


SISTEMAS DIGITALES I



**Automatización de un sistema de
iluminación y seguridad de una vivienda**

Presentado por:

- **Xavier Chalen Jaramillo**
- **José Aguirre Vacas**

MAQUETA Y SISTEMA

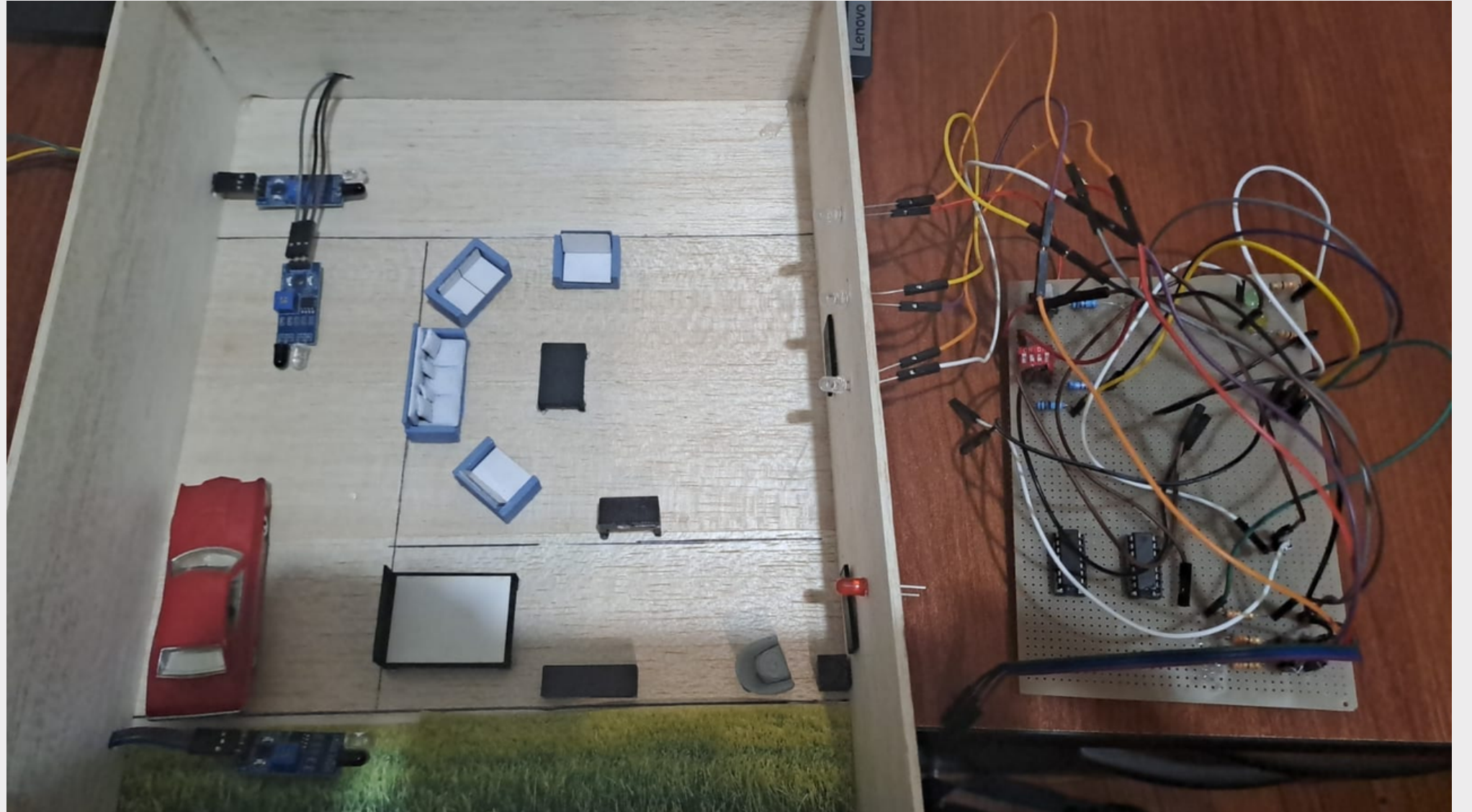
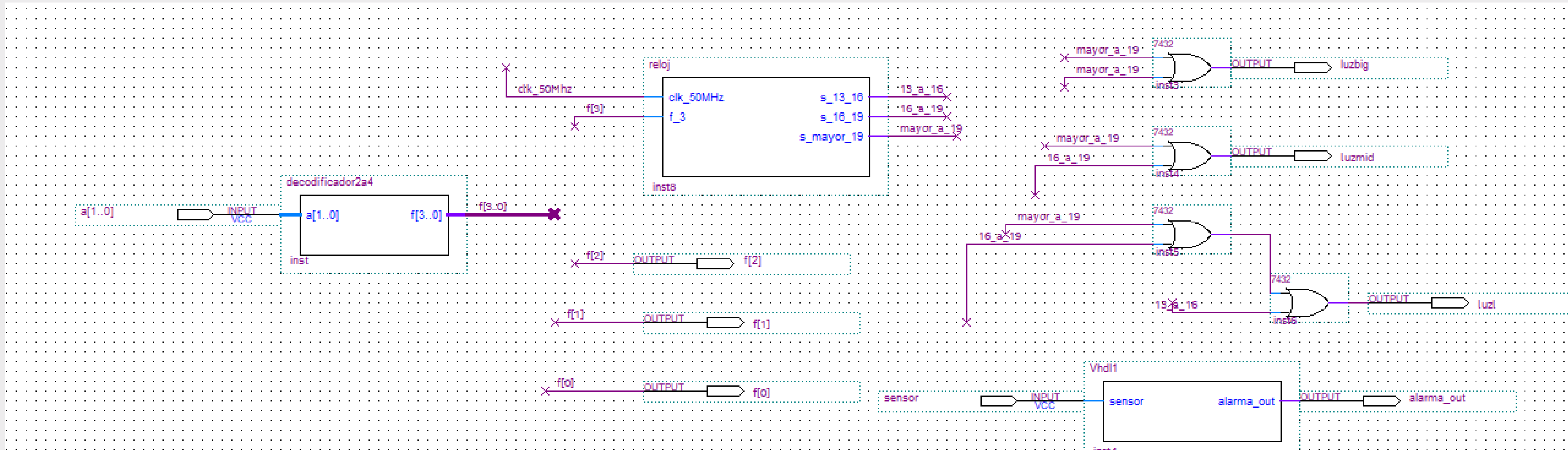


DIAGRAMA DE BLOQUES



CÓDIGO DE RELOJ

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
use IEEE.NUMERIC_STD.ALL;

entity reloj is
    Port ( clk_50MHz : in STD_LOGIC;           -- Entrada de reloj de 50MHz
          f_3 : in STD_LOGIC;                -- Entrada interna para activar el comportamiento
          s_13_16 : out STD_LOGIC;
          s_16_19 : out STD_LOGIC;
          s_mayor_19 : out STD_LOGIC);
end reloj;

architecture comportamiento of reloj is
    signal contador : INTEGER range 0 to 24 := 0;    -- Contador ajustado para contar hasta 24
    signal divisor : INTEGER := 0;
    signal tick : STD_LOGIC := '0';                -- Inicializa la señal de pulso a '0'
    signal reseteo : INTEGER range 0 to 499999999 := 0; -- Temporizador para 240 segundos

begin
    -- Divisor de frecuencia: genera un pulso cada 500 millones de ciclos de reloj (50MHz / 0.1Hz)
    process(clk_50MHz, f_3)
    begin
        if rising_edge(clk_50MHz) then
            -- Verifica la entrada de activación antes de continuar con el comportamiento
            if f_3 = '1' then
                -- Temporizador de reset: reinicia después de 240 segundos (24 horas)
                if reseteo = 499999999 then
                    reseteo <= 0; -- Reinicia el temporizador
                    divisor <= 0;
                end if;
            end if;
        end if;
    end process;
end;
```

```

        divisor <= 0;
        tick <= '1'; -- Genera un pulso cuando el temporizador y el divisor coinciden
    else
        reseteo <= reseteo + 1; -- Temporizador sigue avanzando
        divisor <= divisor + 1;
        tick <= '0';
    end if;
end if;
end if;
end process;

-- Contador: incrementa en cada pulso de 10 segundossss
process(clk_50MHz)
begin
    if rising_edge(clk_50MHz) then
        if tick = '1' then
            if contador = 24 then
                contador <= 0; -- Reinicia después de contar hasta 24
            else
                contador <= contador + 1;
            end if;
        end if;
    end if;
end process;

s_13_16 <= '1' when (contador >= 13 and contador <= 16) else '0';
s_16_19 <= '1' when (contador > 16 and contador <= 19) else '0';
s_mayor_19 <= '1' when (contador > 19) else '0';

end comportamiento;

```

CÓDIGO DEL DECODIFICADOR

```
library ieee;
use ieee.std_logic_1164.all;

entity decodificador2a4 is
port (a: in std_logic_vector(1 downto 0);
f: out std_logic_vector(3 downto 0));
end decodificador2a4;

architecture funcionamiento of decodificador2a4 is
begin
process(a)
begin
case a is
    when "00" =>
        f <= "1100";
    when "01" =>
        f <= "0110";
    when "10" =>
        f <= "0110";
    when "11" =>
        f <= "0011";
    end case;
end process;
end funcionamiento;
```

CÓDIGO CORRECTOR DEL SENSOR

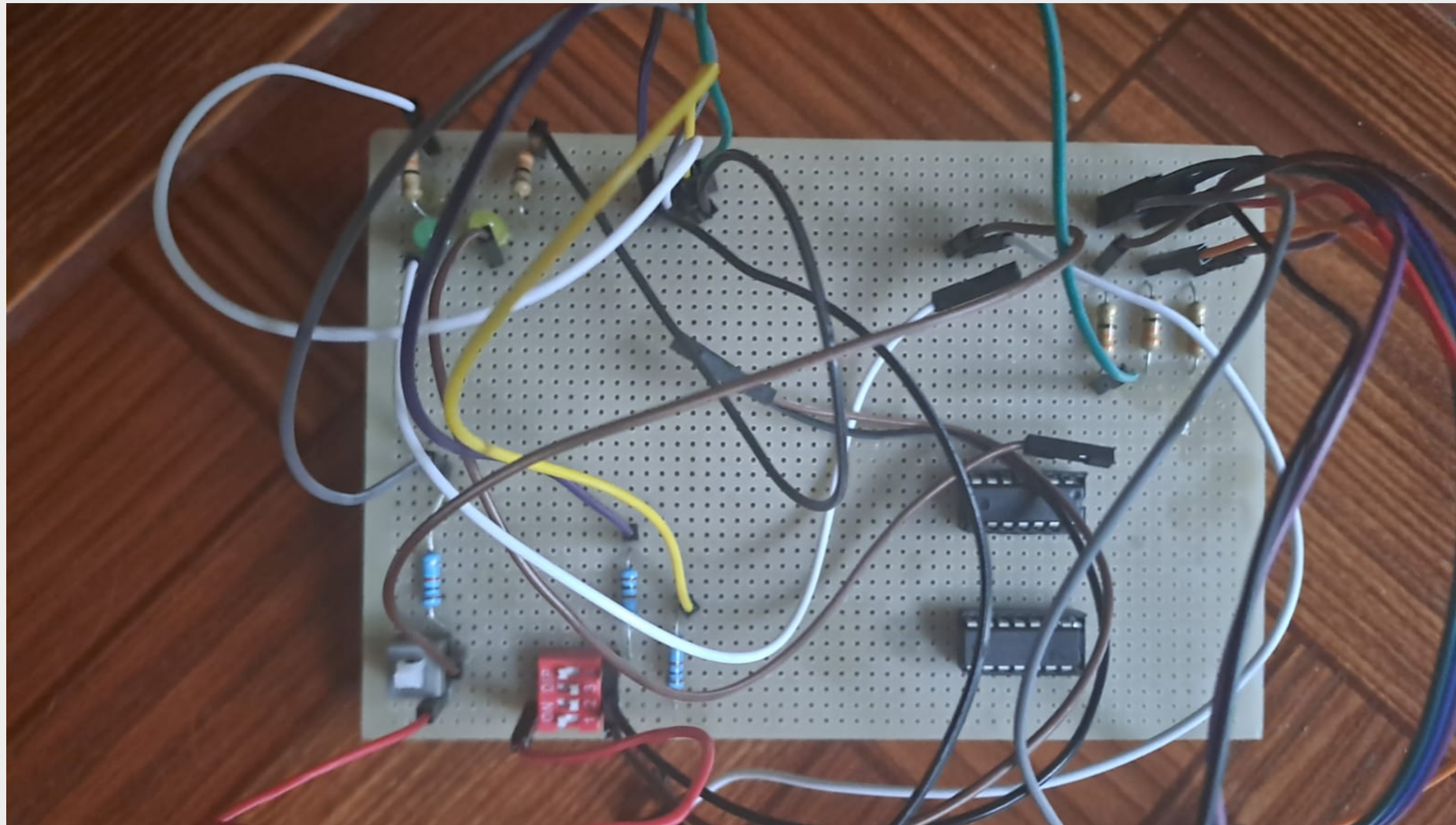
```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity vhd11 is
    Port ( sensor : in STD_LOGIC;          -- salida del sensor
          alarma_out : out STD_LOGIC); -- salida de la alarma
end vhd11;

architecture Behavioral of vhd11 is
    signal alarma_activada : STD_LOGIC := '0'; -- Estado interno de la alarma
begin
    process(sensor)
    begin
        if sensor = '1' then
            alarma_activada <= '1'; -- Activar la alarma si se detecta movimiento
        end if;

        -- La alarma permanece activada incluso si el sensor deja de detectar movimiento
        alarma_out <= alarma_activada;
    end process;
end Behavioral;
```


Conexiones físicas





GRACIAS