias para protoboard, pero inspiran las buenas prácticas académicas.  **✅ Lista de Verificación para Circuito Lógico en Tinkercad**  Completa esta lista antes de entregar tu circuito. Marca cada ítem solo si estás completamente seguro de que lo has cumplido.  **✔ Ítem de verificación**  ☐ He utilizado una fuente de alimentación de 5V (no batería de 9V).  ☐ He coloreado los cables siguiendo las convenciones recomendadas.  ☐ He identificado correctamente cada compuerta lógica (ej. 74HC08 = AND, 74HC32 = OR, 74HC04 = NOT).  ☐ He conectado una resistencia de pulldown en cada salida del interruptor DIP hacia la línea negativa.  ☐ Cada LED tiene una resistencia en serie para limitar la corriente.  ☐ La salida del LED está conectada hacia la línea de alimentación positiva, no directamente al Vcc.  ☐ He utilizado puentes de alimentación entre líneas positivas y negativas superior e inferior del protoboard.  ☐ La fuente de alimentación está conectada solo a las líneas de alimentación del protoboard.  ☐ El DIP switch está orientado con ON hacia arriba y OFF hacia abajo.  ☐ He identificado correctamente las patas del LED: ánodo = más larga o doblada / cátodo = más corta = negativo.   1. Imagen inicial del circuito terminado desarrollado por el alumno. (La imagen se corroborará contra su espacio en TinkerCad):      1. Tablas de verdad:   **📝 Instrucciones:**  Realiza todas las combinaciones posibles (8 en total) de los interruptores A, B y C (000 a 111).  **Para cada combinación:**   1. Simula el circuito en Tinkercad. 2. Observa el estado del LED correspondiente. 3. Registra si el LED está ENCENDIDO o APAGADO. 4. Indica el nivel lógico de salida (1 si está encendido, 0 si está apagado). 5. Toma una captura de pantalla clara de cada simulación y colócala como evidencia. 6. Colorea el LED en el simulador para facilitar la visibilidad en el circuito.   📋 Tabla 1: **COMPUERTA NOT** Monitoreo de la salida del circuito (llenar 8 combinaciones).   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **COMPORTAMIENTO DE LA COMPUERTA NOT Y RESPECTIVO LED**  **(Procurar colorear el LED de tal forma que impacte en la visibilidad en el circuito)** | **Tabla: Monitoreo de la salida del circuito** | | | | | | **DIP** | | |  |  | | **A**  **0/1** | **B**  **0/1** | **C**  **0/1** | **ESTADO DEL LED**  (ENCENDIDO / APAGADO) | **NIVEL LÓGICO DE SALIDA** (0/1) | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 1 |  |  |   📋 Tabla 2: **COMPUERTA AND** Monitoreo de la salida del circuito (llenar 8 combinaciones).   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **COMPORTAMIENTO DE LA COMPUERTA AND Y RESPECTIVO LED**  **(Procurar colorear el LED de tal forma que impacte en la visibilidad en el circuito)** | **Tabla: Monitoreo de la salida del circuito** | | | | | | **DIP** | | |  |  | | **A**  **0/1** | **B**  **0/1** | **C**  **0/1** | **ESTADO DEL LED**  (ENCENDIDO / APAGADO) | **NIVEL LÓGICO DE SALIDA**  **0/1** | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 1 |  |  | |  |  |  |  |  |  |   📋 Tabla 3: **COMPUERTA OR** Monitoreo de la salida del circuito (llenar 8 combinaciones).   |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | | **COMPORTAMIENTO DE LA COMPUERTA OR Y RESPECTIVO LED**  **(Procurar colorear el LED de tal forma que impacte en la visibilidad en el circuito)** | **Tabla: Monitoreo de la salida del circuito** | | | | | | **DIP** | | |  |  | | **A**  **0/1** | **B**  **0/1** | **C**  **0/1** | **ESTADO DEL LED**  (ENCENDIDO / APAGADO) | **NIVEL LÓGICO DE SALIDA**  **0/1** | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 0 | 1 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 0 | 1 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 0 |  |  | | Imagen de la combinación en simulador TinkecCad | 1 | 1 | 1 |  |  |   **CONCLUSIONES**  **EVALUACIÓN**  RUBRO 1 / Entrega de la práctica en evidencia fotográfica y documento PDF: Entrega todas las evidencias, incluye los pasos más importantes de la práctica en documento PDF con evidencias fotográficas que demuestran la comprobación de la tabla de verdad para cada valor.    RUBRO 2 / Bibliografía: Incluye las dos fuentes bibliográficas y en formato APA 7.  RUBRO 3 / Materiales y métodos: Utilizó los componentes o materiales clave en el desarrollo de la práctica. Describe el método utilizado.  RUBRO 4 / Conclusiones: Las conclusiones son fundamentales, claras y concisas, cumplen con la media cuartilla.  RUBRO 5 / Presentación. Orden y visibilidad: Hay orden en la entrega. Las imágenes se distinguen correctamente.  **Fuentes Bibliográficas (Formato APA7)**  Fuente 1: Whitesitt, J. E. (1971). *Álgebra booleana y sus aplicaciones*. Compañía Editorial Continental.  Fuente 2:  Fuente 3:  Fuente 4:  Fuente 5:  Puede incluir tantas fuentes como haya consultado.  **COMPLEMENTO:**   1. **Introducción a Aurduino Uno**    1. Revisar **grabación** 1 de 10 [01 Introducción a Arduino.webm](https://sabesedu.sharepoint.com/:v:/s/LaboratorioArduino-SistemasDigitales/EerHYCpHp0dJm1bpqlLPT1gBt5YqUTthbC5XS78RT5pFSQ?e=92BQrm)   url: <https://sabesedu.sharepoint.com/:v:/s/LaboratorioArduino-SistemasDigitales/EerHYCpHp0dJm1bpqlLPT1gBt5YqUTthbC5XS78RT5pFSQ?e=92BQrm>   * 1. Revisar **grabación** 2 de 10 [02 Salidas Entradas Analogicas Digitales.webm](https://sabesedu.sharepoint.com/:v:/s/LaboratorioArduino-SistemasDigitales/ET0q8LOmtV1CkD7vg4DcREABeglbxZFCXXp0e2X0M9iETA?e=eEZwni)   url: https://sabesedu.sharepoint.com/:v:/s/LaboratorioArduino-SistemasDigitales/ET0q8LOmtV1CkD7vg4DcREABeglbxZFCXXp0e2X0M9iETA?e=eEZwni   1. **ACTIVIDAD LAB1:** Semana 1/12 https://campusuniversidad.sabes.edu.mx/u2023/mod/resource/view.php?id=45258   **Evidencia ACTIVIDADES COMPLEMENTO Lab1:** *https://www.tinkercad.com/classrooms/e1v5DJPj4qA/activities* 🔧 Laboratorio 1: Encendido de Pin **Objetivo:** Encender y apagar el LED integrado de Arduino UNO explicando paso a paso el uso del editor y compilador. Pasos:  1. Ejecutar **Arduino IDE**. 2. Conectar la tarjeta Arduino a la computadora (LED verde “ON” debe encenderse). 3. En el menú Herramientas → Placa, seleccionar “Arduino/Genuino Uno”. 4. En el menú Herramientas → Puerto, seleccionar el puerto COM correspondiente. 5. Borrar el código del editor y escribir lo siguiente:      1. Verificar el código (ícono de palomita). 2. Nombrar el programa “MiPrimerProgramaArduino”. 3. Subirlo y ejecutar. 4. **Imagen del circuito sin simular:** 5. **Imagen del ingresado / Código/Activar** 6. **Imagen del circuito al simular:** 7. **Comprobación de la realización en el espacio asignado en TinkerCad:**   **SI existe/NO existe**  **🔍 Explicación de funciones usadas en Lab 1 – "Encendido de Pin"** 🔹 void setup()  * Es una **función especial** que **se ejecuta una sola vez** al inicio del programa (cuando Arduino se reinicia o se energiza). * Aquí se colocan todas las configuraciones iniciales del programa. * En este caso, se utiliza para definir que el pin 13 será una **salida digital**.  🔹 pinMode(13, OUTPUT);  * Esta instrucción **configura el pin 13 como salida**. * 13 es el número del pin digital (en la placa Arduino Uno, ese pin tiene conectado un LED integrado). * OUTPUT indica que el pin se usará para **enviar voltaje**, no para recibirlo.   void setup() {  pinMode(13, OUTPUT);  } 🔹 void loop()  * Esta función **se repite infinitamente** después de que termina setup(). * Aquí va el código que **se ejecuta continuamente** mientras Arduino esté encendido.   void loop() {  digitalWrite(13, HIGH);  delay(1000);  digitalWrite(13, LOW);  delay(1000);  } 🔹 digitalWrite(13, HIGH);  * Envia una **señal de 5V** al pin 13 (estado **ALTO** o 1). * En este caso, enciende el LED conectado al pin 13.  🔹 delay(1000);  * Pausa el programa durante **1000 milisegundos (1 segundo)**. * Se usa para **crear un retardo** antes de la siguiente instrucción. * Es útil para que el LED se mantenga encendido o apagado el tiempo suficiente para ser visible.  🔹 digitalWrite(13, LOW);  * Envía una señal de **0V** al pin 13 (estado **BAJO** o 0). * Esto **apaga el LED**.   **🧠 Resultado final:**  El programa enciende el LED durante un segundo, lo apaga otro segundo, y repite este ciclo infinitamente.   1. **ACTIVIDAD LAB2:** Semana 1/12   https://campusuniversidad.sabes.edu.mx/u2023/mod/resource/view.php?id=45259  **Evidencia ACTIVIDADES COMPLEMENTO Lab2:** *https://www.tinkercad.com/classrooms/e1v5DJPj4qA/activities* 🔧 Laboratorio 2: Entradas y Salidas con Pulsador y LED **Objetivo:** Detectar la presión de un pulsador y encender un LED, usando condicionales y estructura antirebote. Pasos:  1. Ejecutar Arduino IDE. 2. Conectar la tarjeta. 3. Borrar el código del editor y escribir lo siguiente:      1. Verificar el código (palomita). 2. Guardar como “EntradasySalidas”. 3. Subirlo y ejecutar. 4. **Imagen del circuito sin simular:** 5. **Imagen del ingresado / Código/Activar** 6. **Imagen del circuito al simular:** 7. **Comprobación de la realización en el espacio asignado en TinkerCad:**   **SI existe/NO existe**  **🔍 Explicación de funciones usadas en Lab 2 – "Entradas y Salidas con Pulsador y LED"** Declaración de variables  * Se crean dos **variables tipo entero** para representar los pines conectados al pulsador (entrada) y al LED (salida). * PULSADOR = 2 significa que el botón está conectado al pin digital 2. * LED = 3 indica que el LED está conectado al pin digital 3.   int PULSADOR = 2;  int LED = 3; 🔹 setup()  * Se ejecuta una sola vez al iniciar el programa. * Se configuran los **modos de funcionamiento de los pines**:   + PULSADOR como **entrada** (INPUT) para leer su estado.   + LED como **salida** (OUTPUT) para encenderlo o apagarlo.   void setup() {  pinMode(PULSADOR, INPUT);  pinMode(LED, OUTPUT);  }  **🔹 while (digitalRead(PULSADOR) == LOW) {}**   * Esta línea indica: **"espera aquí hasta que se presione el botón"**. * digitalRead(PULSADOR) lee el estado del pin 2. * Si está en LOW (botón no presionado), el programa **queda detenido** dentro del while.   void loop() {  while (digitalRead(PULSADOR) == LOW) {}  **🔹 ESTADO = digitalRead(LED);**   * Se **lee el estado actual del LED** (si está encendido o apagado). * Este valor se guarda en la variable ESTADO. *(Nota: en el código real, ESTADO debería haber sido declarada previamente para evitar error).*   ESTADO = digitalRead(LED); 🔹 digitalWrite(LED, !ESTADO);  * Esta línea **cambia el estado del LED**: si está encendido, lo apaga; si está apagado, lo enciende. * !ESTADO es el operador **NOT lógico**, invierte el valor actual.   digitalWrite(LED, !ESTADO); 🔹 Segundo while  * Evita que el LED se siga alternando si el botón se mantiene presionado. * Espera a que el usuario **suelte el botón** para volver al inicio del loop().   while (digitalRead(PULSADOR) == HIGH) {}  **🧠 ¿Qué hace todo el programa?**   * Espera a que el usuario **presione el pulsador**. * Cuando lo hace, **cambia el estado del LED** (enciende o apaga). * Luego **espera a que el botón se suelte**, evitando múltiples lecturas por un solo toque (esto se llama **antirebote manual**). * Repite este proceso indefinidamente.  1. **ACTIVIDAD LAB3: Semana 2/12**   <https://campusuniversidad.sabes.edu.mx/u2023/mod/resource/view.php?id=45267>  **Evidencia ACTIVIDADES COMPLEMENTO Lab3:** *https://www.tinkercad.com/classrooms/e1v5DJPj4qA/activities* 🔧 Laboratorio 3: Compuerta Lógica AND **Objetivo:** Simular el comportamiento de una compuerta lógica AND usando dos pulsadores. El LED se enciende solo si ambas entradas están en alto. Material:  * 2 pulsadores * 2 resistencias de 1k * 1 resistencia de 220 ohm * 1 LED * Protoboard * Arduino UNO * Cables de conexión  Pasos:  1. Ejecutar Arduino IDE. 2. Conectar la tarjeta. 3. Borrar el código del editor y escribir lo siguiente:  Código:   💡 *Para probar una compuerta OR, simplemente cambia la instrucción if (varx && vary) por if (varx || vary).*   1. Verificar el código. 2. Guardar como “CompuertaAND”. 3. Subirlo y ejecutar. 4. **Imagen del circuito sin simular:** 5. **Imagen del ingresado / Código/Activar** 6. **Imagen del circuito al simular:** 7. **Comprobación de la realización en el espacio asignado en TinkerCad:**   **SI existe/NO existe**  **🔍 Explicación de funciones usadas en Lab 3 – "Compuerta Lógica AND"**  **🔹 Declaración de variables**   * Se declaran **cinco variables enteras**:   + PINX y PINY: indican los pines digitales donde están conectados los dos **pulsadores** (entradas).   + LED: pin donde se conecta el **LED** (salida).   + varx y vary: variables que guardarán el estado leído de cada pulsador (0 o 1).   int PINX = 8;  int PINY = 3;  int LED = 10;  int varx = 0;  int vary = 0;  **🔹 setup()**   * Se ejecuta una sola vez al iniciar el programa. * Define los modos de uso de los pines:   + PINX y PINY como **entradas** para leer el estado de los pulsadores.   + LED como **salida** para encenderlo cuando se cumpla la condición lógica.   void setup() {  pinMode(PINX, INPUT);  pinMode(PINY, INPUT);  pinMode(LED, OUTPUT);  }  **🔹 Lectura de entradas**   * digitalRead(PINX) y digitalRead(PINY) leen el estado de los pulsadores conectados a los pines 8 y 3. * El resultado (0 o 1) se guarda en varx y vary.   void loop() {  varx = digitalRead(PINX);  vary = digitalRead(PINY);  **🔹 Condición lógica AND**   * if (varx && vary) significa: **“Si ambos pulsadores están presionados (valor 1), entonces…”** * Si se cumple la condición:   + El LED se enciende (digitalWrite(LED, HIGH);) * Si **no** se cumple (uno o ambos pulsadores están sueltos):   + El LED se apaga (digitalWrite(LED, LOW);)   if (varx && vary)  digitalWrite(LED, HIGH);  else  digitalWrite(LED, LOW);  }  **🔹 Circuito en TinkerCad / Usando push buttons (Problema en simulador, no se pueden presionar dos botones al mismo tiempo con el ratón)**    **🔹 Circuito en TinkerCad / Despliegue del bloque de escritura de código (en el ejemplo se usa el caso de la compuerta OR)**    **🔹 Circuito en TinkerCad / Alternativa al uso de push buttons, usar DIP**    **🧠 ¿Qué hace el programa?**   * Lee el estado de dos entradas (pulsadores). * Aplica una condición lógica tipo **AND**: solo si ambos pulsadores están presionados al mismo tiempo, el LED se encenderá. * Si alguno o ambos están en estado bajo (no presionados), el LED se mantendrá apagado. * El ciclo se repite constantemente.   **💡 ¿Y si quiero simular una compuerta OR?**  Solo cambia esta línea:  if (varx && vary)  por:  if (varx || vary)  El operador || representa el **OR lógico**, es decir, el LED se encenderá si **al menos uno** de los pulsadores está presionado. |