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

entity velocimetro is
    Port ( clk          : in STD_LOGIC;
           reset        : in STD_LOGIC;
           sensor       : in STD_LOGIC;
           VueltasOut   : out std_logic_vector(5 downto 0));
end velocimetro;

architecture Behavioral of velocimetro is

    constant Conta1segMAX : integer := 50*(10**6) - 1;
    signal conta1seg : integer range 0 to Conta1segMAX;
    signal UnSeg : std_logic;

    signal Vueltas: unsigned(5 downto 0);

begin

    -- Divisor de Frecuencia a 1 segundo
    DivFreq: process(clk,reset)
    begin
        if reset = '1' then
            conta1seg <= 0;
        elsif clk'event and clk = '1' then
            if conta1seg < Conta1segMAX then
                conta1seg <= conta1seg + 1;
            else
                conta1seg <= 0;
            end if;
        end if;
    end process;
    UnSeg <= '1' when conta1seg = Conta1segMAX else '0';

    -- Proceso en el que se cuentan las vueltas y se actualiza la salida (1 vez 
    por segundo)
    ContaVueltas: process(clk,reset)
    begin
        if reset = '1' then
            Vueltas     <= "000000";
            VueltasOut  <= "000000";
        elsif clk'event and clk = '1' then
            if Unseg = '1' then
                VueltasOut <= std_logic_vector(Vueltas);
                Vueltas <= "000000";
            elsif sensor = '1' then
                    Vueltas <= Vueltas + 1;
            end if;
        end if;
    end process;

end Behavioral;

```

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

entity TopVelocimetro is
    Port ( clk : in STD_LOGIC;
           reset : in STD_LOGIC;
           sensor : in STD_LOGIC;
           leds : out STD_LOGIC_VECTOR(7 DOWNTO 0)
          );
end TopVelocimetro;

architecture Behavioral of TopVelocimetro is

    component velocimetro is
        Port ( clk : in STD_LOGIC;
               reset : in STD_LOGIC;
               sensor : in STD_LOGIC;
               VueltasOut   : out std_logic_vector(5 downto 0));
    end component;

    signal VueltasOut : std_logic_vector(5 downto 0);
    signal uV : unsigned(5 downto 0);

begin

    -- Instancia del componente velocímetro
    instVelo: velocimetro
               port map(
                clk => clk,
                reset => reset,
                sensor => sensor,
                VueltasOut => VueltasOut
                );

    --Transformación a unsigned para facilitar la codificación
    uV <= unsigned(VueltasOut);

    leds <= "00000001" when uV < 8 else
            "00000010" when uV < 16 else
            "00000100" when uV < 24 else
            "00001000" when uV < 32 else
            "00010000" when uV < 40 else
            "00100000" when uV < 48 else
            "01000000" when uV < 56 else
            "10000000";
end Behavioral;

```

```vhdl
-------------------------
-- Electrónica Digital --
-- PEC diciembre 2017  --
-- Cuestión 3          --
-------------------------

library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity c3 is
    port (
        clk   : in  std_logic;
        reset : in  std_logic;
        seq   : out std_logic
    );
end c3;

-------------------------------------------
-- La solución comienza a partir de aquí --
-------------------------------------------

-- a) Base: contador binario síncrono
architecture cnt_binario of c3 is
    signal cnt : unsigned(2 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            cnt <= (others => '0');
        elsif clk'event and clk = '1' then
            cnt <= cnt + 1;
        end if;
    end process;

    with cnt select
        seq <= '1' when "000",
               '1' when "001",
               '0' when "010",
               '1' when "011",
               '1' when "100",
               '0' when "101",
               '0' when "110",
               '1' when "111",
               '-' when others;

end cnt_binario;

-- b) Base: contador Johnson
architecture cnt_johnson of c3 is
    signal shiftreg : std_logic_vector(3 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(2 downto 0) & not shiftreg(3);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "0000",
               '1' when "0001",
               '0' when "0011",
               '1' when "0111",
               '1' when "1111",
               '0' when "1110",
               '0' when "1100",
               '1' when "1000",
               '-' when others;
```

```vhdl

end cnt_johnson;

-- c) Base: contador en anillo
architecture cnt_anillo of c3 is
    signal shiftreg : std_logic_vector(7 downto 0);
begin

    process(clk,reset)
    begin
        if reset = '1' then
            shiftreg <= (0 => '1', others => '0');
        elsif clk'event and clk = '1' then
            shiftreg <= shiftreg(6 downto 0) & shiftreg(7);
        end if;
    end process;

    with shiftreg select
        seq <= '1' when "00000001",
               '1' when "00000010",
               '0' when "00000100",
               '1' when "00001000",
               '1' when "00010000",
               '0' when "00100000",
               '0' when "01000000",
               '1' when "10000000",
               '-' when others;

end cnt_anillo;

```