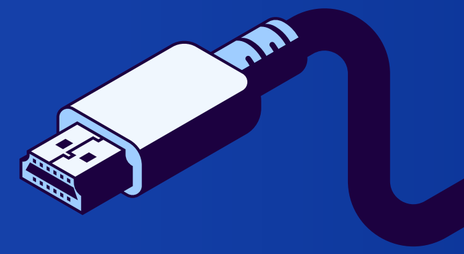
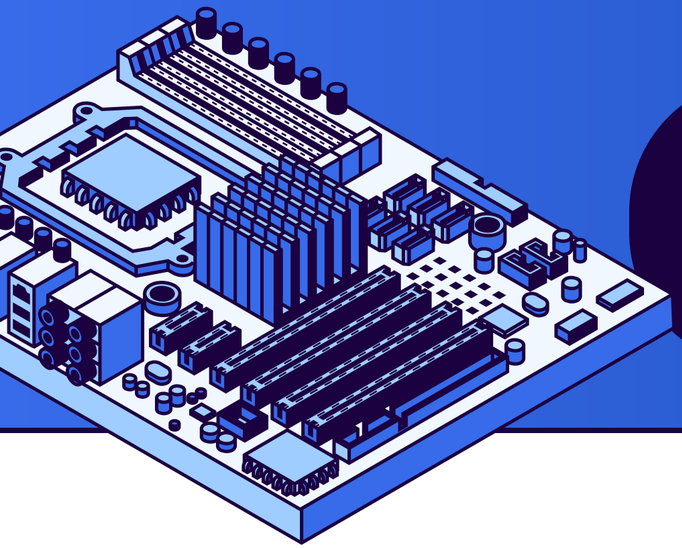
SASE: Proyecto Estudiantil

2025

Sistema FPGA para control inalámbrico y visualización gráfica



DE PIETRO, GUILLERMO L.; PREVES, SANTIAGO; PUEBLA, RODRIGO A.



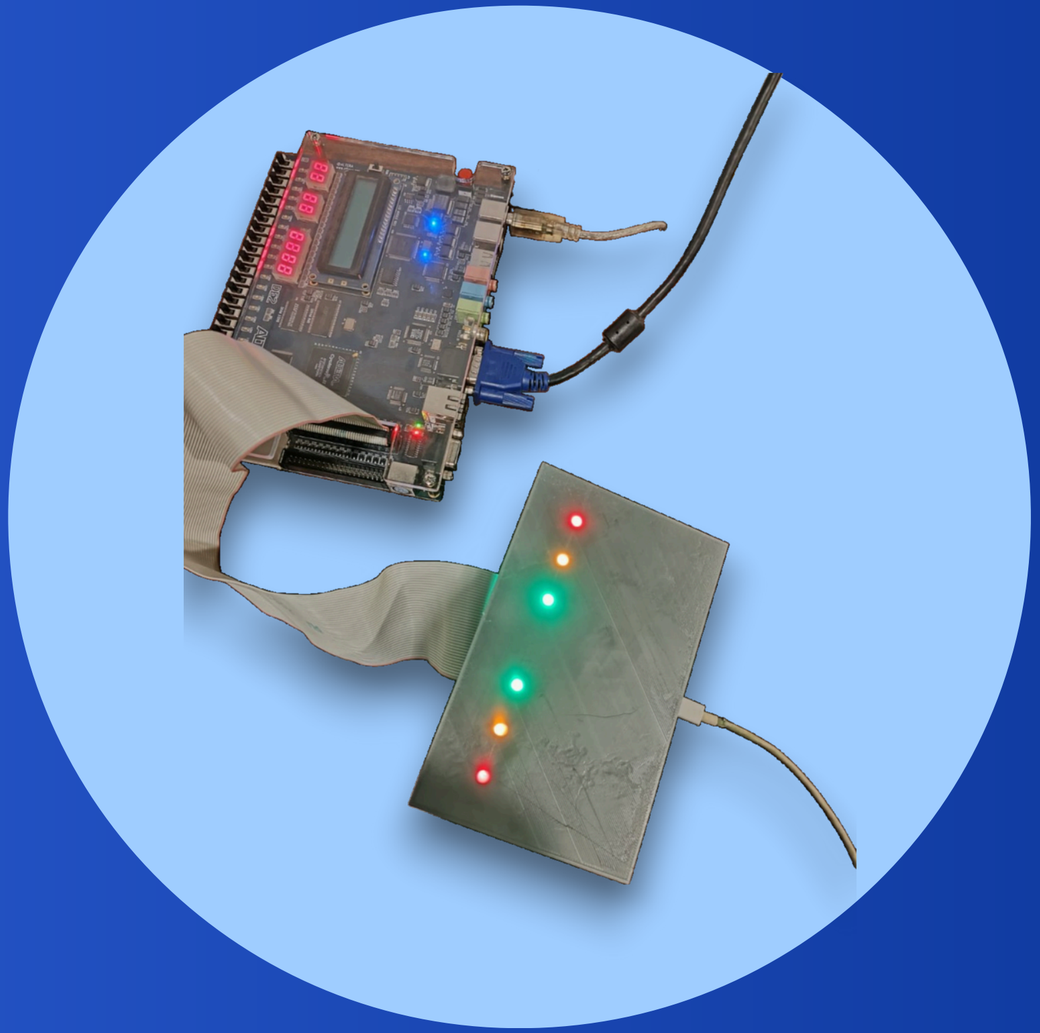
DEPARTAMENTO DE INGENIERÍA ELECTRÓNICA Y COMPUTACIÓN

DOCENTES: MEDINA, M.; RABIOGLIO, L.

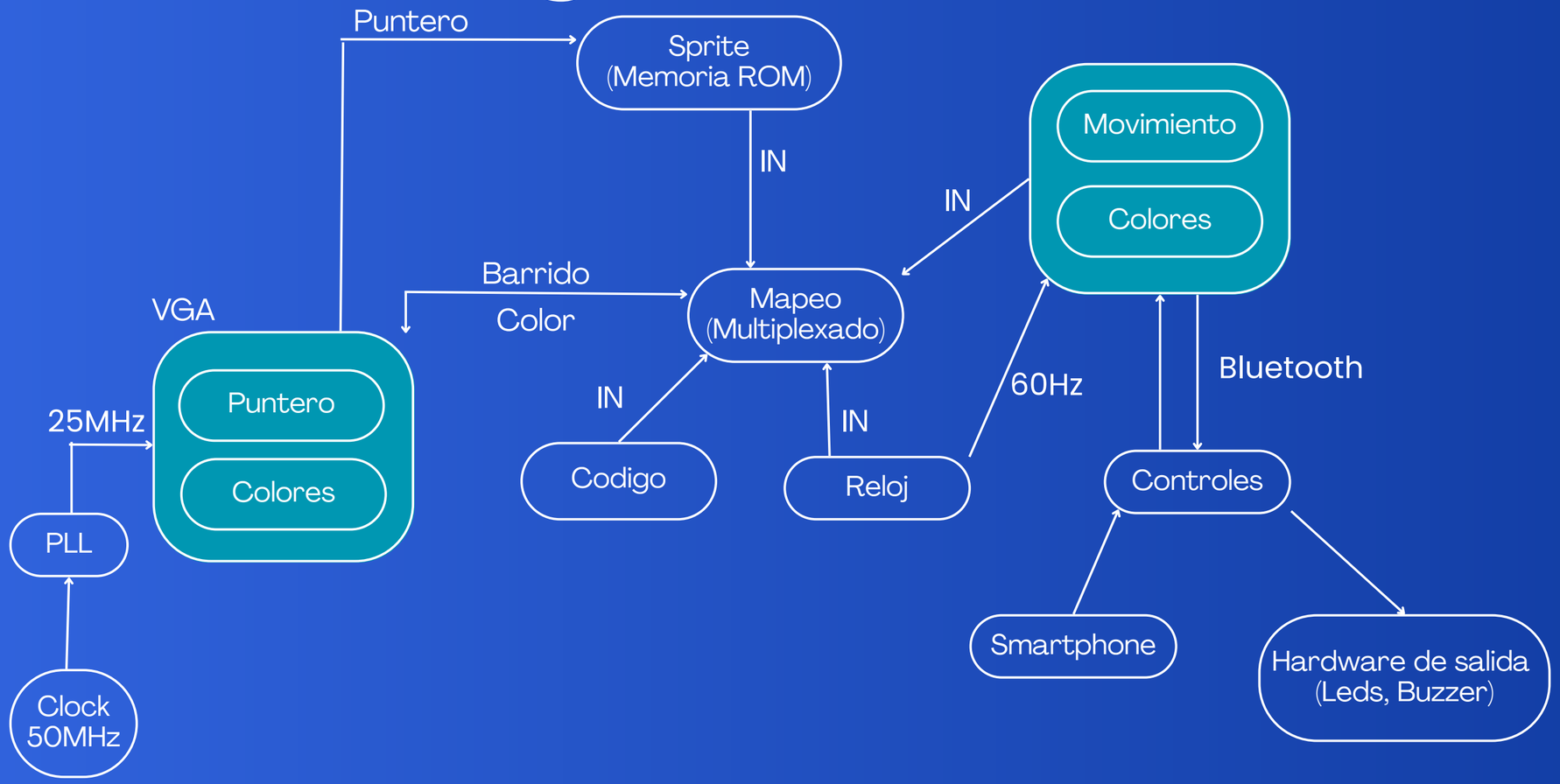
Facultad de Ingeniería
Universidad Nacional de Mar del Plata

Motivación y objetivos

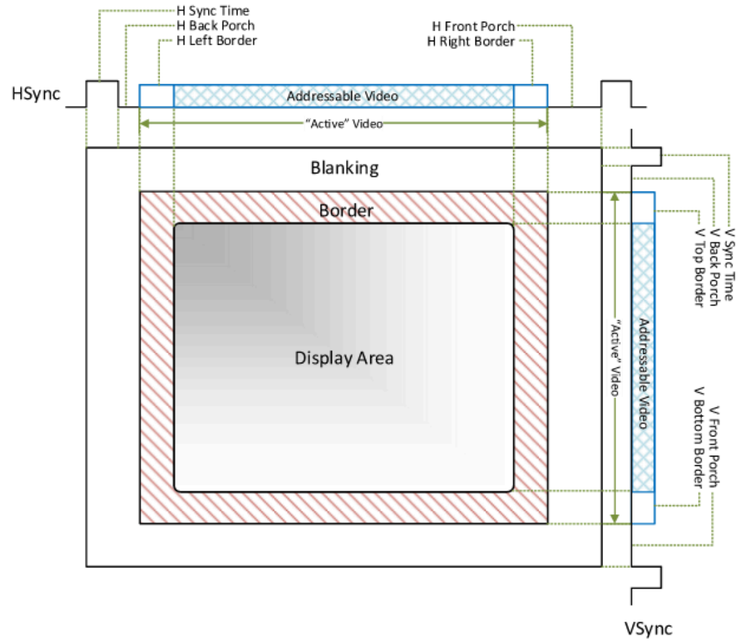
- ¿Por qué un Pong?
- Explorar integración entre FPGA y microcontroladores.
- Implementar lógica digital, comunicaciones y visualización.



Estructura general del sistema

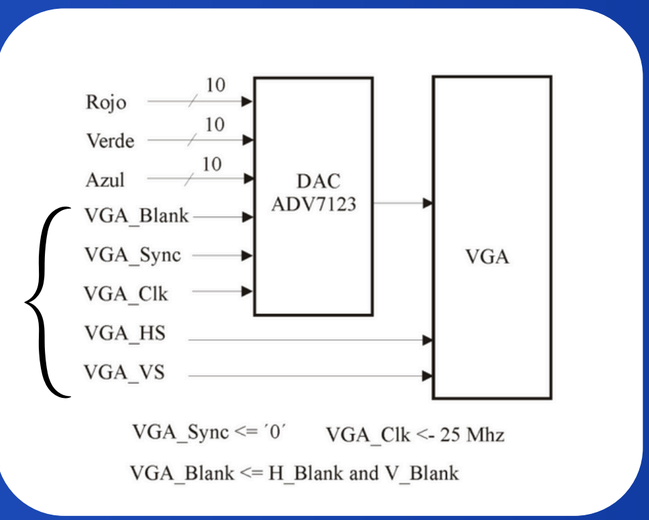


¿Cómo funciona el protocolo VGA?



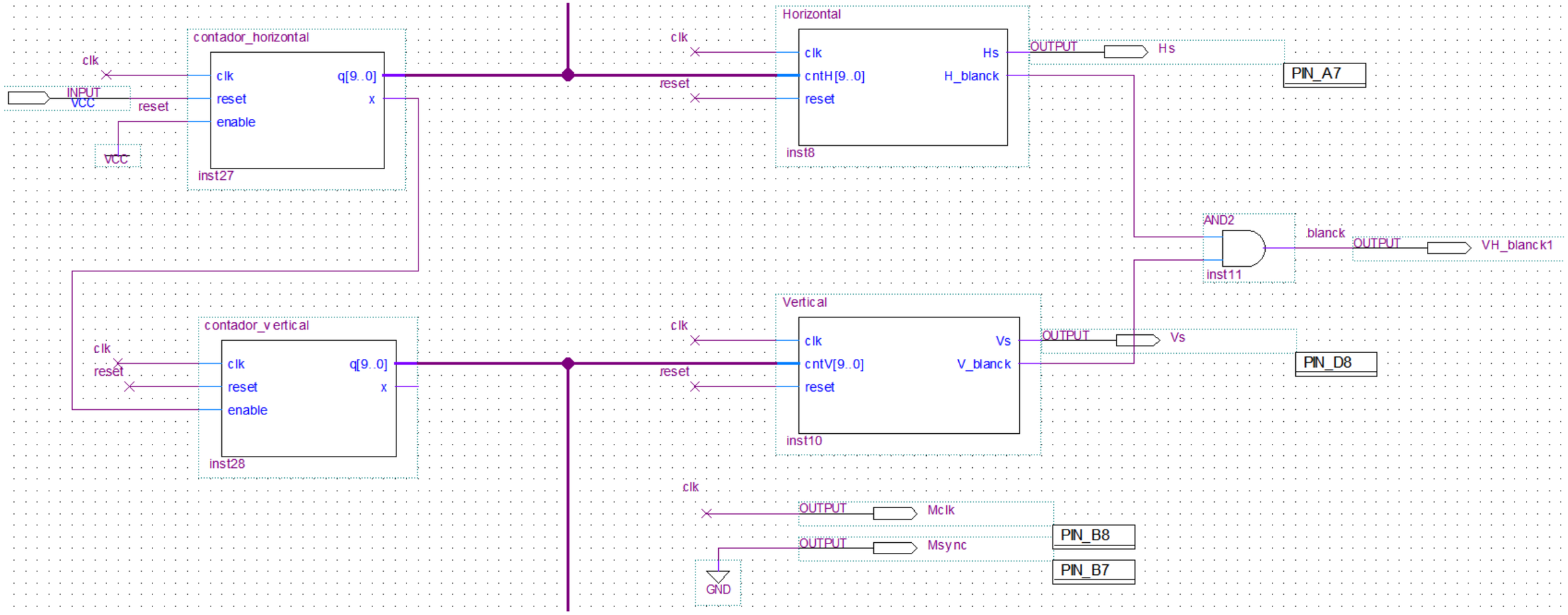
Señales a generar

Bloques internos de la FPGA



SASE: Proyecto Estudiantil

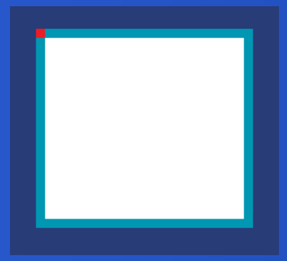
2025

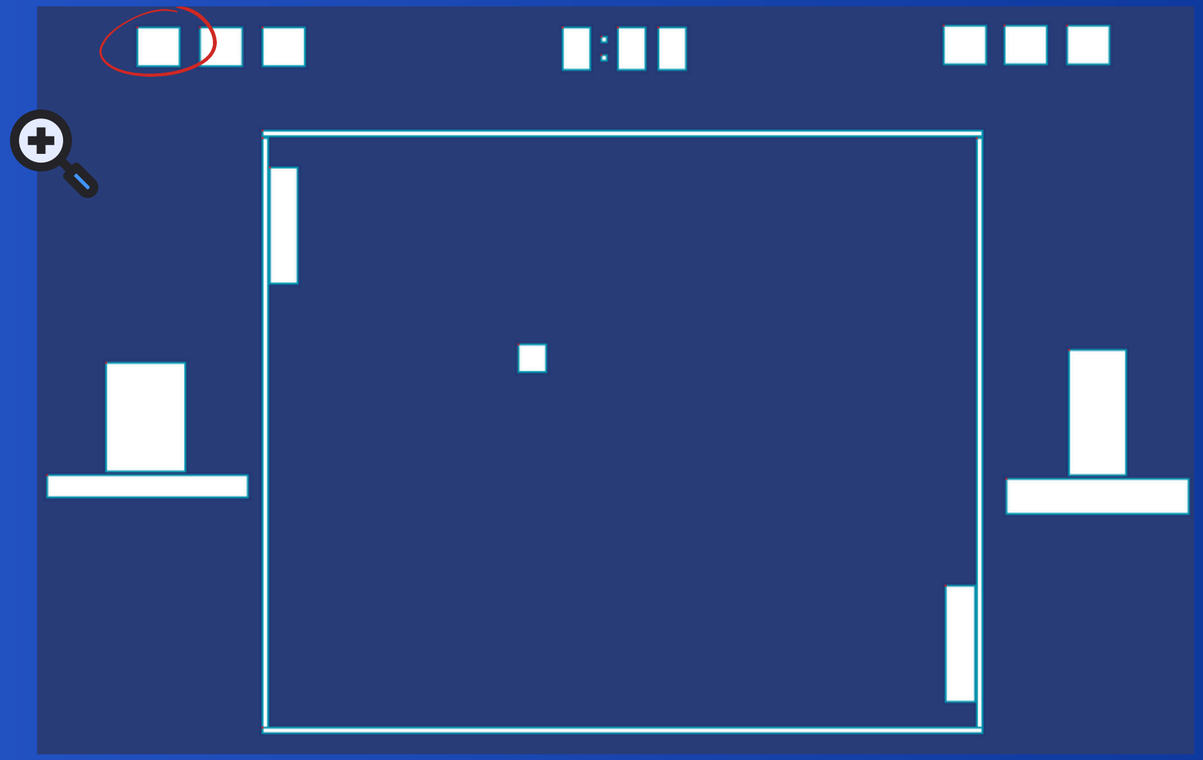


Facultad de Ingeniería
Universidad Nacional de Mar del Plata

¿Cómo se crea la interfaz?

- Subdividir la pantalla en recuadros para asignarles un comportamiento distinto



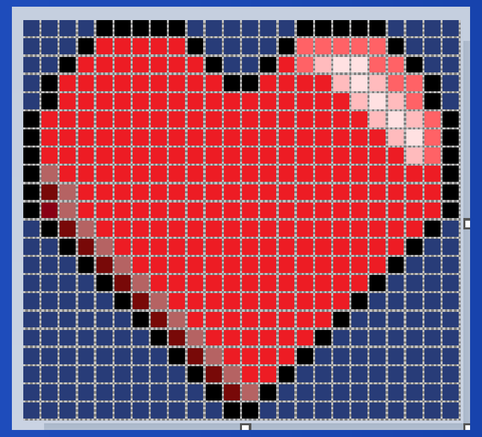


```
if (cntH >= x0) and (cntH < x0 + ancho) and
   (cntV >= y0) and (cntV < y0 + alto) then

   RojoM    <=  Rojo_deseado;
   VerdeM   <=  Verde_deseado;
   AzulM    <=  Azul_deseado;

end if;
```

¿Cómo generar imágenes?

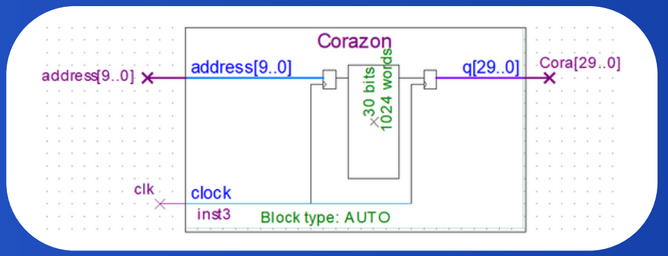
Opción 1: Sprites

- Dibujar imágenes
- Transformar a .mif
- Cargar en ROM
- Generar puntero para recorrer esa ROM y sincronizar con barrido



```
WIDTH=30;
DEPTH=528;

ADDRESS_RADIX=DEC;
DATA_RADIX=BIN;

CONTENT BEGIN
0 : 000010100000011110000011111000;
1 : 000010100000011110000011111000;
2 : 000010100000011110000011111000;
3 : 000010100000011110000011111000;
4 : 000000000000000000000000000000;
5 : 000000000000000000000000000000;
6 : 000000000000000000000000000000;
7 : 000000000000000000000000000000;
8 : 000000000000000000000000000000;
9 : 000010100000011110000011111000;
10 : 000010100000011110000011111000;
11 : 000010100000011110000011111000;
```

```
    signal cnt : integer range 0 to 2640 := 0;
    constant X0 : integer := 200 ; -- 56 + 144borde negro
    constant Y0 : integer := 55;  -- 20 + 35bordenegro
    signal   Tx: integer := 0;
    constant Ty : integer := 202+35;
begin
    Tx <= 555 + 144 when Led_gPl = '0' else 39 + 144;

    process(clk)
    begin
        if rising_edge(clk) then
            if (cntH >= X0-3 and cntH < X0-3 + 24 and cntV >= Y0 and cntV < Y0 + 22) then    -- el -3 es para sincronizar
                cnt <= (cntH-(X0-3))+(cntV-Y0)*24;
            elsif (cntH >= Tx-3 and cntH < Tx-3 + 44 and cntV >= Ty and cntV < Ty + 60) then
                cnt <= (cntH-(Tx-3))+(cntV-Ty)*44;
            end if;
        end if;
    end process;
    address <= std_logic_vector(to_unsigned(cnt,12));
end rtl;
```



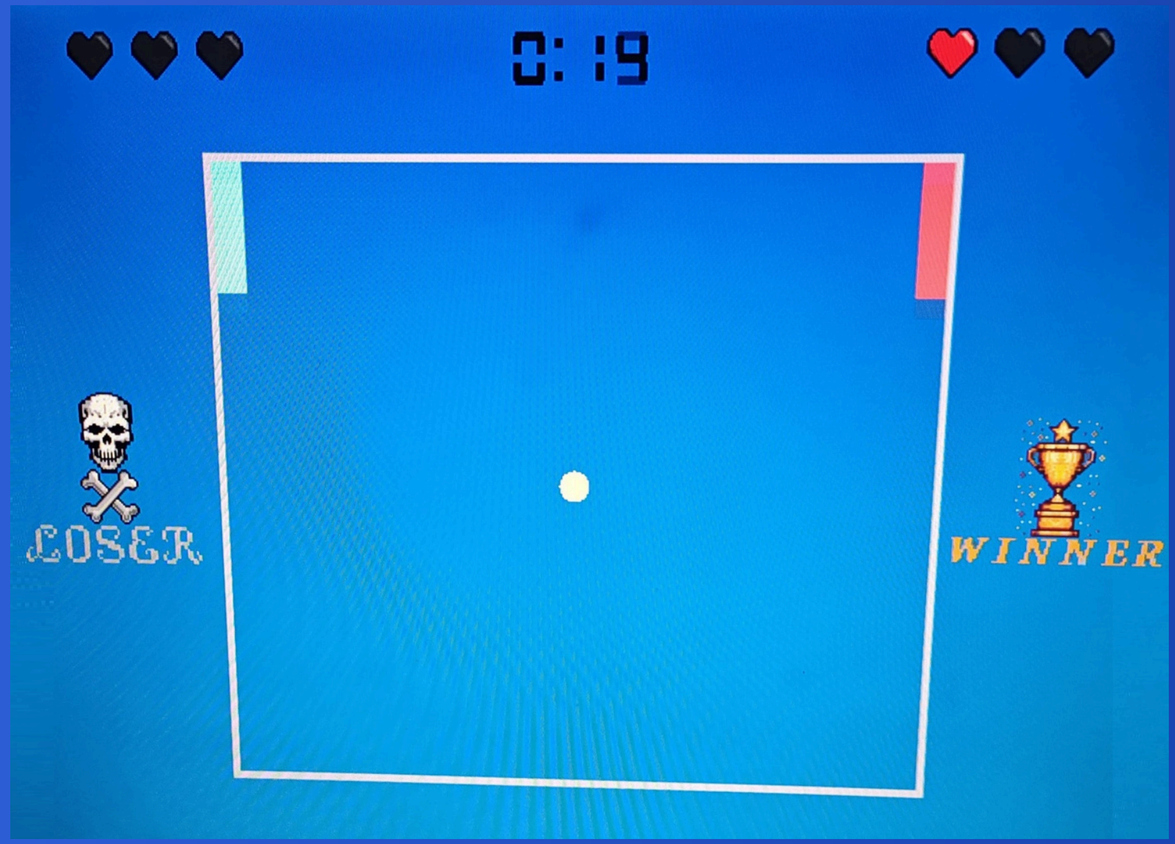
¿Cómo generar imágenes?

Opción 2: Sectorizar subdivisión

Cada número se representa como un display de 7 segmentos, activando zonas específicas según su valor, sin necesidad de usar memoria para imágenes

```
function graficar_digito(digito : integer; H, V : integer; baseX, baseY : integer) return boolean is -- funcion para graficar digitos
    constant x : integer := H - baseX;  -- normalizo x
    constant y : integer := V - baseY;  -- normalizo y
    variable segs : std_logic_vector(6 downto 0);
    begin
        case digito is
            when 0 => segs := "1111110";
            when 1 => segs := "0110000";
            when 2 => segs := "1101101";
            when 3 => segs := "1111001";
            when 4 => segs := "0110011";
            when 5 => segs := "1011011";
            when 6 => segs := "1011111";
            when 7 => segs := "1110000";
            when 8 => segs := "1111111";
            when 9 => segs := "1111011";
            when others => segs := (others => '0');
        end case;
            if segs(6) = '1' and y >= 0  and y <= 3  and x >= 1  and x <= 14 then return true; end if; -- a (horizontal)
            if segs(5) = '1' and x >= 12 and x <= 15 and y >= 2  and y <= 9  then return true; end if; -- b (vertical)
            if segs(4) = '1' and x >= 12 and x <= 15 and y >= 14 and y <= 21 then return true; end if; -- c (vertical)
            if segs(3) = '1' and y >= 20 and y <= 23 and x >= 1  and x <= 14 then return true; end if; -- d (horizontal)
            if segs(2) = '1' and x >= 0  and x <= 3  and y >= 14 and y <= 21 then return true; end if; -- e (vertical)
            if segs(1) = '1' and x >= 0  and x <= 3  and y >= 2  and y <= 9  then return true; end if; -- f (vertical)
            if segs(0) = '1' and y >= 10 and y <= 13 and x >= 1  and x <= 14 then return true; end if; -- g (horizontal)
        return false;
end function;
```

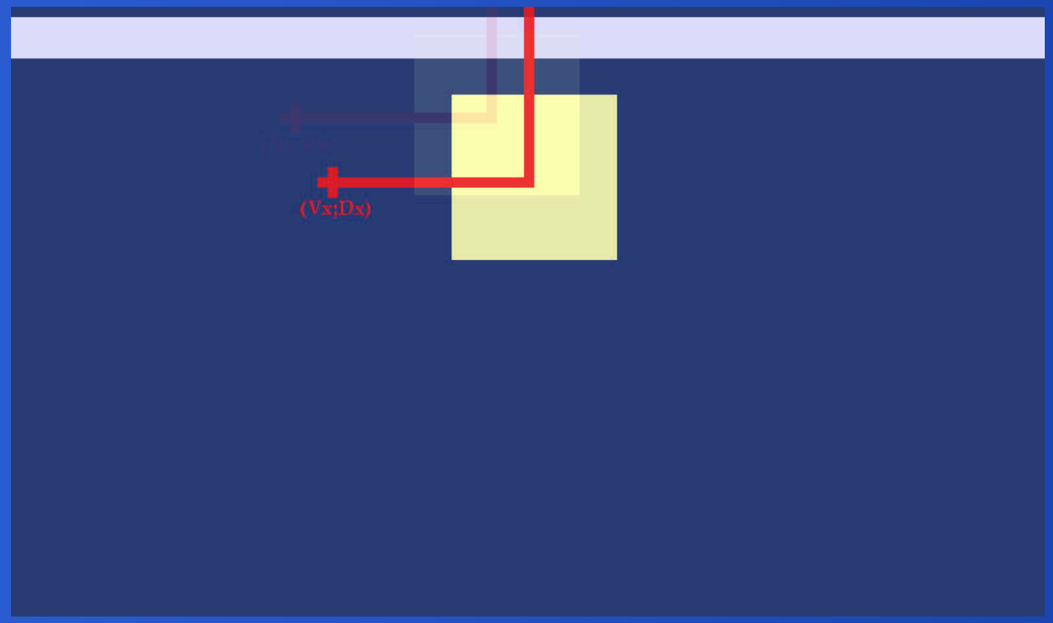
Resultado final



Mecanicas

Movimiento

```
if diry = 1 then
    p_ejey <= p_ejey + Vy;
else
    p_ejey <= p_ejey - Vy;
end if;
```



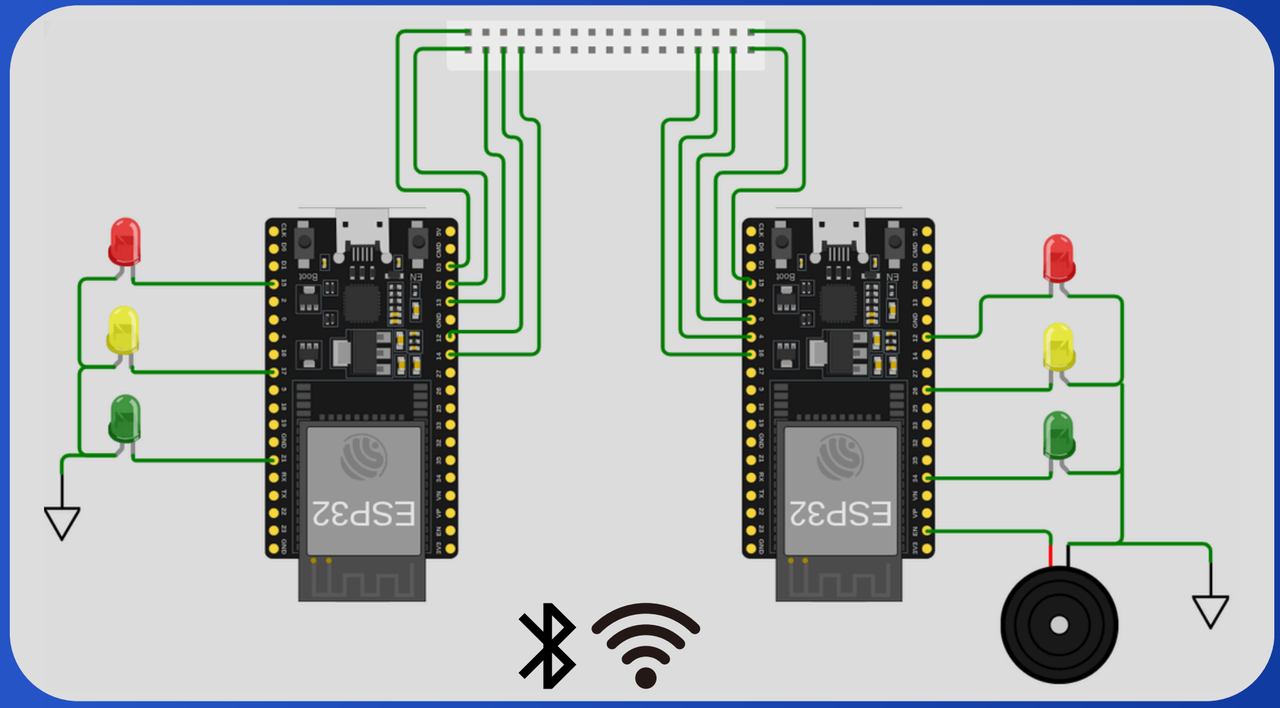
Colisión

```
if p_ejey <= 115 then
    p_ejey <= 115 ;
    diry <= 1;
    Colision <= '1';
```

- Mediante dos contadores truncados que operan a frecuencias distinas se logra velocidad aleatoria.

Controles con ESP32

- Se conecta por Bluetooth con smartphone.
- Envía señales de control (arriba, abajo y start).
- Recibe señales desde FPGA (ganador y colisión).
- Genera lógica de LED y BUZZ.



Posibles ampliaciones

- Conexión por red WIFI y jugador remoto.
- Modo de entrenamiento con distintos niveles de dificultad.
- Menú de selección de modo.
- Power-up (doble pelota, cambio de tamaño de barra, etc).

Repositorios

PLAY STORE- ARDUINO BLUETOOTH CONTROLLER

GITHUB DEL PROYECTO

MUCHAS GRACIAS

PARA MAS INFORMACIÓN:
GDEPIETRO; SPREVES; RODRIGOPUEBLA (@ALUMNOS.FI.MDP.EDU.AR)