Universidad de San Carlos de Guatemala

Facultad de Ingenieria

Escuela de Ciencias y Sistemas

Organización Computacional

Sección B

Ingeniero Otto Leiva

Aux Christian Real

22 de Noviembre de 2019, Guatemala

# Viborita Serial

| No. | Carnet | Nombre | Ponderación |
|---|---|---|---|
| 1 | 201700308 | Adrián Byron Ernesto Alvarado Alfaro | 100 |
| 2 | 201700317 | Carlos Alejandro Tenes Mejía | 100 |
| 3 | 201700521 | Ludwing Gabriel Paz Hernández | 100 |
| 4 | 201700733 | Byron Antonio Orellana Alburez | 100 |
| 5 | 201709166 | Jackeline Alexandra Benitez Benitez | 100 |

# Introducción

A lo largo de los años la tecnología ha ido avanzando a pasos agigantados, hemos llegado a tal punto que todos los dispositivos que están siendo creados tienen una tendencia a ser cada vez más pequeños con cada año que pasa. Un circuito lógico secuencial es aquel cuyas salidas no solo dependen de sus entradas actuales, sino también de su posición o estado actual, almacenada en elementos de memoria.

Es importante conocer las funcionalidades de un nuevo programa, sobre todo si no conocemos nada de el. En este caso este programa es destinado para jugadores de Snake, el famoso juego de la culebrita en donde mientras la culebra come, esta incrementa su tamaño. La diferencia de este snake con los convencionales es que puede jugarse desde la computadora, y además mostrarse en una matriz de LED.

Para esto, se utiliza un puerto paralelo simulando un puerto serial. Un puerto paralelo es una interfaz entre un computador y un periférico, cuya principal característica es que los bits de datos viajan juntos, enviando un paquete de byte a la vez. Es decir, se implementa un cable o una vía física para cada bit de datos formando un bus. Con los cual se puede realizar la comunicación serial, La comunicación serie o comunicación secuencial, en telecomunicaciones e informática, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

## Descripcion del Problema:

La Empresa "ACTIVISION SPORTS" ha un nuevo videojuego basado en el clásico juego "Snake", se requiere que el videojuego sea diferente de una forma no convencional. Por lo que se solicita una demostración del funcionamiento del prototipo de dicho videojuego que deberá cubrir los siguientes requerimientos:

Elaboración de una matriz de LED (12 x 12) que será el monitor del videojuego dicha matriz que se deberá conectar con un puerto (interfaz) de la PC para mostrar el juego de la serpiente y controlarlo mediante un control que será la entrada de movimiento de la serpiente.

La empresa decidió que para poder vender dicho videojuego es necesario que este incorpore un control para el movimiento de la serpiente de manera independiente. Por lo que es necesario que el control posea las teclas esenciales de arriba, abajo, izquierda,

derecha y pausa.

La aplicación deberá poseer look and feel y deberá contar con un Log in, Opcion de registrar, sesión administrador, con respecto a la serpiente se moverá automáticamente en la dirección que apunte su cabeza y se cambiará su dirección con las flechas del teclado teniendo los 4 movimientos básicos, arriba ( ↑ ), abajo ( ↓ ), izquierda ( ← ) y derecha ( → ); cuando se vaya encontrando con una fruta o bolita, la serpiente crecerá y esto generará un puntaje definido por el estudiante. El juego terminará si es que se acaban los tres niveles y se gana, si la serpiente se topa con algún obstáculo o al momento de toparse con sigo misma; en cualquiera de los casos se guardará toda la información necesaria para los reportes y se regresará al menú principal.

## Lógica del Sistema:

Log in:

En la interfaz del inicio de sesión, se ingresa el nombre de usuario y la contraseña, estas previamente deben ser registrados en la parte de "Registrar usuario" en donde se debe poner los nombres de usuario, la contraseña y su respectiva confirmación contraseña. Al momento de inicio de sesión se puede ingresar con el usuario y el password y nos deplaza al juego.

```
   if (txtUser.Text==""||txtContra.Text=="")
            {
                MessageBox.Show("Hay campos vacios");
                limpiar();
            }
            else {
                if (Manejador.getInstancia().login(txtUser.Text, txtContra.Text))
                {
                    Usuario user = Manejador.getInstancia().buscarUsuario(txtUser.Text);
                    MessageBox.Show("Bienvenido " + user.getUserName());
                    Manejador.getInstancia().setUsuario(user);
                    Informacion informacion = new Informacion();
                    //Juego informacion = new Juego();
                    this.Hide();
                    limpiar();
                    informacion.Show();
                }
                else if (txtUser.Text == "Admin_ORGA" && txtContra.Text == "12345")
                {
                    MessageBox.Show("Bienvenido Admin_ORGA");
                    Administrador administrador = new Administrador();
                    limpiar();
                    administrador.Show();
                    this.Hide();
                }
                else
                {
                    limpiar();
                    MessageBox.Show("Datos Erroneos");
                }
            }

public void limpiar() {
            txtUser.Text = "";
            txtContra.Text = "";
        }
```

Reporteria:

-Top tiempo:

Muestra el top 10 con los mejores tiempos que se batieron a lo largo e las partidas, como mínimo debe tener 10 usuarios registrados que hayan jugado por lo menos una vez.

Para este se utiliza la función "Order By Descending" la cual ordena de mayor a menor el punteo de los 10 mejores jugadores.

```
public TopTiempo()
        {
            InitializeComponent();
            IEnumerable<Usuario> top =
Manejador.getInstancia().getUsuarios().OrderByDescending(usuario => usuario.getTiempo());
            dataGridView1.Columns.Add("usuario", "Usuario");
            dataGridView1.Columns.Add("tiempo", "Tiempo");
            dataGridView1.Columns.Add("punteo", "Punteo");
            for (int i = 0; i < 10; i++)
            {
                dataGridView1.Rows.Add(top.ElementAt(i).getUserName(),
top.ElementAt(i).getTiempo(), top.ElementAt(i).getPuntos());
            }
        }
```

-Top Puntos:

Muestra el top 10 con los mejores punteos que se batieron a lo largo de las partidas, como mínimo debe tener 10 usuarios registrados que hayan jugado por lo menos una vez.

Para este se utiliza la función "Order By Descending" la cual ordena de mayor a menor el punteo de los 10 mejores jugadores.

```
private void TopPuntos_Load(object sender, EventArgs e)
        {
            IEnumerable<Usuario> top =
Manejador.getInstancia().getUsuarios().OrderByDescending(usuario => usuario.getPunto());
            dataGridView1.Columns.Add("usuario", "Usuario");
            dataGridView1.Columns.Add("tiempo", "Tiempo");
            dataGridView1.Columns.Add("punteo", "Punteo");
            for (int i = 0; i < 10; i++)
            {
                dataGridView1.Rows.Add(top.ElementAt(i).getUserName(),
top.ElementAt(i).getTiempo(), top.ElementAt(i).getPuntos());
            }
        }
```

-Ver Usuario:

Con esta función se muestran todos usuario registrados en la aplicación en un Data Grid View

```
private void VerUsuario_Load(object sender, EventArgs e)
        {
            dataGridView1.Columns.Add("Usuario", "Usuario");
            dataGridView1.Columns.Add("Contrasenia", "Contraseña");
            dataGridView1.Columns.Add("tiempo", "Tiempo");
            dataGridView1.Columns.Add("punteo", "Punteo");
            dataGridView1.Columns.Add("nivel", "Nivel");
            foreach (var item in Manejador.getInstancia().getUsuarios())
            {

dataGridView1.Rows.Add(item.getUserName(),item.getContra(),item.getTiempo(),item.getPuntos(),item.
getNivel());
            }
        }
```

-Eliminar Usuario:

Con esta función se muestran todos usuario registrados en la aplicación en un Data Grid View y al seleccionar un usuario en especifico se elimina.

```
public partial class Eliminar : Form
    {
        public Eliminar()
        {
            InitializeComponent();
            dataGridView1.Columns.Add("Usuario", "Usuario");
            dataGridView1.Columns.Add("Contrasenia", "Contraseña");
            dataGridView1.Columns.Add("Eliminar", "Eliminar");
            foreach (var item in Manejador.getInstancia().getUsuarios())
            {
                RadioButton radio = new RadioButton();
                dataGridView1.Rows.Add(item.getUserName(), item.getContra());
            }
        }
```

Funcionamiento del Juego:

Al iniciar el juego, en un pictureBox (300x300) se genera un cuadrito de 25x25. Dicho cuadro es la cabeza de la serpiente y cada vez que come, genera una nueva cabeza detrás de es. Basicamente la comida nueva se convierte en una nueva cabeza que se anida a la original convirtiéndola en cola.

```
    class Cola : objeto
    {
        Cola siguiente;
        public Cola(int x, int y)
        {
            this.x = x;
            this.y = y;
            siguiente = null;
        }
        public void dibujar(Graphics g)
        {
            if (siguiente!=null)
            {
                siguiente.dibujar(g);
            }
            g.FillRectangle(new SolidBrush(Color.Chartreuse), this.x, this.y, this.ancho,
this.ancho);

        }
        public void setXY(int x, int y)
        {
            if (siguiente!=null)
            {
                siguiente.setXY(this.x, this.y);
            }
            this.x=x;
            this.y=y;
        }
        public void meter() {
            if (siguiente==null)
            {
                siguiente = new Cola(this.x,this.y);
            }
            else
            {
                siguiente.meter();

            }
        }

        public int verX()
        {
            return this.x;
        }
        public int verY()
        {
            return this.y;
        }
        public Cola verSiguiente()
        {
            return siguiente;

        }

    }
class Comida : objeto
    {
        int lechuga=30;
```

```
        public Comida()
        {
            //dimensiones de la pantalla/10 en las que se genera la comida
            this.x = generar(lechuga);
            this.y = generar(lechuga);
        }
        public void dibujar(Graphics g)
        {
            g.FillRectangle(new SolidBrush(Color.Blue), this.x, this.y, this.ancho, this.ancho);
        }
        public int generar(int n)
        {
            Random random = new Random();
            int num = random.Next(0, n)*10;
            return num;
        }

        public void colocar()
        {
            this.x = generar(lechuga);
            this.y = generar(lechuga);
        }
    }
```

-Funcionamiento del movimiento:

El movimiento de la culebrita es gracias a los eventos KeyDown de las teclas arriba, abajo, derecha, izquierda. Cuando se mueve se agrega 25 pixeles según la posición que ordenó.

```
  private void Juego_KeyDown(object sender, KeyEventArgs e)
        {

            if (ejex)
            {
                if (e.KeyCode== Keys.Up)
                {
                    ydir = -cuadro;
                    xdir = 0;
                    ejex = false;
                    ejey = true;
                    abajo = 0;
                    izquierda = 0;
                    derecha = 0;
                    arriba = 1;
                    Console.WriteLine(arriba + " " + abajo + " " + izquierda + " " + derecha);
                }
                if (e.KeyCode==Keys.Down)
                {
                    ydir = cuadro;
                    xdir = 0;
                    ejex = false;
                    ejey = true;
                    arriba = 0;
                    izquierda = 0;
                    derecha = 0;
                    abajo = 1;
```

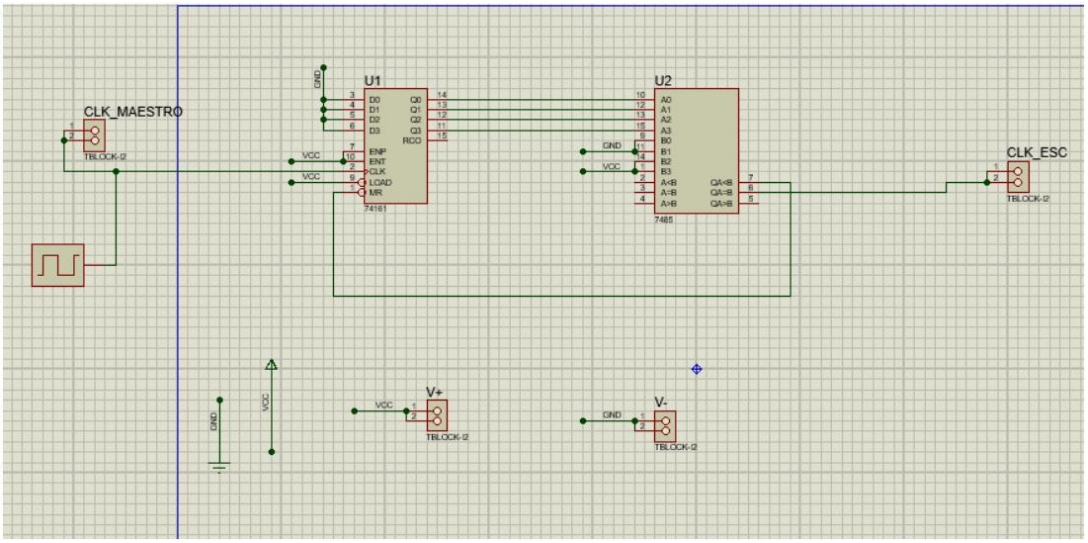
```
            Console.WriteLine(arriba + " " + abajo + " " + izquierda + " " + derecha);
        }
    }
    if (ejey)
    {

        if (e.KeyCode==Keys.Right)
        {
            ydir = 0;
            xdir = cuadro;
            ejex = true;
            ejey = false;
            arriba = 0;
            abajo = 0;
            izquierda = 0;
            derecha = 1;
            Console.WriteLine(arriba+" "+abajo+" "+izquierda+" "+derecha);
        }
        if (e.KeyCode==Keys.Left)
        {
            ydir = 0;
            xdir = -cuadro;
            ejex = true;
            ejey = false;
            arriba = 0;
            abajo = 0;
            derecha = 0;
            izquierda = 1;
            Console.WriteLine(arriba + " " + abajo + " " + izquierda + " " + derecha);
        }
    }
```

**Funciones Booleanas y Diagramas:**

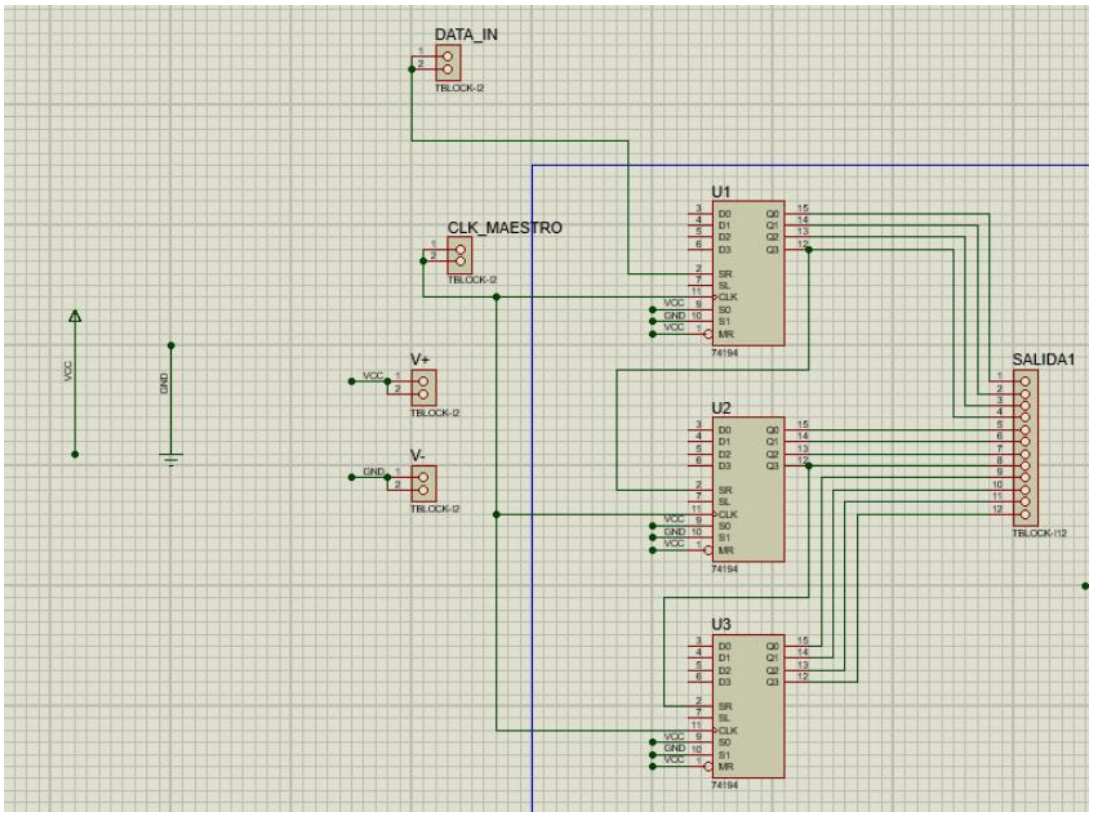
Para el presente proyecto, como lo especifica el enunciado, se requiere de un tipo de monitor realizado con leds representando una matriz de 12x12, la computadora, manda de manera continua los pulsos por columna a modo de pila, teniendo registro serial y un buffer de datos que permiten un flujo continuo de datos, a su vez una demultiplexación para que la cadena de bits se dirija a su respectiva columna.

## Partes de los circuitos y Diagramas con Explicación

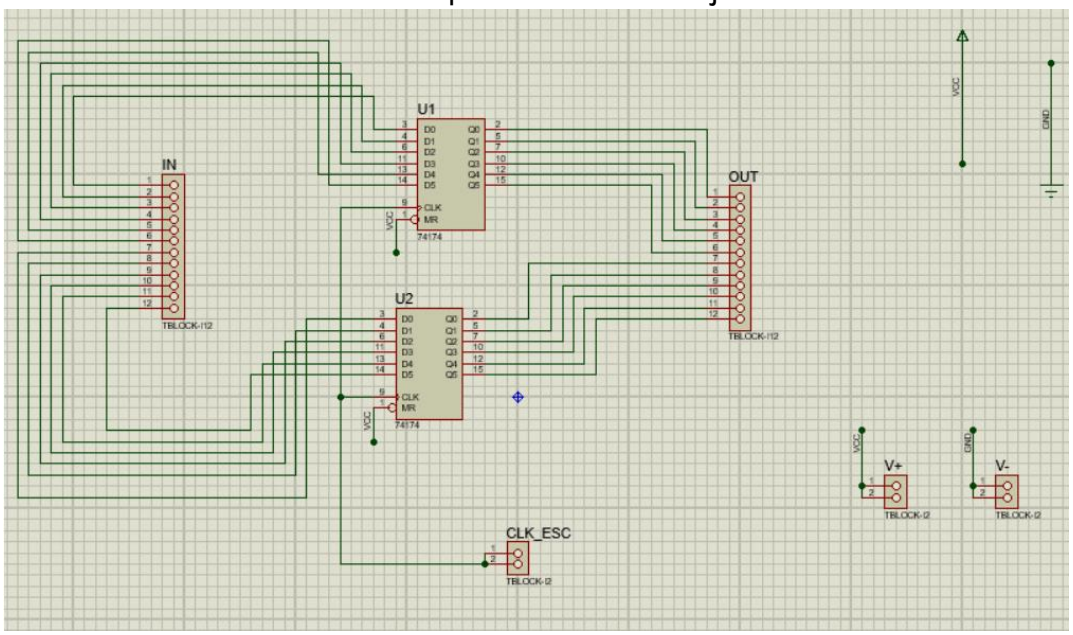
- Contador maestro: es el encargado de contar los pulsos de entrada, este cuando llega a 12 se reinicia mandando un pulso de reloj secundario.



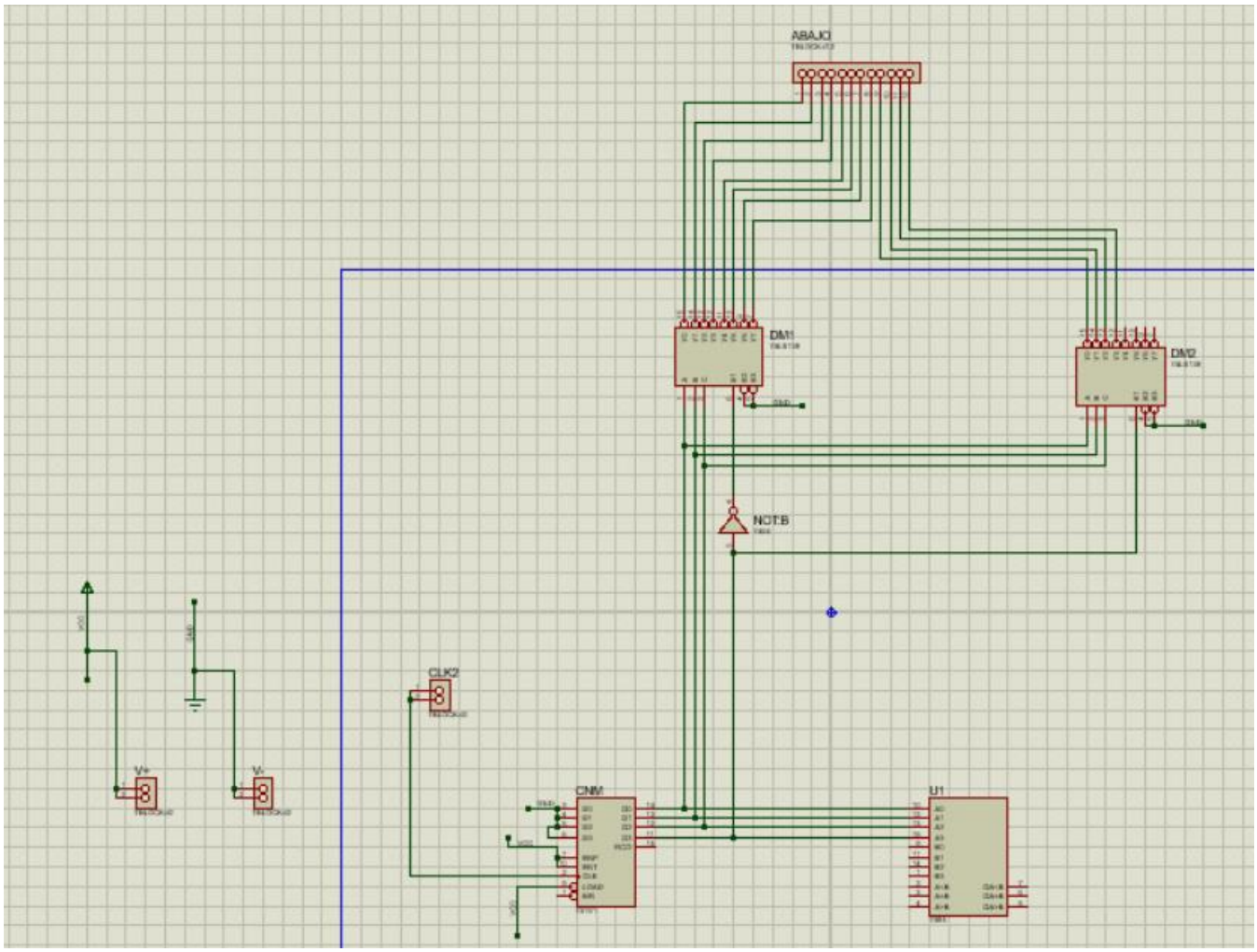

- Memoria serial: es la encargada de almacenar los datos que llegan desde el computador estos solo se desplazan sobre los 12 registros existentes recibiendo los pulsos de reloj que provienen del computador.



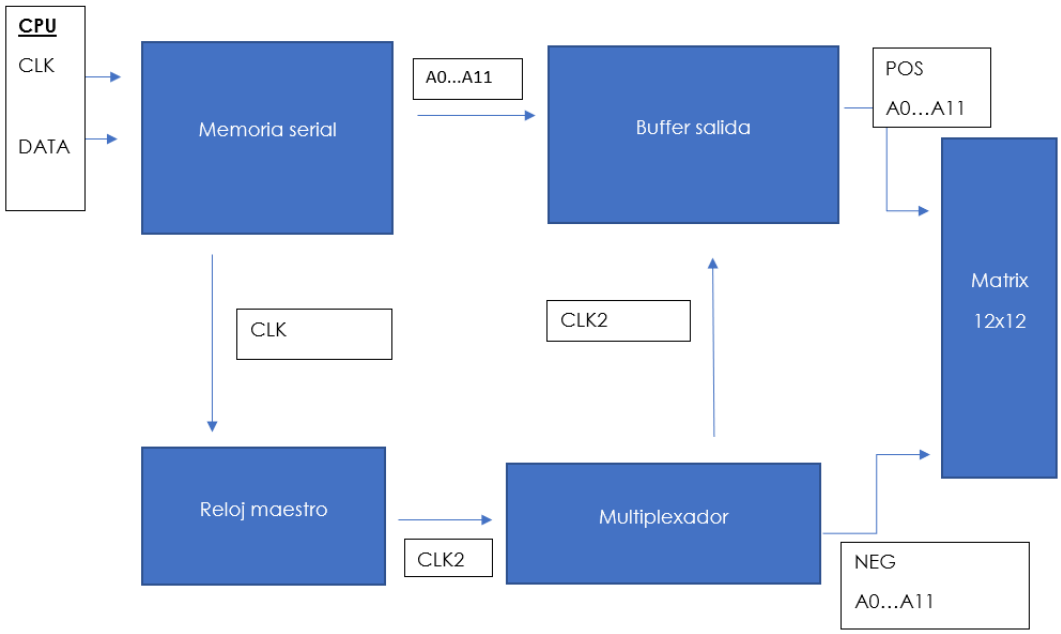

- Buffer salida: su acción es el de almacenar los datos del registro serial, en este caso son 12 bits que se desplazan de manera paralela hacia los leds, teniendo a su entrada de reloj un reloj secundario, el cual se activa cada 12 pulsos del reloj maestro.



- Demultiplexación: es la encargada de completar el circuito de tierra de los leds de la matriz, como los demultiplexores según sea su línea de selección este dará un cero lógico en tal línea, así completando el circuito para los ánodos de los leds.



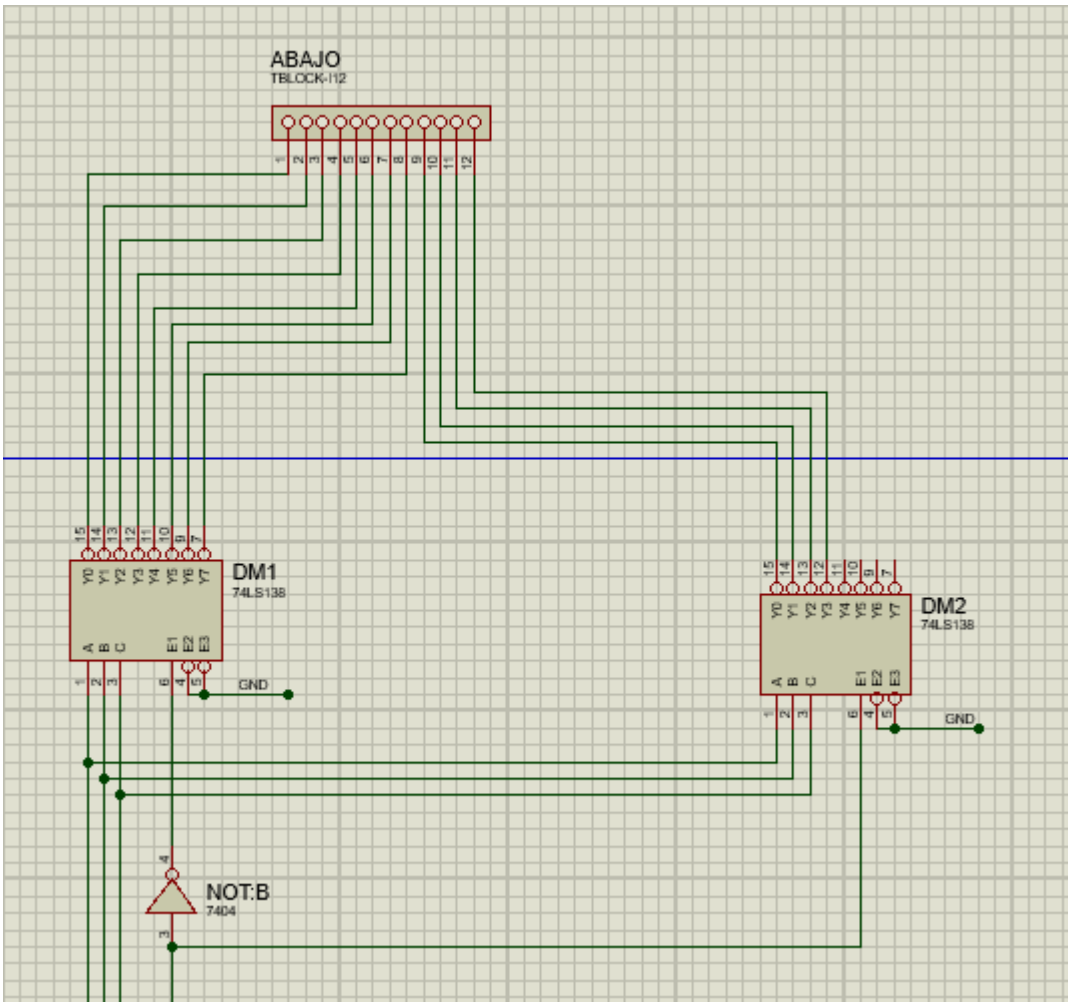
## Vista a Bloque




# Funcionamiento de la multiplexación

Para que la matriz funcione de una manera optima es necesario que exista un selector, que cada vez que una línea se cargue por completo exista un salto en la selección, sin embargo, los demultiplexores existentes solo tienen 8 salidas, entonces, por medio de una tabla de verdad se logró obtener un decoder de 4x16, como solo se necesitan 12 líneas, se ignoran las ultimas 4 líneas.

| Entrada | A0 | A1 | A2 | A3 | A4 | A5 | A6 | A7 | A8 | A9 | A10 | A11 | A12 | A13 | A14 | A15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0000 | 1 | | | | | | | | | | | | | | | |
| 0001 | | 1 | | | | | | | | | | | | | | |
| 0010 | | | 1 | | | | | | | | | | | | | |
| 0011 | | | | 1 | | | | | | | | | | | | |
| 0100 | | | | | 1 | | | | | | | | | | | |
| 0101 | | | | | | 1 | | | | | | | | | | |
| 0110 | | | | | | | 1 | | | | | | | | | |
| 0111 | | | | | | | | 1 | | | | | | | | |
| 1000 | | | | | | | | | 1 | | | | | | | |
| 1001 | | | | | | | | | | 1 | | | | | | |
| 1010 | | | | | | | | | | | 1 | | | | | |
| 1011 | | | | | | | | | | | | 1 | | | | |
| 1100 | | | | | | | | | | | | | 1 | | | |
| 1101 | | | | | | | | | | | | | | 1 | | |
| 1110 | | | | | | | | | | | | | | | 1 | |
| 1111 | | | | | | | | | | | | | | | | 1 |


ABAJO
TBLOCK-I12
DM1
74LS138
DM2
74LS138
GND
GND
NOT:B
7404

## Equipo Utilizado:

- 2 comparadores de magnitud (7485)
- 2 contadores binarios (74161)
- 2 Flip-Flop D 74174
- 3 registros de corrimiento (74194)
- 1 compuerta NOT (7404)
- 2 Demultiplexores (74138)
- Cautin
- 144 LED's azules
- Estaño
- 5 lb de cable galvanizado
- 5 m de cable para protoboard
- Placa de cobre
- 2 protoboard
- 1 multimetro

## Presupuesto:

| Nombre | Código | Cantidad | Precio Unitario | Total |
|---|---|---|---|---|
| **Resistencia** | Variedad | 50 | Q0.50 | Q25.00 |
| **Led Azul** | - | 200 | Q0.50 | Q100.00 |
| **Cable** | - | 6 m. | Q2.00 | Q12.00 |
| **Pasta** | - | 1 | Q7.00 | Q7.00 |
| **Placa de cobre 20x30 cm.** | - | 5 | Q45.00 | Q225.00 |
| **Acido Ferrico** | - | 3 | Q10.00 | Q30.00 |
| **Bases para compuertas** | - | 20 | Q2.00 | Q40.00 |
| **Impresiones Laser Papel fotográfico** | - | 10 | Q5.50 | Q55.00 |
| **Protoboards** | - | 2 | Q28.00 | Q56.00 |
| **Compuerta NOT** | 74LS04 | 5 | Q6.00 | Q30.00 |
| **Comparadores de magnitud** | 74LS85 | 2 | Q10.00 | Q20.00 |
| **Contadores binarios** | 74LS161 | 2 | Q9.00 | Q18.00 |
| **Flip Flop Tipo D** | 74174 | 2 | Q14.00 | Q28.00 |
| **Registros de Corriminto** | 74194 | 3 | Q15.00 | Q45.00 |
| **Demultiplexeros** | 74138 | 3 | Q12.00 | Q24.00 |
| | | | **Total** | Q715.00 |

## Aporte por Integrante:

1. Byron Orellana: Q143.00
2. Gabriel Paz: Q143.00
3. Jackeline Benitez: Q143.00
4. Carlos Tenes: Q143.00
5. Adrián Alvarado: Q143.00

## Conclusiones:

- La comunicación serie o comunicación secuencial, en telecomunicaciones e informática, es el proceso de envío de datos de un bit a la vez, de forma secuencial, sobre un canal de comunicación o un bus.

- La multiplexación es la encargada de completar el circuito de tierra de los leds de la matriz.

- La memoria serial es la encargada de almacenar los datos que llegan desde el computador estos solo se desplazan sobre los 12 registros existentes

## Recomendaciones

- Economizar materiales con la reutilización de componentes de practicas anteriores.

- Utilizar un puerto paralelo para poder regular el voltaje al circuito y sin necesidad de usar un rectificador de voltaje

- Empezar con antelación y planificar previamente la realización del proyecto