INSTITUTO TECNOLÓGICO DE COSTA RICA

ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA

MT 4002
Laboratorio de Electrónica Digital

Proyecto 1:
Detección de límite digital

Gabriel Orlando González Rodríguez – 2019057548

Emmanuel Naranjo Blanco – 2019053605

David José Rodríguez Camacho – 2019024546

Profesor: Ing. Ana María Murillo Morgan.

Semestre II - 2021

I. Descripción

El proyecto consiste en realizar la comparación de dos datos digitales de 4 bits para determinar cuál de los dos es mayor o si son iguales. Uno de los datos vendrá de un sensor analógico conectado a un ADC (Analog to Digital Converter), del cual se tomarán los 4 bits menos significativos. El otro dato vendrá de un dip switch, también de 4 bits que ejercerá como el límite digital. Por tanto, se comparará si el valor del sensor es menor que, igual que o mayor que el valor del límite.

Esto se hará utilizando únicamente compuertas lógicas de la familia CMOS. Para la salida se debe encender uno de tres LEDs: rojo (valor mayor que el límite), amarillo (valor igual que el límite) y verde (valor menor que el límite). En la Figura 1 pueden observar el diagrama funcional del comparador.

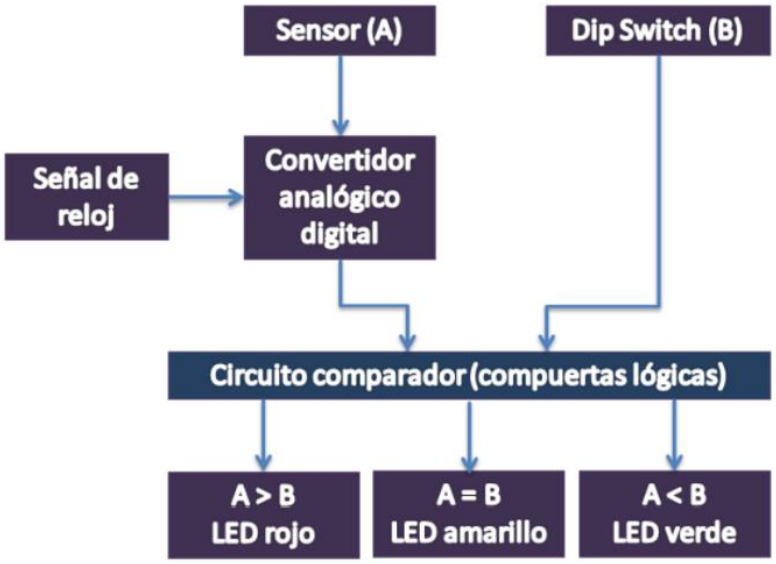


Figura 1. Diagrama funcional del comparador.

II. Diseño

A continuación, se presenta el proceso de diseño realizado para obtener las tres distintas funciones de salida según los distintos valores de entrada, representados por el sensor y el Dip-Switch. En este caso se considera que A es el sensor y B es el switch.

i. Función A = B

En esta sentencia lógica lo que se hace es comparar dos bits (uno del sensor y otro del dip-switch, y que correspondan al mismo nivel con respecto a su cifra significativa) y verificar que sus valores sean iguales. Para que el valor numérico de los 4 bits del sensor sea igual al valor numérico de los 4 bits del switch significa que hay que hacer esta misma comparación entre cada par de bits entre el sensor y el switch.

Como se desea un valor de "true" únicamente cuando ambos bits posean el mismo valor, se tiene la siguiente tabla de verdad, que corresponde a una compuerta XNOR. Para este caso se comparan los bits más significativos de cada componente, nombrados como A3 y B3.

Tabla 1. Tabla de verdad de comparador de dos bits para A3 = B3.

A3	B3	Y
0	0	1
0	1	0
1	0	0
1	1	1

Como la función es una compuerta XNOR, su ecuación se deduce fácilmente y se presenta a continuación.

$$Y = \overline{(A \oplus B)}$$

De este modo, para concluir que los juegos de 4 bits son iguales, todas las comparaciones de cata set de 2 bits tienen que ser verdaderas al mismo tiempo, por eso se usa la compuerta

AND para unir las 4 expresiones XNOR y así, se obtiene la ecuación E, de “equal”. (Ecuación 1).

$$E = \overline{(A3 \oplus B3)} \cdot \overline{(A2 \oplus B2)} \cdot \overline{(A1 \oplus B1)} \cdot \overline{(A0 \oplus B0)}$$

Ecuación 1. Ecuación para comparar la equivalencia de ambos valores.

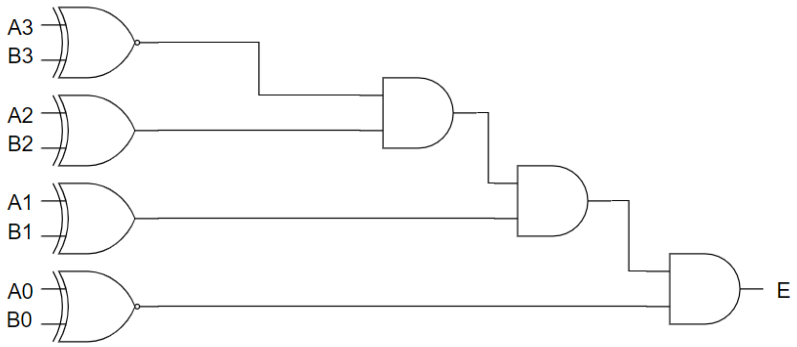


Figura 2. Circuito lógico de la ecuación 1.

ii. Función A>B

Para comparar si un set de n bits contiene un valor numérico mayor o menor que otro se tiene que partir de la comparación entre sus bits más significativos, ya que de esta forma si uno posee un valor de 1 y el otro de 0, se sabe que el del valor de 1 es mayor sin necesidad de comparar los demás bits, y en el caso de que ambos posean el mismo valor no se puede llegar a ninguna conclusión y se tiene que comparar el siguiente par de bits.

Utilizando este análisis para nuestro caso, comparando los bits más significativos, como queremos saber si A3 es mayor que B3, podemos escribir la siguiente tabla de verdad. (Tabla 2).

Tabla 2. Tabla de verdad de comparador de dos bits para A3 > B3.

A3	B3	Y
0	0	0
0	1	0
1	0	1
1	1	0

Se pone un 0 a la salida del caso de que ambas entradas sean iguales porque lo que se quiere es comprobar cuando A3 sea mayor que B3, ya que en este caso se llega directamente a la conclusión de que A es mayor que B. Con el procedimiento de mintérminos se obtiene la expresión $A3\overline{B3}$ para esta tabla de verdad.

$$Y = A3\overline{B3}$$

Cuando la salida es 0 hay dos posibilidades, que A3 sea menor que B3 o que sean iguales, para el primer caso también se podría llegar a una conclusión directamente, pero en el caso en que son iguales no se puede saber y hay que comprar el siguiente par de bits. Para confirmar si son iguales se hace uso de las compuertas XNOR obtenidas en caso anterior (A3 = B3) y se unen a la expresión de la inequidad (a partir del segundo par de bit) con la compuerta AND, ya que todas tienen que ser verdaderas para que esa expresión sea verdadera y poder pasar a la próxima pareja de bits. Cuando se tienen todas estas expresiones, se unen con compuertas OR, ya que si solo una da 0 el resultado final debería ser 0 porque algún par puede no obedecer a la sentencia lógica de A > B; así obteniendo la ecuación G, de "greater". (Ecuación 2).

$$G = A3\overline{B3} + \overline{(A3 \oplus B3)}(A2\overline{B2}) + \overline{(A3 \oplus B3)(A2 \oplus B2)}(A1\overline{B1}) + \overline{(A3 \oplus B3)(A2 \oplus B2)(A1 \oplus B1)}(A0\overline{B0})$$

Ecuación 2. Ecuación para la sentencia A > B.

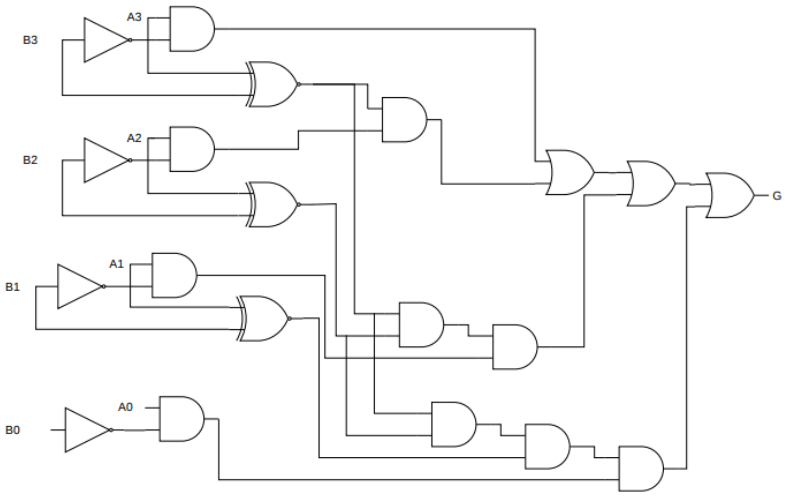


Figura 3. Circuito lógico de la ecuación 2.

iii. Función A< B

Para esta función se utiliza la misma lógica que para el caso en que A > B, solo que como se desea verificar en este caso si B es mayor, se invierte la entrada que va negada a la AND, en otras palabras, se niega A ahora ya que es la que se toma como menor en la tabla de verdad, la cual se presenta a continuación (Tabla 3).

Tabla 3. Tabla de verdad de comparador de dos bits para A3 < B3.

A3	B3	Y
0	0	0
0	1	1
1	0	0
1	1	0

De la cual se deduce la ecuación que representa dicho funcionamiento.

$$Y = \overline{A3}B3$$

De este modo, la ecuación resultante que modela cuando los datos de entrada del sensor son menores a los datos del dip switch se presenta a continuación (ecuación 3) y se modela con la letra L. de “lower”. (Ecuación 3).

$$L = \overline{A3}B3 + \overline{(A3 \oplus B3)}(\overline{A2}B2) + \overline{(A3 \oplus B3)(A2 \oplus B2)}(\overline{A1}B1) + \overline{(A3 \oplus B3)(A2 \oplus B2)(A1 \oplus B1)}(\overline{A0}B0)$$

Ecuación 3. Ecuación para la sentencia A < B.

Por último, el conjunto de compuertas lógicas se muestra en la Figura 4. Cabe destacar que por facilidad de implementación, en este caso no se usó en las simulaciones en TinkerCad ni en el circuito físico, en este se utilizó la operación NOR de las dos ecuaciones anteriores, porque si ambas ecuaciones (las de A = B y A > B) dan 0 eso quiere decir que los valores no cumplían ni con A = B ni A > B por lo tanto, deben obedecer A < B. La ecuación que representa este análisis corresponda a la siguiente. (Ecuación 4).

$$L = \overline{(E + G)}$$

Ecuación 4. Simplificación de la ecuación 3.

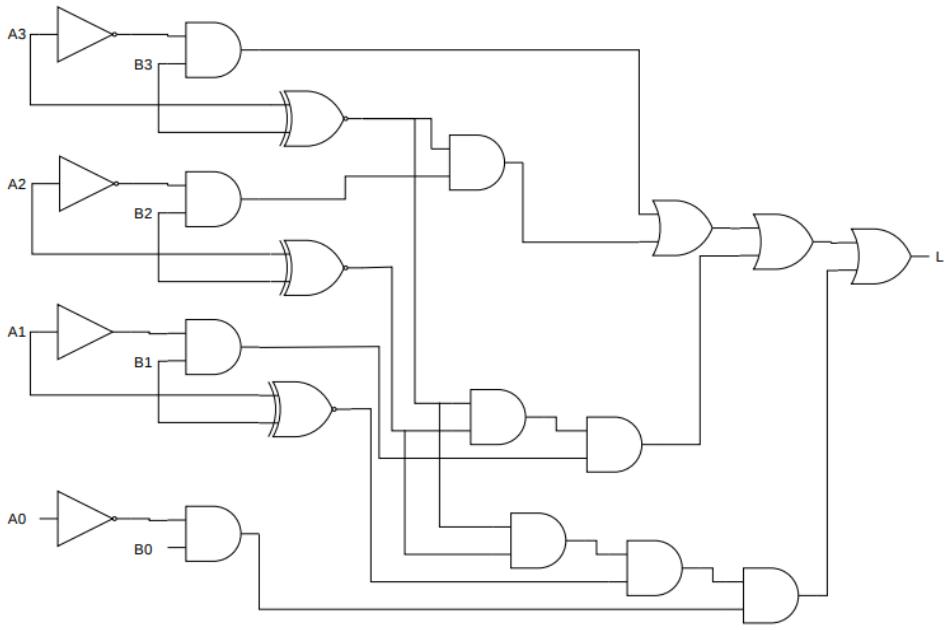


Figura 4. Circuito lógico de la ecuación 3.

iv. ADC y su programación

El ADC consiste en un convertidor analógico a digital que permite la interacción de los circuitos digitales con las variables analógicas del mundo real. Este se encarga de codificar una señal analógica en un número binario. Dicha señal analógica se obtiene a partir de un sensor, sea de temperatura, luz, movimiento, entre otros. En este caso se escogió un sensor de fuerza.

Para el presente proyecto se utilizó el convertidor analógico-digital integrado en el arduino UNO, que consiste una característica sumamente útil ya que permite convertir un voltaje de la lectura de un pin analógico a un valor digital. En este caso se trata de un ADC de 10 bits, cuyos valores varían entre 0 y 1023. Según Seidle[1] (2021), una de las maneras más usadas de lograr esto es cargando un capacitor con el voltaje de la lectura analógica y luego medir el tiempo que dura en descargarse. El microcontrolador monitorea la cantidad de ciclos de reloj que dura el capacitor en descargarse y este número de ciclos es el resultado del ADC.

No obstante, al usar este ADC el problema es que su resolución es de 10 bits, pero se ocupan únicamente 4 bits (valores del 0 al 15), Seidle indica que hay una manera de relacionar el voltaje análogo, el voltaje del sistema, la lectura del ADC y su resolución. Esto se logra con la siguiente ecuación 5.

$$\frac{resolución\ ADC}{voltaje\ del\ sistema} = \frac{lectura\ ADC}{VAC\ medido}$$

Ecuación 5. Ecuación del valor radiométrico del ADC.

Esta ecuación 5 dice que cualquier tensión menor al voltaje del sistema será un valor en la proporción que existe entre la resolución y el voltaje del sistema. Con esto es posible encontrar el voltaje análogo de la salida del sensor con la ecuación 6.

1 Seidle, N. (2021). *Analog to Digital Conversion.* https://learn.sparkfun.com/tutorials/analog-to-digital-conversion/all

$$VAC\ medido\ =\ lectura\ ADC \cdot \frac{voltaje\ del\ sistema}{resolución\ ADC}$$

Ecuación 6. Ecuación para encontrar el voltaje análogo.

Por último, se puede calcular el nuevo valor de la lectura del ADC con la resolución que necesitamos con la siguiente ecuación.

$$lectura\ ADC = VAC\ medido\ \cdot \frac{resolución\ ADC}{voltaje\ del\ sistema}$$

Ecuación 7. Ecuación para encontrar el nuevo valor en la resolución correcta.

Luego, se tiene que convertir este valor encontrado a binario, con el fin de representar estos resultados como las entradas del sistema, para esto se usó el método de la división entre 2, el cual consiste en dividir el valor en sistema decimal entre 2 las veces que sea necesario e ir anotando el residuo de las divisiones de derecha a izquierda y así se obtiene el equivalente en el sistema binario.

Con esta información se obtiene el siguiente código mostrado en las Figuras 5, 6 y 7. En la primera figura se pueden ver las declaraciones de las variables globales del programa que son *valorADC, vACSensor, valor4b* y el *setup*, en esta segunda parte se inicia la comunicación con el monitor serial y se define la función de los pines digitales 2, 3, 4 y 5.

En la Figura 6 se tiene el *loop* del programa, esta parte se cicla desde el inicio del programa y todo lo que está aquí es lo que se va a correr. Aquí primero se obtiene el valor de entrada en el analógico A0 y se guarda en la variable *valorADC*, luego se aplica un *map* para pasar un valor de un rango a otro, estos rangos son los expresados en la Ecuación 5. Este nuevo valor se guarda en la variable *valor4b*, esta corresponde al valor decimal que se va a pasar a binario de 4 bits. Dentro de este map se tiene la variable *valorADC* cuyo valor corresponde al valor que se obtuvo con el ADC integrado al arduino (en 10bits) y se encuentra con la Ecuación 7, luego se imprimen los valores de estas variables para fines de depuración.

```
int valorADC;
float vACSensor;
int valor4b;

void setup()
{
  Serial.begin(9600);
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
}
```

Figura 5. Variables globales y Setup del programa.

En la segunda parte del loop (cuadro anaranjado) se llama la ecuación *BinaryConversion* y se le manda como parámero la varaible *valor4b,* se tiene un delay de 2 segundos y se apagan todos los LEDs, ya que el programa se va a refrescar cada 2 segundos y si no se apagan todos los LEDs en cada vuelta, nunca se actualizaría el valor que representan.

```
void loop()
{
  valorADC = analogRead(A0);
  valor4b = map(valorADC, 0, 890, 0, 15);
  vACSensor = valorADC*(5.0/1023);

  Serial.println("Binario 4 bits");
  Serial.print("Decimal 10b: ");
  Serial.println(valorADC);
  Serial.print("Decimal 4b: ");
  Serial.println(valor4b);
  Serial.print("Tension: ");
  Serial.println(vACSensor);
  Serial.println("");

  BinaryConversion(valor4b);
  delay(2000);

  digitalWrite(2, LOW);
  digitalWrite(3, LOW);
  digitalWrite(4, LOW);
  digitalWrite(5, LOW);
}
```

Figura 6. Loop del programa.

En la función *BinaryConversion* obtiene el valor en binario y se encienden los LEDs para saber con qué valor se está trabajando, además ocupa como parámetro un valor entero *num para convertir a binario*. Primero se crea un arreglo *Result* de tamaño 4 para guardar los bits del número en binario, luego se inicializa el contador z en dos, porque este es el número más pequeño de la salida digital que se va a utilizar. El primer *for* recorre el arreglo desde la posición 0 a la 3, va dividiendo el valor *num* entre dos, y guardando los residuos de la división de *num* entre dos y divide el *num* entre dos. Esto se repite 4 veces para obtener el número en binario, cuyo bit más significativo quedará en la posición 0 del arreglo *Result*. El segundo *for* recorre el arreglo de la misma manera que el anterior, pero compara si el valor guardado en esa posición es igual a 1, si esto es verdadero va a hacer que el pin digital de valor z genere una salida ALTO, afuera de este condicional se aumenta el contador z en uno para mover el contador al siguiente pin digital; esto también se repite 4 veces.

```
int BinaryConversion(int num){
  int Result[4] = {};
  int z = 2;
  for (int i = 0; i < 4 ; i++){
  Result[i] = num % 2;
  num = num / 2;
  }

  for(int j = 0; j < 4; j++){
    if (Result[j] == 1){
      digitalWrite(z, HIGH);
    }
    z = z+1;
  }
}
```

Figura 7. Función BinaryConversion.

III. Implementación

A partir de las tres funciones obtenidas anteriormente, se integraron los circuitos para obtener el resultado final del circuito comparador. En esta sección se presenta los diagramas finales. Por un lado, el diagrama indicando las compuertas lógicas correspondientes se muestra en la Figura 9, por otro, el circuito resultante en físico se muestra en la Figura 10. Para esto se utilizan compuertas de la familia CMOS y se armó en la plataforma Tinkercad.

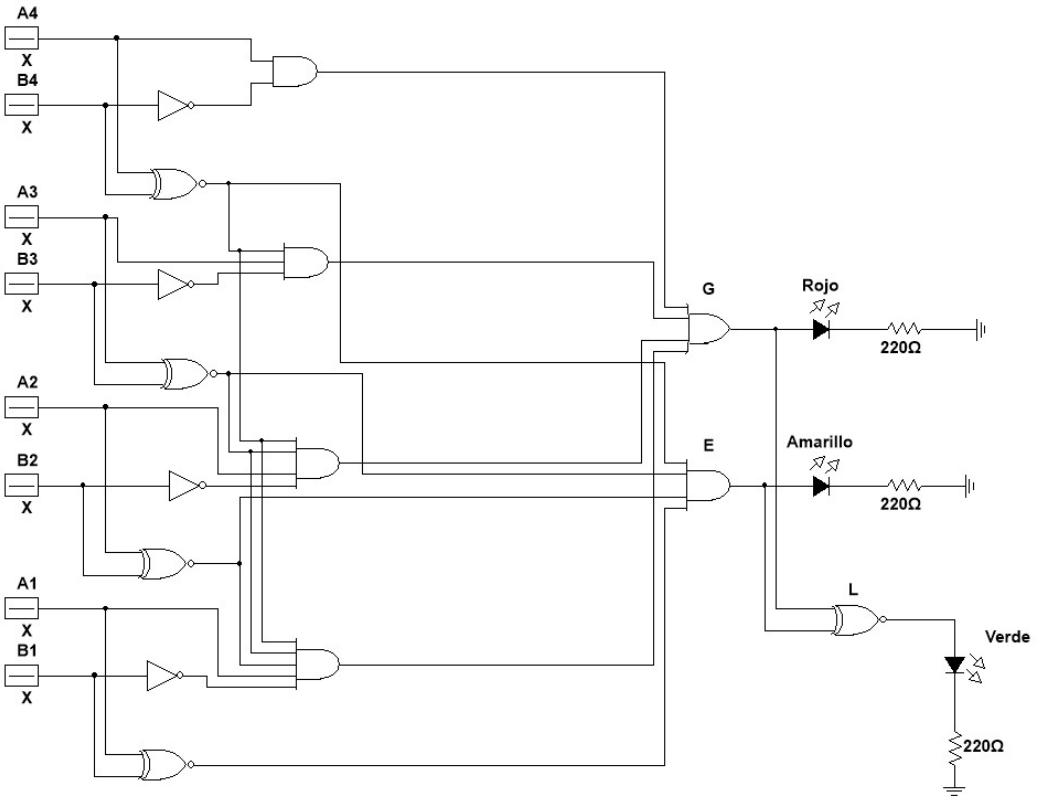
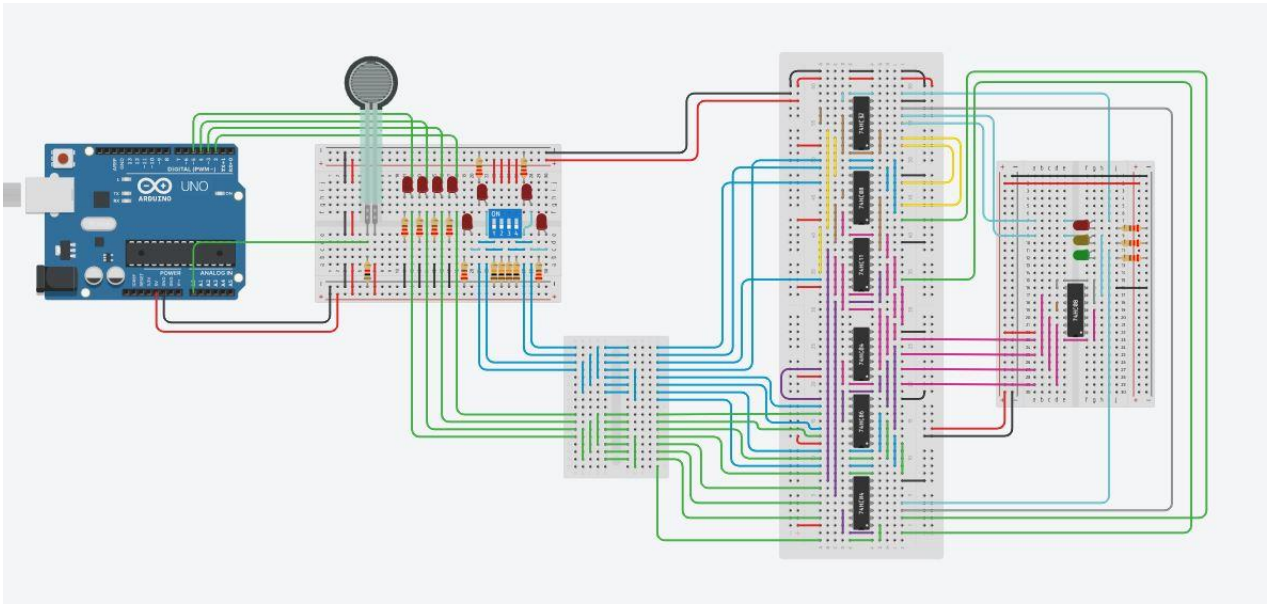


Figura 9. Circuito comparador de 4 bits.



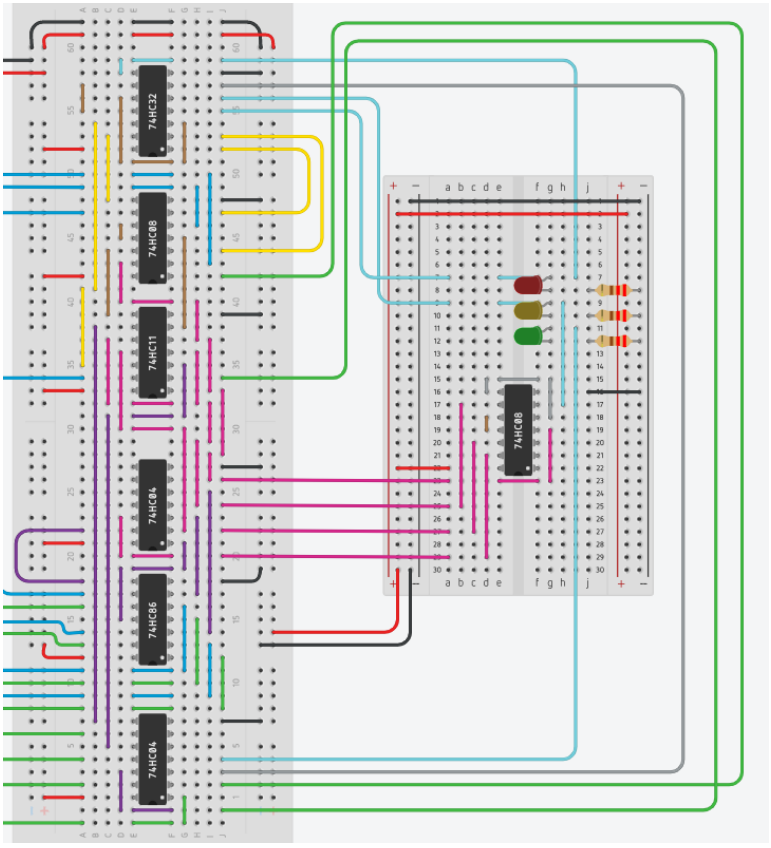
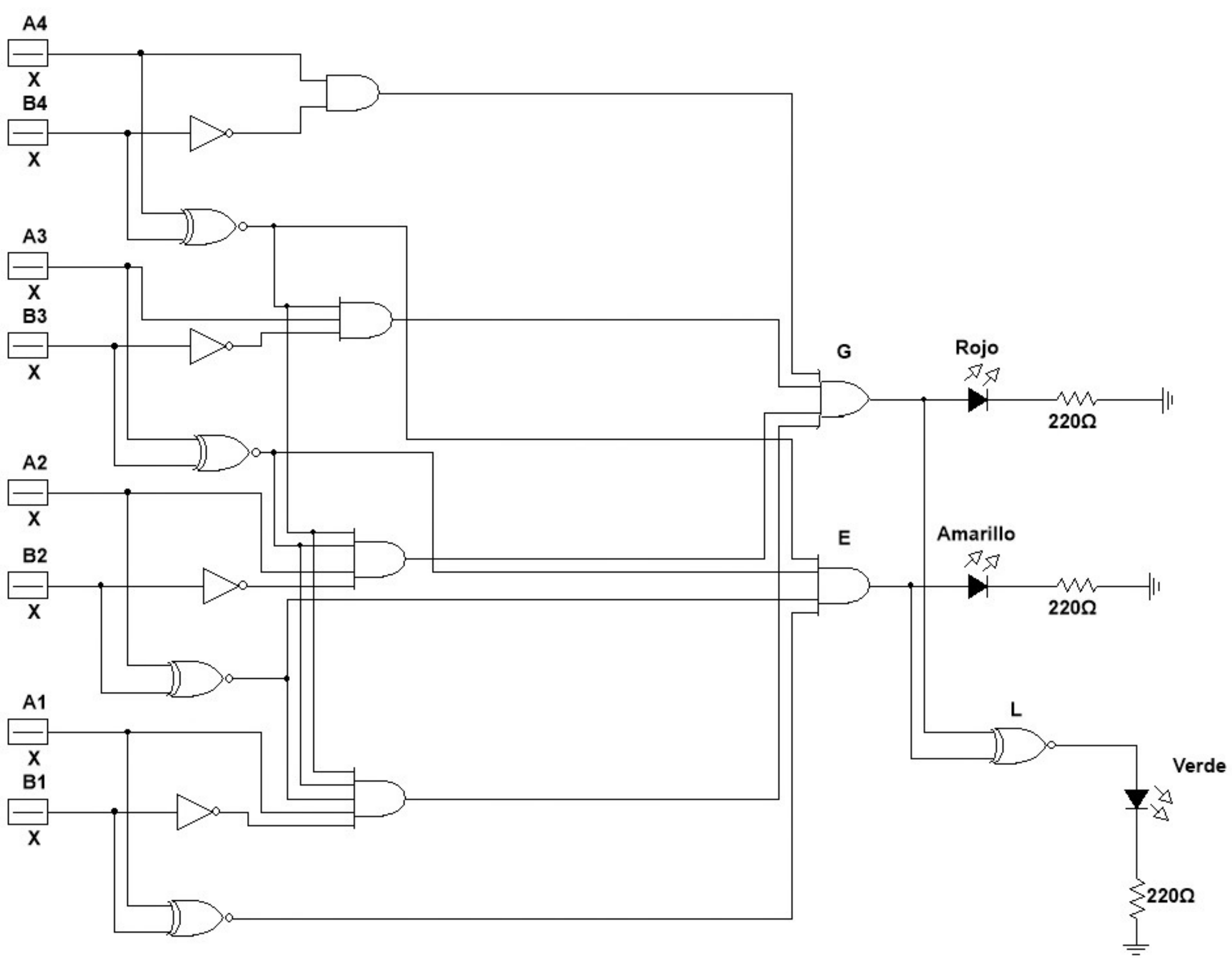


Figura 10. Circuito comparador de 4 bits armado en Tinkercad.

Diagrama del circuito

En esta sección se presenta el diagrama del circuito en formato A3.



TEC | INSTITUTO TECNOLÓGICO DE COSTA RICA ÁREA ACADÉMICA DE INGENIERÍA MECATRÓNICA	LABORATORIO DE ELECTRÓNICA DIGITAL	
	DAVID RODRÍGUEZ	2019024546
	EMMANUEL NARANJO	2019053605
	GABRIEL GONZÁLEZ	2019057548
PROYECTO 1: DETECCIÓN DE LÍMITE DIGITAL	LÁMINA	1 / 1

CREADO CON UNA VERSIÓN PARA ESTUDIANTES DE AUTODESK