Universidad Politécnica de Madrid
ETSI de Telecomunicación

Departamento de Ingeniería Electrónica

Circuitos Electrónicos (CELT)

Plantilla de la memoria

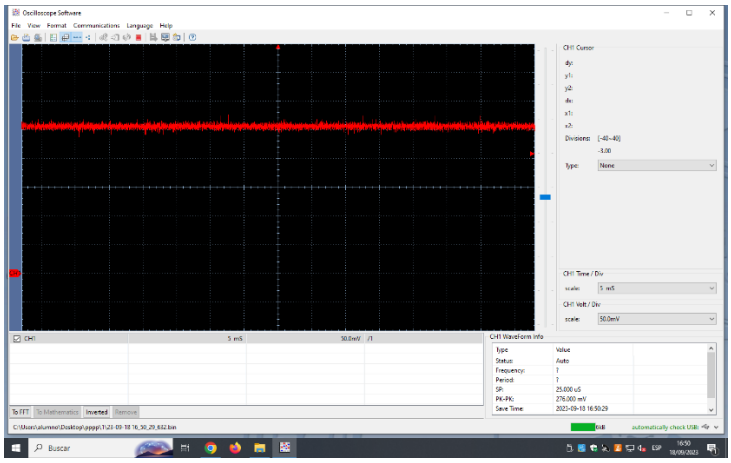
Curso 2023-2024

Termómetro digital

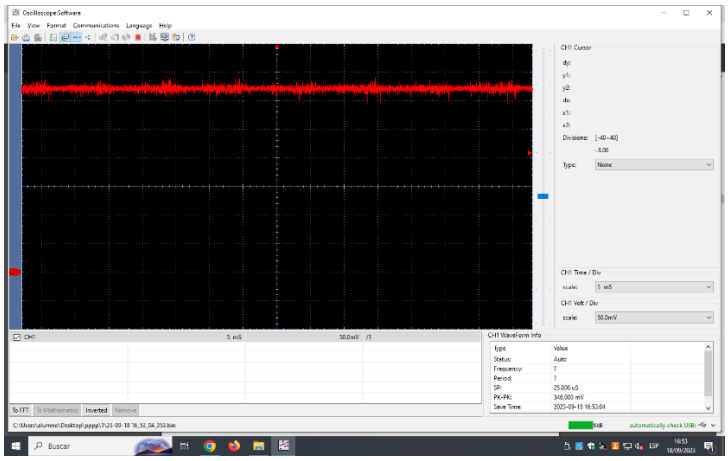
Alumno/a 1	Ignacio Aguirre Candia
Alumno/a 2	Alex Martin-Romo González
Turno	Lunes tarde

MÓDULO 1: El sensor de temperatura LM35

1. Adjunte una captura de pantalla del osciloscopio donde se lea la tensión medida correspondiente a la temperatura de la sala. (También puede emplear las herramientas de medida de tensión del osciloscopio. En el menú "Measure" puede hacer que aparezca el valor medio (V_m) de la tensión en la pantalla del osciloscopio).



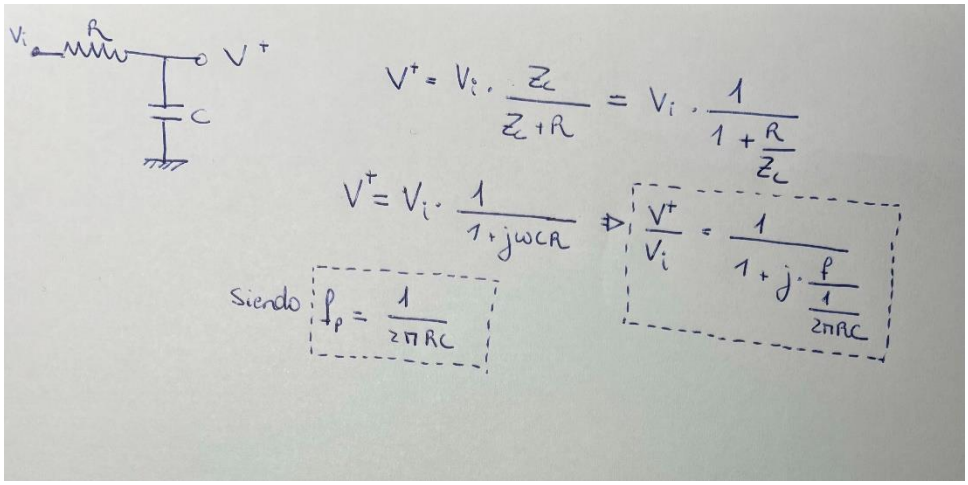
2. Adjunte una captura de pantalla del osciloscopio donde se lea la tensión máxima obtenida una vez calentado el sensor con los dedos.



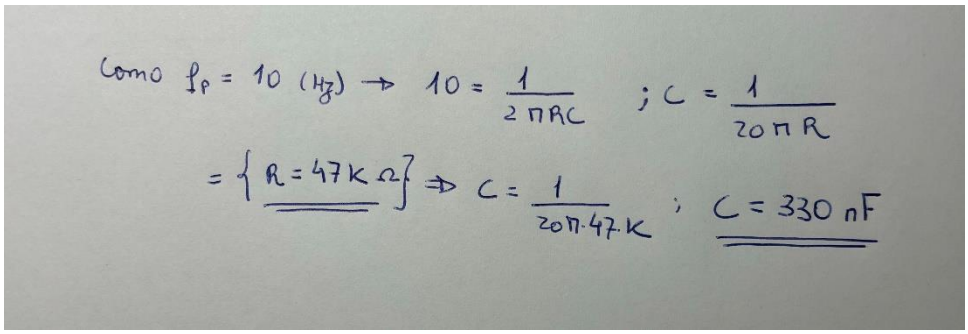
Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

MÓDULO 2: Filtro paso bajo

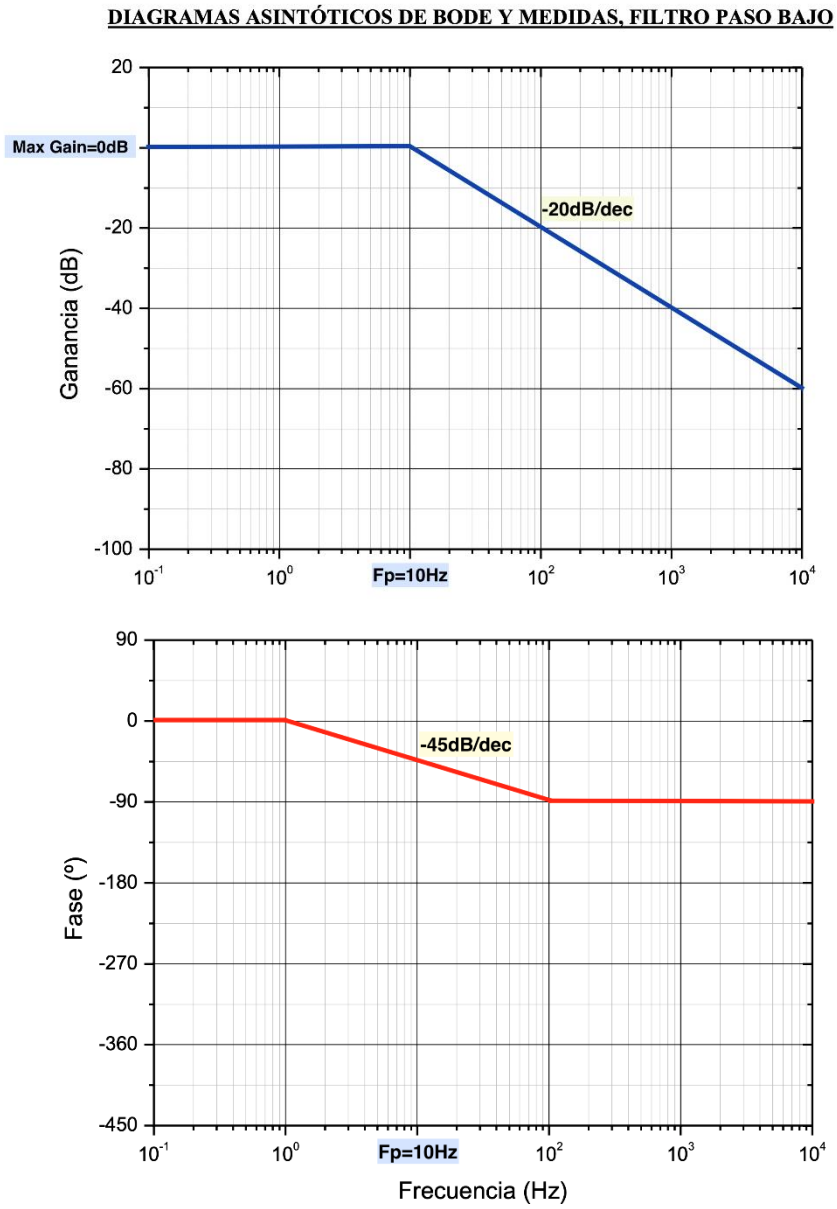
1. Detalle el cálculo de la función de transferencia del filtro paso bajo.



2. Detalle el cálculo de los valores R y C.

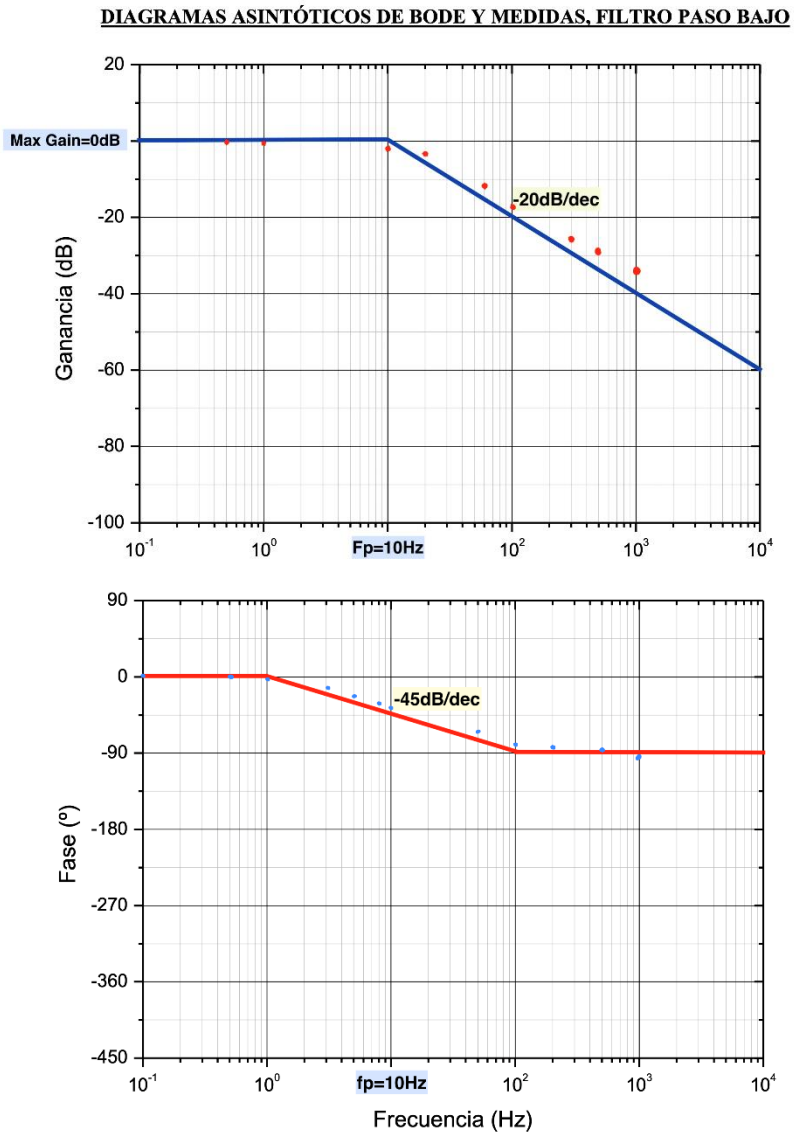


3. Dibuje los diagramas asintóticos de Bode en módulo y fase del filtro empleando las plantillas de las páginas siguientes. Indique la frecuencia del polo, la pendiente de la atenuación, el máximo valor de la ganancia y la pendiente de la fase.



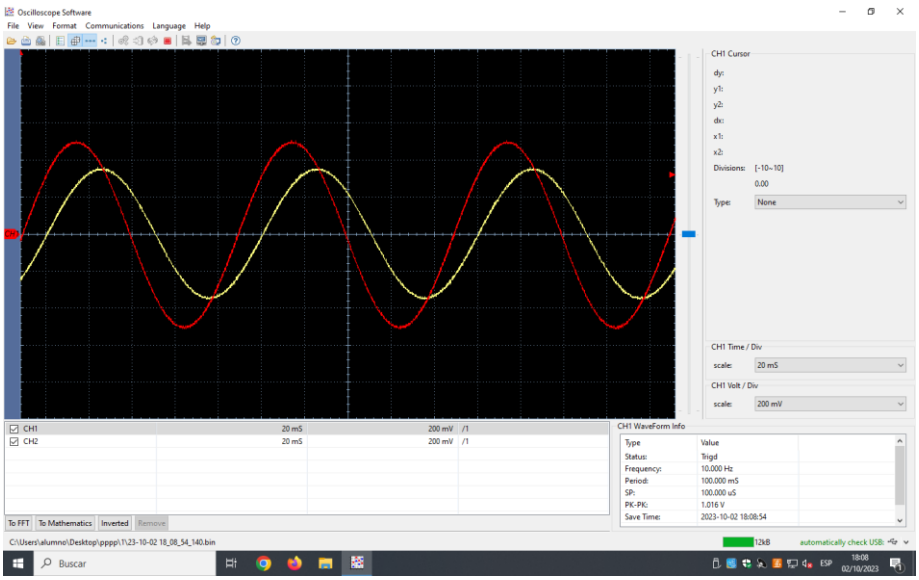
4. Rellene la tabla para la caracterización del filtro e inclúyala.

5. Superponga los puntos de las medidas sobre el diagrama asintótico de Bode de modo que se vean las gráficas de función de transferencia de ganancia y fase



6. Indique el valor de la frecuencia de corte (f_c) medida con precisión.

7. Incluya una captura de pantalla del osciloscopio donde se aprecie la señal a la salida del filtro paso bajo (punto v_+) en el canal 1 y a la entrada (punto v_s) en el canal 2 cuando éste se encuentra alimentado con la frecuencia de corte exacta.



Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

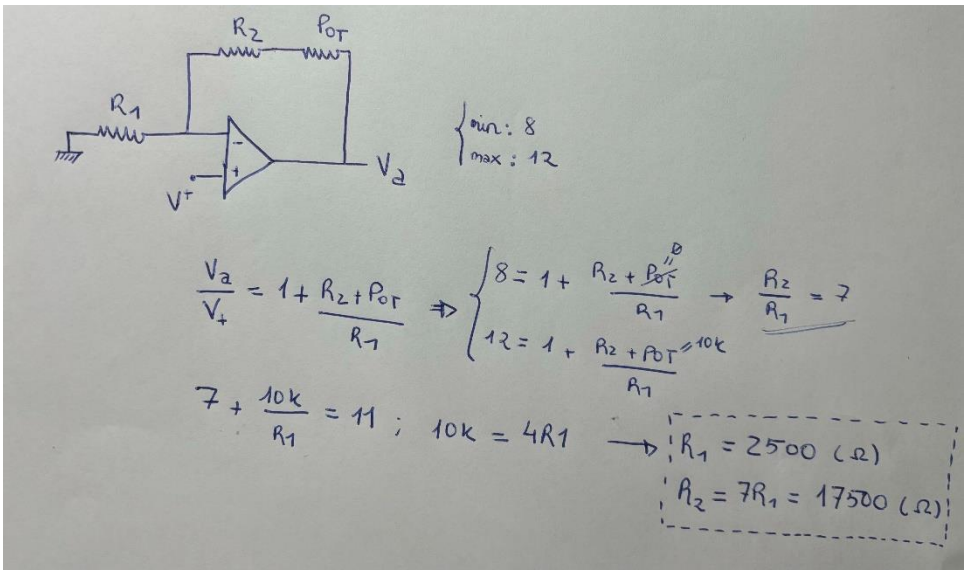
Tabla para la medida del filtro paso bajo

Frecuencia (Hz)	Entrada (Vpp)	Salida (Vpp)	Ganancia	Ganancia (dB)	ΔT (s)	Fase (º)
0,5	1	0.994	0.994	-0.052	0.0154	-2.772
1	1	0.936	0.936	-0.574	0.012	-4.32
3	1	0.912	0.912	-0.800	0.0136	-14.68
5	1	0.872	0.872	-1.189	0.0132	-23.76
7	1	0.812	0.812	-1.808	0.0128	-32.25
8	1	0.776	0.776	-2.202	0.0134	-38.59
9	1	0.744	0.744	-2.568	0.0118	-38.23
10	1	0.708	0.708	-2.999	0.0116	-41.4
11	1	0.680	0.680	-3.349	0.0108	-42.76
12	1	0.656	0.656	-3.661	0.0112	-48.38
13	1	0.624	0.624	-4.096	0.0106	-49.60
14	1	0.604	0.604	-4.379	0.0102	-51.40
15	1	0.576	0.576	-4.791	0.0101	-54.40
17	1	0.532	0.532	-5.481	0.009	-55.08
20	1	0.472	0.472	-6.521	0.0083	-59.76
30	1	0.344	0.344	-9.268	0.0062	-66.96
50	1	0.224	0.224	-12.995	0.0041	-73.8
70	1	0.168	0.168	-15.493	0.0032	-80.64
100	1	0.118	0.118	-18.562	0.0023	-82.8
200	1	0.0704	0.0704	-23.048	0.00121	-87.12
300	1	0.054	0.054	-25.352	0.000800	-86.4
500	1	0.038	0.038	-28.404	0.000480	-86.4
700	1	0.033	0.033	29.629	0.000338	-86.176
1.000	1	0.021	0.021	-33.555	0.000272	-97.92

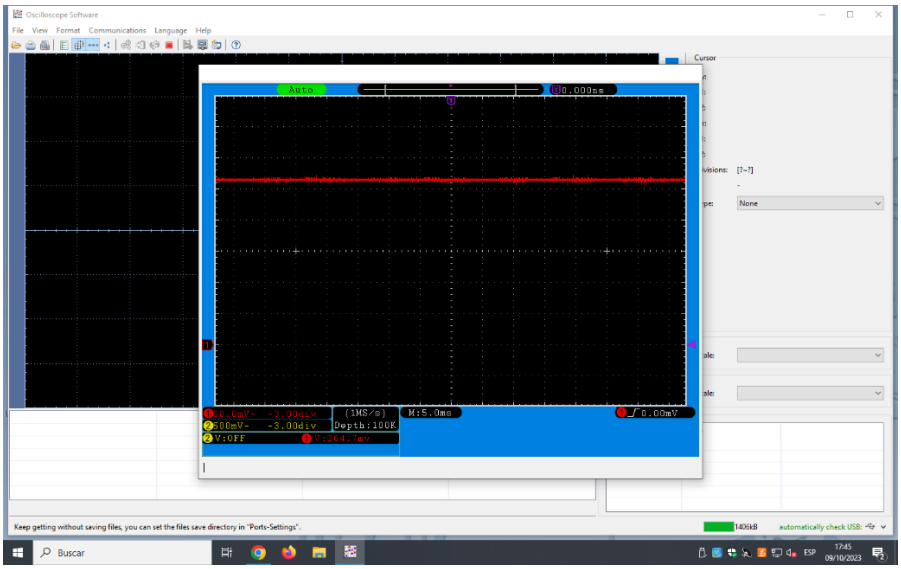
Frecuencia de corte (*fc*) exacta medida	**10 Hz**

MÓDULO 3: Amplificador

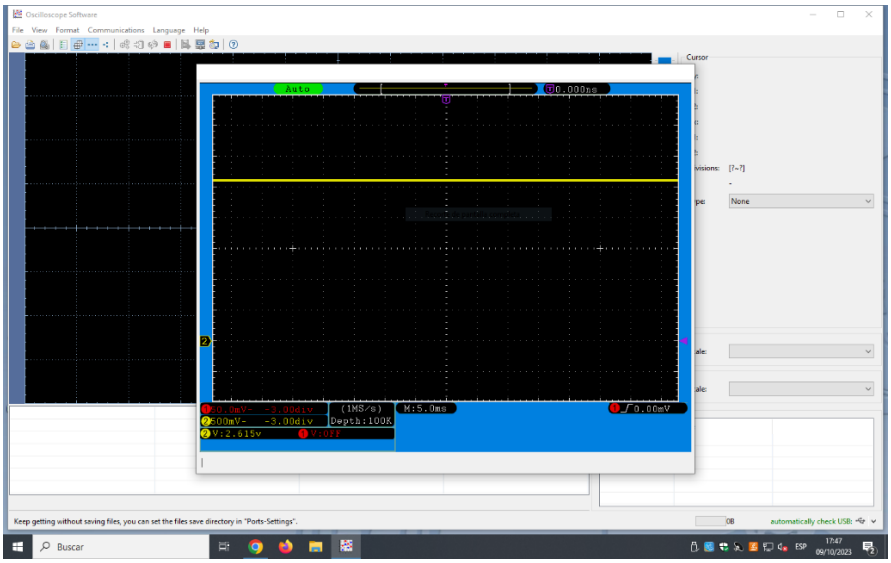
1. Detalle el cálculo de las resistencias R1 y R2.



2. Incluya una captura de pantalla del osciloscopio donde se aprecie sólo la señal a la entrada del amplificador con la escala de 50 mV/div (Puede emplear las herramientas de medida de tensión del osciloscopio. En el menú “Measure” puede hacer que aparezca el valor medio de la tensión en la pantalla del osciloscopio).



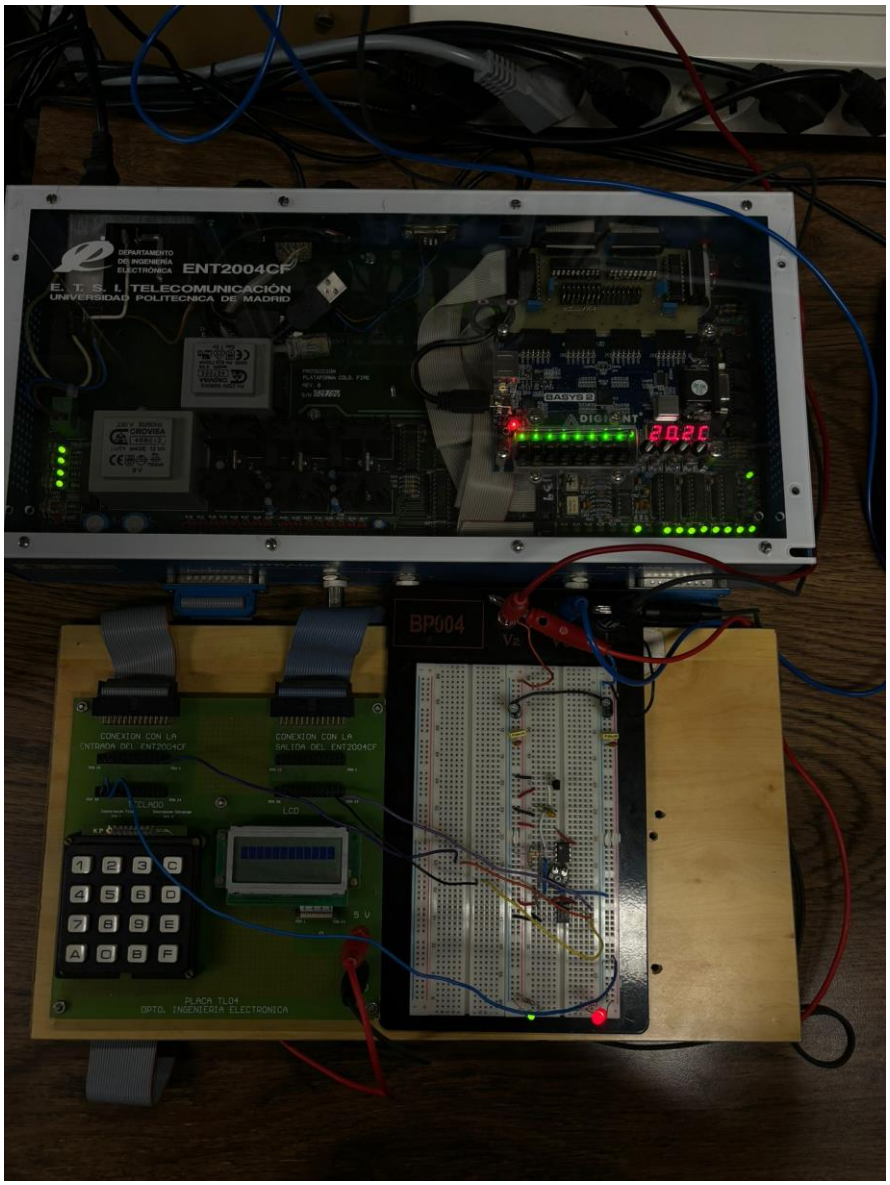
3. Incluya una captura de pantalla del osciloscopio donde se aprecie sólo la señal a la salida del amplificador con la escala de 500 mV/div (Puede emplear las herramientas de medida de tensión del osciloscopio. En el menú “Measure” puede hacer que aparezca el valor medio de la tensión en la pantalla del osciloscopio).



Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

MÓDULO 4: Convertidor analógico-digital MCP3201

1. Incluya una fotografía del circuito conectado a la FPGA donde se vean las conexiones y se lea la temperatura ambiente indicada en los displays.



2. A continuación caliente el sensor con los dedos e incluya otra fotografía donde se vean las conexiones y se lea la temperatura en los displays.


DEPARTAMENTO
DE INGENIERÍA
ELECTRÓNICA
ENT2004CF
E. T. S. I. TELECOMUNICACIÓN
UNIVERSIDAD POLITECNICA DE MADRID
BASYS 2
DIGILENT
BP004
CONEXION CON LA
ENTRADA DEL ENT2004CF
CONEXION CON LA
SALIDA DEL ENT20
PLACA TL04

MÓDULO 5: Módulo de visualización

1. Explique razonadamente la elección de los valores para los bits “XXXX” en la entrada E0 del registro.

   Se han escogido los valores de 1100 para E0 puesto que se debe representar la letra C, que es representada con dichos números en el decodificador.

2. Detalle el código VHDL de los cuatro módulos (registro.vhd, MUX4x4.vhd, refresco.vhd, visualizacion.vhd) debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

```
----------------------------------------------------------------------------
-------
--
-- Registro que almacena los datos de DECENAS, UNIDADES y DCIMAS
-- cada vez que se activa el ENABLE
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity registro is
    Port ( CLK     : in   STD_LOGIC;                          -- entrada de
reloj
              ENABLE  : in   STD_LOGIC;                          -- enable
           E0      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E0
           E1      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E1
           E2      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E2
           E3      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E3
           Q0      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q0
           Q1      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q1
           Q2      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q2
           Q3      : out  STD_LOGIC_VECTOR (3 downto 0));  -- salida Q3
end registro;

architecture a_registro of registro is

signal QS0 : STD_LOGIC_VECTOR (3 downto 0):="0000"; -- seal que almacena el
valor de Q0
signal QS1 : STD_LOGIC_VECTOR (3 downto 0):="0000"; -- seal que almacena el
valor de Q1
signal QS2 : STD_LOGIC_VECTOR (3 downto 0):="0000"; -- seal que almacena el
valor de Q2
```

```
signal QS3 : STD_LOGIC_VECTOR (3 downto 0):="0000"; -- seal que almacena el
valor de Q3

begin
  process (CLK)
    begin
      if (CLK'event and CLK='1') then   -- con cada flanco activo
                if(ENABLE='1') then    --y cuando la senal de enable esta
activada

                    -- Realizar el proceso de captura
                    QS0<=E0;
                    QS1<=E1;
                    QS2<=E2;
                    QS3<=E3;

                    end if;
             end if;
  end process;
  -- CABLEADO DE LAS SALIDAS
--Las senales van a su correspondiente salida
      Q0<=QS0;
      Q1<=QS1;
      Q2<=QS2;
      Q3<=QS3;

end a_registro;

--------------------------------------------------------------------------------
-------
-- MUX4x4

--  Multiplexor de 4 entradas de 4 bits
--
--------------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity MUX4x4 is
    Port ( E0 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 0
           E1 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 1
           E2 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 2
           E3 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 3
           S : in  STD_LOGIC_VECTOR (1 downto 0);  -- Seal de control
           Y : out  STD_LOGIC_VECTOR (3 downto 0)); -- Salida
end MUX4x4;

architecture a_MUX4x4 of MUX4x4 is

begin
```

```
    with S select Y<=

                    E0 when "00",-- Selecciona la entrada 0 cuando el bit de
control es 0
                    E1 when "01", -- Selecciona la entrada 1
                    E2 when "10", -- Selecciona la entrada 2
                    E3 when others; -- Selecciona la entrada 3 por defecto

end a_MUX4x4;

----------------------------------------------------------------------------
-------
-- refresco

-- Circuito que refresca los displays peridicamente
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;


entity refresco is
    Port ( CLK : in  STD_LOGIC;                   -- reloj de refresco
           S : out  STD_LOGIC_VECTOR (1 downto 0);   -- Control para el mux
                     DP : out STD_LOGIC;                          -- Control
del punto decimal
           AN : out   STD_LOGIC_VECTOR (3 downto 0)); -- Control displays
individuales
end refresco;

architecture a_refresco of refresco is

            signal QS : unsigned (1 downto 0):="00"; --la que va con s
            --signal PAN : unsigned (3 downto 0):="0000";
            --signal DPS : STD_LOGIC:='1';

begin

  process (CLK)
    begin
       if (CLK'event and CLK='1') then
                   QS <= QS+1;

       -- if (QS="01") then
            --     DPS <='0';
             --end if;
       end if;

       end process;
```

```
-- Completar cableado de:

            S <= STD_LOGIC_VECTOR(QS);
                  --se activan con un 0 en funcion de la senal qs
            AN<= "1110" when QS="00" else--cuando se recibe 00 se activa
el panel de la derecha
                  "1101" when QS="01" else--con un 01 el segundo de la
drcha
                  "1011" when QS="10" else--con 10 el segundo de la izqda
                  "0111" when QS="11"; --con un 11 el de la izda

            DP <= '1' when QS="00" else
                  '1' when QS="01" else
                  '0' when QS="10" else
                  --cuando se activa el segundo panel se activa el
                  --dot point (con un 0)
                  '1' when QS="11";


 end a_refresco;

----------------------------------------------------------------------------
-------
--
-- Mdulo de visualizacin, presenta los datos
-- en los displays capturndolos con la seal ENABLE.
--
----------------------------------------------------------------------------
-------

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity visualizacion is
    Port ( ENABLE : in STD_LOGIC;                              -- Entrada de
ENABLE
              DECENAS : in   STD_LOGIC_VECTOR (3 downto 0);    -- Entrada
decenas
              UNIDADES : in   STD_LOGIC_VECTOR (3 downto 0);    -- Entrada
unidades
              DECIMAS : in   STD_LOGIC_VECTOR (3 downto 0);    -- Entrada
dcimas
           CLK     : in  STD_LOGIC;                    -- Entrada de reloj
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);       -- Salida para los
displays
                     DP   : out STD_LOGIC;                         -- Salida
punto decimal
           AN     :  out   STD_LOGIC_VECTOR  (3  downto  0));  --  Activacin
individual
end visualizacion;
```

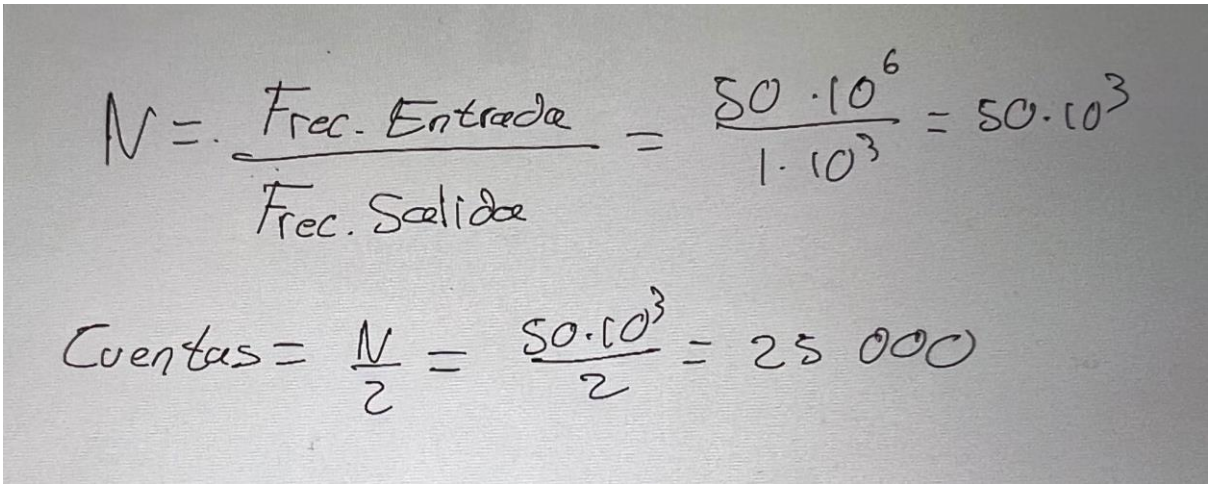
```
architecture a_visualizacion of visualizacion is

-- COMPONENTES
--se declaran todas las partes como componentes
component decodBCDa7s
  Port ( BCD : in  STD_LOGIC_VECTOR (3 downto 0);     -- Entrada del valor
BCD
         SEGMENTOS  :  out   STD_LOGIC_VECTOR  (0  to  6));   --  Salidas  al
display
end component;

component registro
  Port (   CLK     : in   STD_LOGIC;                          -- entrada de
reloj
              ENABLE  : in   STD_LOGIC;                        -- enable
           E0      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E0
           E1      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E1
           E2      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E2
           E3      : in   STD_LOGIC_VECTOR (3 downto 0);   -- entrada E3
           Q0      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q0
           Q1      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q1
           Q2      : out  STD_LOGIC_VECTOR (3 downto 0);   -- salida Q2
           Q3      : out  STD_LOGIC_VECTOR (3 downto 0));  -- salida Q3
end component;

component MUX4x4
  Port (   E0 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 0
           E1 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 1
           E2 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 2
           E3 : in  STD_LOGIC_VECTOR (3 downto 0); -- Entrada 3
           S : in  STD_LOGIC_VECTOR (1 downto 0);  -- Senal de control
           Y : out  STD_LOGIC_VECTOR (3 downto 0)); -- Salida
end component;

component refresco
  Port (   CLK : in  STD_LOGIC;                  -- reloj de refresco
           S : out  STD_LOGIC_VECTOR (1 downto 0);   -- Control para el mux
                    DP : out STD_LOGIC;                          -- Control
del punto decimal
           AN : out   STD_LOGIC_VECTOR (3 downto 0)); -- Control displays
individuales
end component;

-- SEALES

--   Definir posibles seales necesarias
             signal S1 : STD_LOGIC_VECTOR (3 downto 0):="1100"; --senal de
e0 que muestra la c
             --signal S2 : STD_LOGIC_VECTOR (3 downto 0); --senal decimas
e1
```

```
            --signal S3 : STD_LOGIC_VECTOR (3 downto 0); --senal unidades
e2
      --    signal S4 : STD_LOGIC_VECTOR (3 downto 0); --senal decenas e3
            signal S5 : STD_LOGIC_VECTOR (3 downto 0); --senal entre q0 y
e0mux
            signal S6 : STD_LOGIC_VECTOR (3 downto 0); --senal entre q1 y
e1mux
            signal S7 : STD_LOGIC_VECTOR (3 downto 0); --senal entre q2 y
e2mux
            signal S8 : STD_LOGIC_VECTOR (3 downto 0); --senal entre q3 y
e3mux
            signal S9 : STD_LOGIC_VECTOR (3 downto 0); --senal Y y BCD
            signal S10 : STD_LOGIC_VECTOR (1 downto 0); --senal S
            --signal S11 : STD_LOGIC_VECTOR (0 to 6); --senal de seg7
            --signal S12 : STD_LOGIC_VECTOR (3 downto 0); --senal de an
            --signal S13 : STD_LOGIC; --senal de dp
            --signal S14 : STD_LOGIC; --senal enable
            --signal S15 : STD_LOGIC; --senal clk



begin --se conectan las diferentes senales a las entradas
            --segun lo establecido antes

U1 : decodBCDa7s
                  port map (
                              BCD=>S9,
                              SEGMENTOS=>SEG7
                        );

U2 : registro
                  port map (
                        E0=>S1,
                        E1=>DECIMAS,
                        E2=>UNIDADES,
                        E3=>DECENAS,
                        Q0=>S5,
                        Q1=>S6,
                        Q2=>S7,
                        Q3=>S8,
                        ENABLE=>ENABLE,
                        CLK=>CLK
                        );

U3 : MUX4x4
                  port map (
                        E0=>S5,
                        E1=>S6,
                        E2=>S7,
                        E3=>S8,
                        S=>S10,
```

```
                              Y=>S9
                              );

    U4 : refresco
                        port map (
                              CLK=>CLK,
                              DP=>DP,
                              S=>S10,
                              AN=>AN
                              );


    end a_visualizacion;
```

MODULO 6: Divisor del reloj

1. Detalle el cálculo de la constante XXXX.



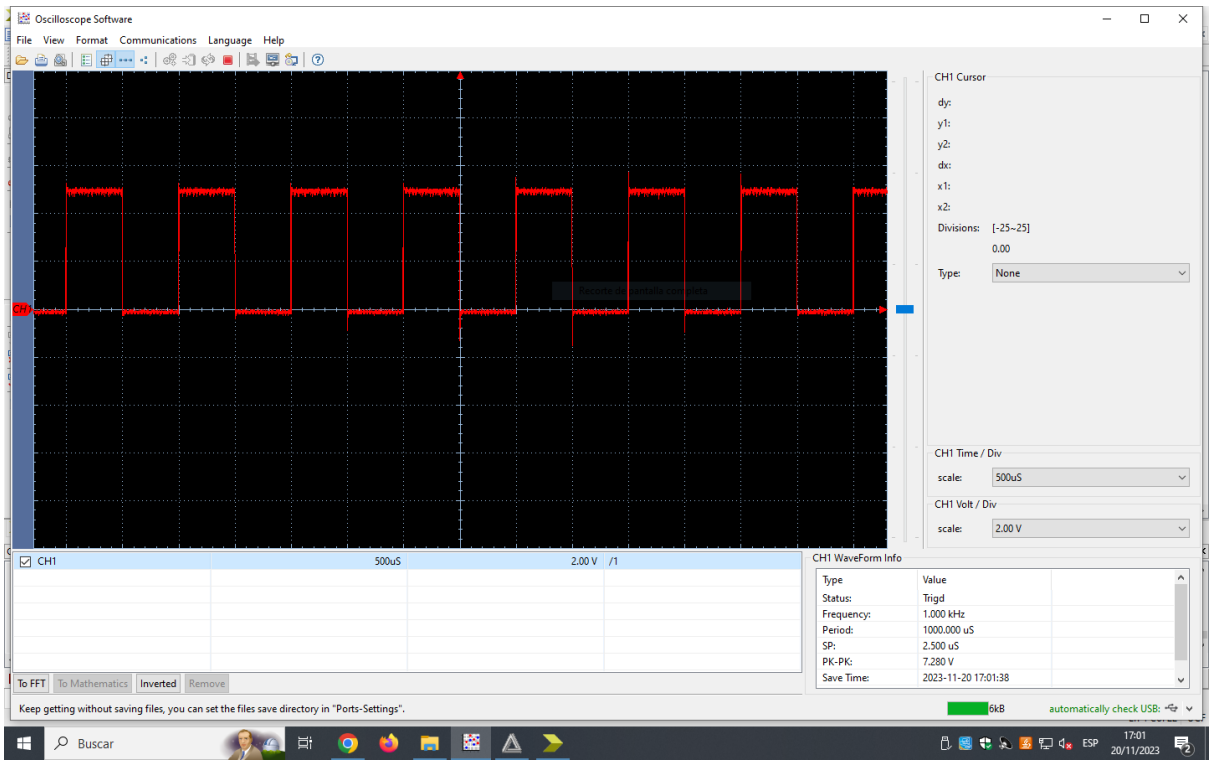
   Dividimos la frecuencia de entrada entre la de salida para hallar N, una vez hallada, realizamos la operación que nos indica el enunciado para determinar la constante XXXX, es decir, el nº de cuentas realizadas.

2. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

```
3. ----------------------------------------------------------------------------
   -------
4. --
5. -- Divisor de la frecuencia del reloj a 1 KHz.
6. --
7. ----------------------------------------------------------------------------
   -------
8. library IEEE;
9. use IEEE.STD_LOGIC_1164.ALL;
10. use IEEE.NUMERIC_STD.ALL;
11.
12. entity div_reloj is
13.     Port ( CLK_50MHz : in  STD_LOGIC;       -- Entrada reloj de la FPGA 50
   MHz
14.            CLK       : out  STD_LOGIC);     -- Salida reloj a 1 KHz
15. end div_reloj;
16.
17. architecture a_div_reloj of div_reloj is
18.
19. signal contador : unsigned (31 downto 0);
20. signal frec_div : STD_LOGIC;
21.
22. begin
23.
24. process(CLK_50MHz)
```

```
25.  begin
26.       if (CLK_50MHz'event and CLK_50MHz='1') then --En cada flanco de
   subida
27.             contador<=contador+1;                          --Incrementar el
   contador
28.              if (contador=25000) then                       -- Asignamos el
   valor de contador         tras realizar las operaciones pertinentes
29.              contador<=(others=>'0');                  -- Poner el contador
   a 0
30.             frec_div<=not frec_div;                    --e intercambiar el
   valor de frec_div
31.              end if;
32.        end if;
33.  end process;
34. CLK<=frec_div;
35. end a_div_reloj;
```

36. Incluya la captura de pantalla del osciloscopio donde pueda observarse la señal de reloj de 1 kHz con niveles de 0 y 5V.



Las capturas de pantalla del osciloscopio deben contener la información suficiente para poder deducir la amplitud de las señales y las magnitudes en el eje de tiempos. Para ello debe incluir las referencias de los ejes (Voltios por división y tiempo por división).

MÓDULO 7: Autómata de control

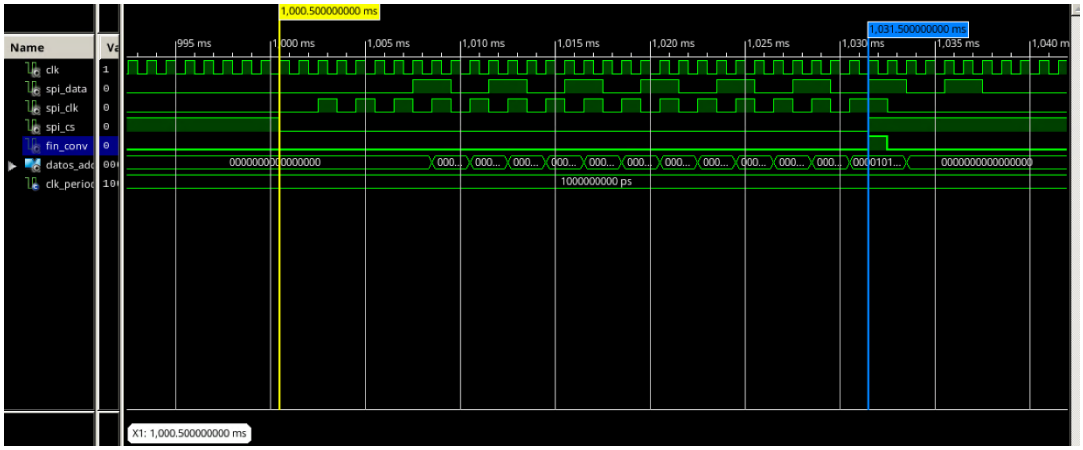
1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

```
----------------------------------------------------------------------------
-------
--
-- Autmata de control del receptor
-- Establece la comunicacin SPI con el MCP3201 y obtiene la muestra leda
--
----------------------------------------------------------------------------
-------

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

entity aut_control is
    Port ( CLK        : in  STD_LOGIC;                          -- reloj del
sistema
           SPI_DATA   : in  STD_LOGIC;                            -- entrada
de datos del puerto SPI
  SPI_CLK    : out  STD_LOGIC;                           -- Salida de reloj
del puerto SPI
           SPI_CS     : out  STD_LOGIC;                             -- Chip
Select del puerto SPI
  FIN_CONV   : out  STD_LOGIC;                          -- Fin de conversin
A/D
  DATOS_ADC  : out  STD_LOGIC_VECTOR (15 downto 0));  -- Dato ledo del ADC
end aut_control;

architecture a_aut_control  of aut_control is

type STATE_TYPE is (ESPERA_1S,ACTIVAR_CS,CLK1,CLK0,DESACT_CS);--el automata
tiene los siguientes estados

signal ST : STATE_TYPE := ESPERA_1S;
signal  cont  :  unsigned  (15  downto  0):=(others=>'0');--senal  para  el
contador
signal data : unsigned (15 downto 0):=(others=>'0');

begin

process (CLK)
  begin
  if (CLK'event and CLK='1') then
    case ST is

--vemos como se comporta el automata en funcion del estado en el que esta
```

```
      when ACTIVAR_CS =>
  cont<=(others=>'0'); --el contador pasa a 0
  ST<=CLK0;        -- cambia el estado a clk0 siempre

-- OTROS CASOS
when CLK0 =>
  --cont<=(others=>'0');
  data(15 downto 0)<=data(14 downto 0) & SPI_DATA; --se desplazan y anaden
los datos
  cont<=cont+1; --se suma 1 al contador
  ST<=CLK1;        --pasa de estado a clk1

when CLK1 =>
  if (cont>=15) then -- cuando el contador llega a 15
ST<=DESACT_CS;     --el estado pasa a ser desac
  else
ST<=CLK0;          --y sino vuelve a clk0
  end if;

when DESACT_CS =>
  cont<=(others=>'0'); --el contador pasa a 0 y
  ST<=ESPERA_1S;                --el estado pasa a espera

when ESPERA_1S =>
  cont<=cont+1;    --en este estado se suma 1 al contador
  data<=(others=>'0');    --y se vacian los datos

  if(cont>=1000) then--si el contador es mayor de 1000
ST<=ACTIVAR_CS;    --el estado pasa a activar
  else
ST<=ESPERA_1S;--y si no llega a 1000 se mantiene en espera
  end if;

    end case;
  end if;
  end process;

-- PARTE COMBINACIONAL

SPI_CS <= '0' when (ST=ACTIVAR_CS or ST=CLK0 or ST=CLK1)--la salida de spi
vale 0 en estos estados
else '1'; --y 1 en el resto
DATOS_ADC <= STD_LOGIC_VECTOR  (data) ;--salen los datos provenientes de
data
FIN_CONV<= '1' when ST=DESACT_CS --solo vale 1 durante el estado de desact
    else '0';   --y vale 0 en el resto de estados

SPI_CLK<= '1' when (ST=CLK1 or ST=DESACT_CS)--la salida es 1 en los estados
de clk1 y desact
  else '0';            --y 0 en el resto
```

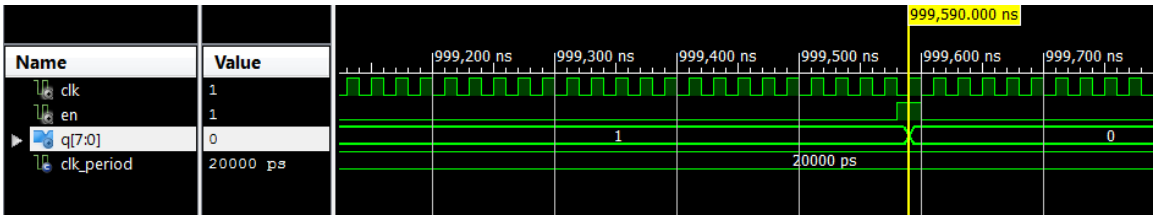
```
end a_aut_control;
```

2. Incluya una captura de pantalla de la simulación donde pueda verse con precisión el tiempo entre 995 y 1035 ms y donde puedan apreciarse las transiciones de las señales de salida. Coloque dos cursores de tiempo: uno en el flanco de bajada de la salida SPI_CS y otro en el flanco de subida de la salida SPI_CS.



Tenga en cuenta que debido al fondo negro de la pantalla del simulador es difícil que se vean las señales adecuadamente. Se recomienda utilizar la herramienta de recorte de Word y ampliar convenientemente la captura de pantalla para resaltar las regiones importantes.

La siguiente figura es un ejemplo de cómo se deben entregar las gráficas de simulación.

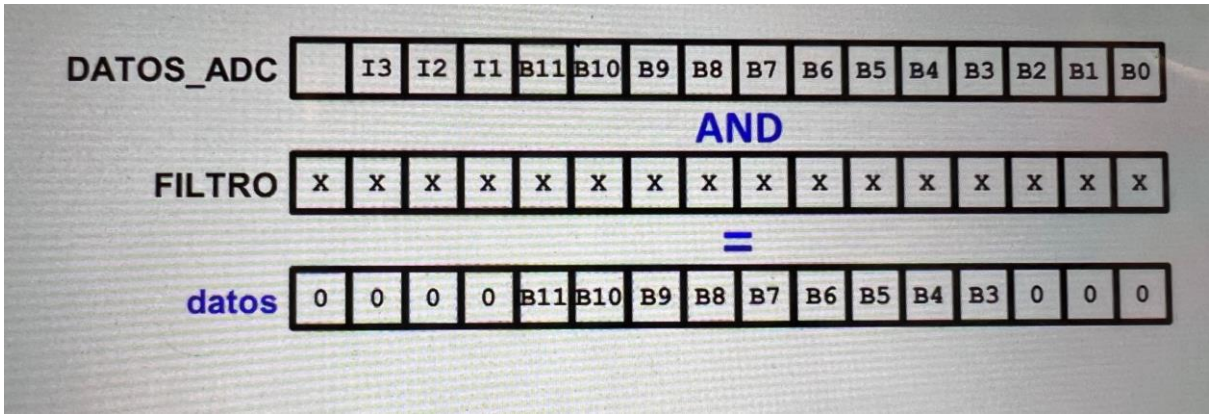


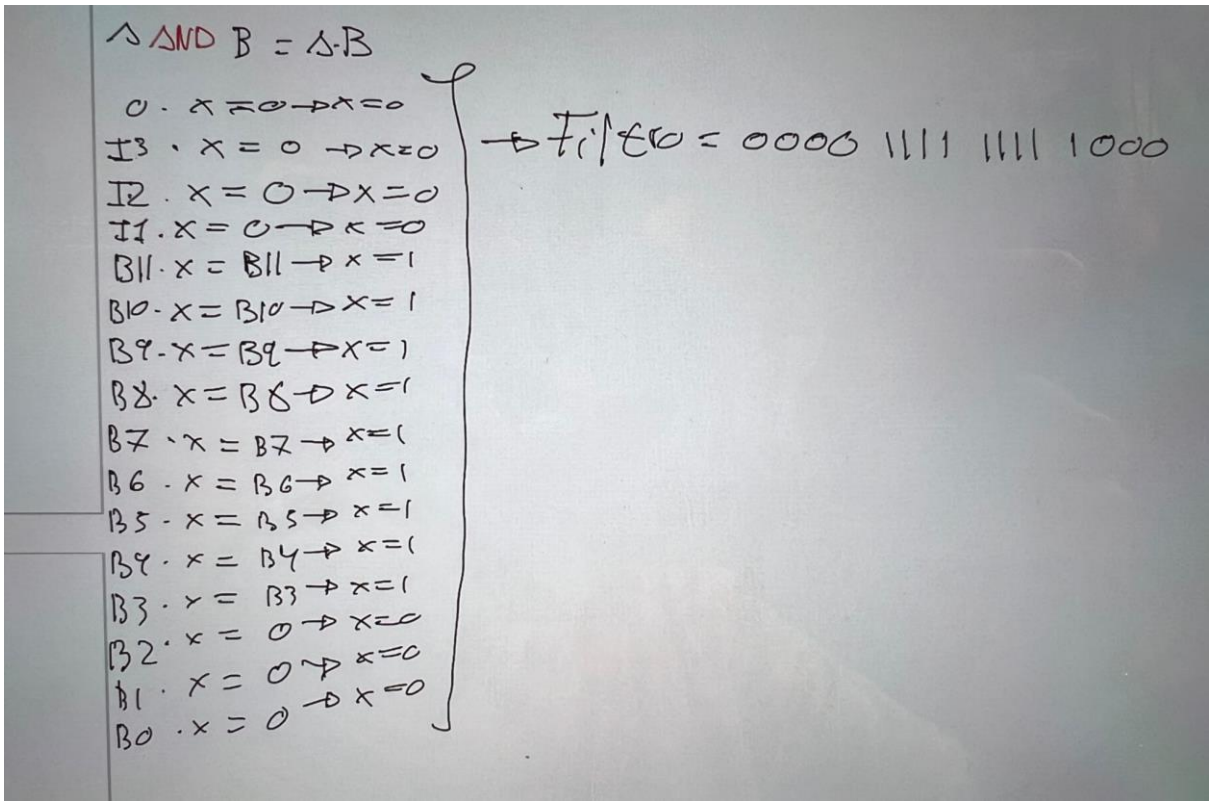
Ejemplo de gráfica de simulación con las lecturas visibles

Intente realizar la memoria de forma creativa, puede emplear diferentes colores para las diferentes señales y varios indicadores de tiempo. El objetivo es conseguir gráficas donde se aprecien con detalle los instantes temporales que se indican en cada señal.

MÓDULO 8: ADC a TEMP

1. Razone el valor elegido para la constante FILTRO.





Tras observar la ecuación planteada en el enunciado y utilizar la operación de la puerta AND, es fácil determinar el valor del filtro (constante) haciendo un simple proceso de despeje de cada incógnita (en este caso hemos cogido todas como x).

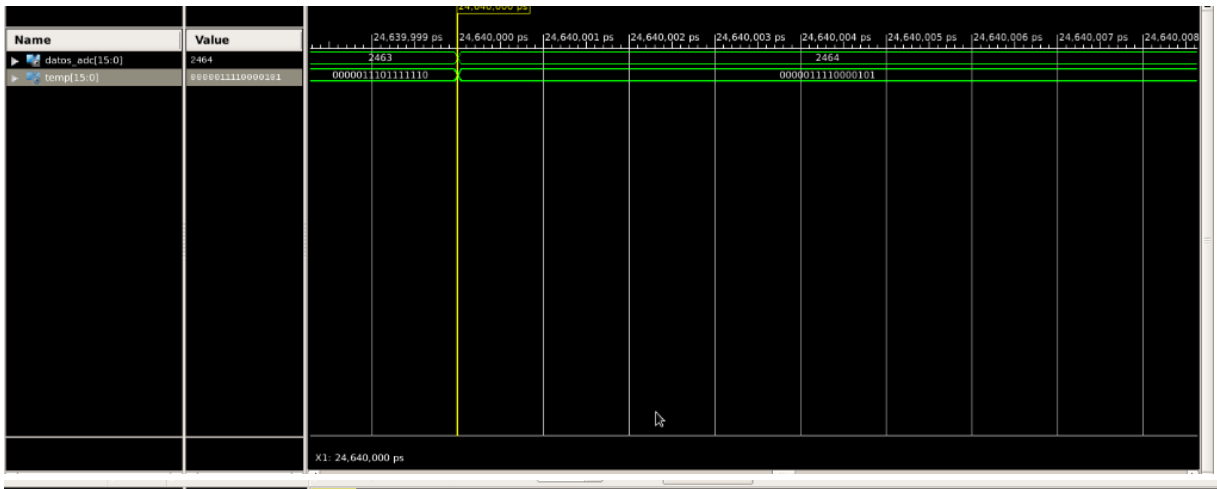
Una vez hecho el despeje como se puede observar en la fotografía adjuntada, podemos juntar los resultados y determinar el valor de la constante FILTRO, siendo esta = 0000111111111000

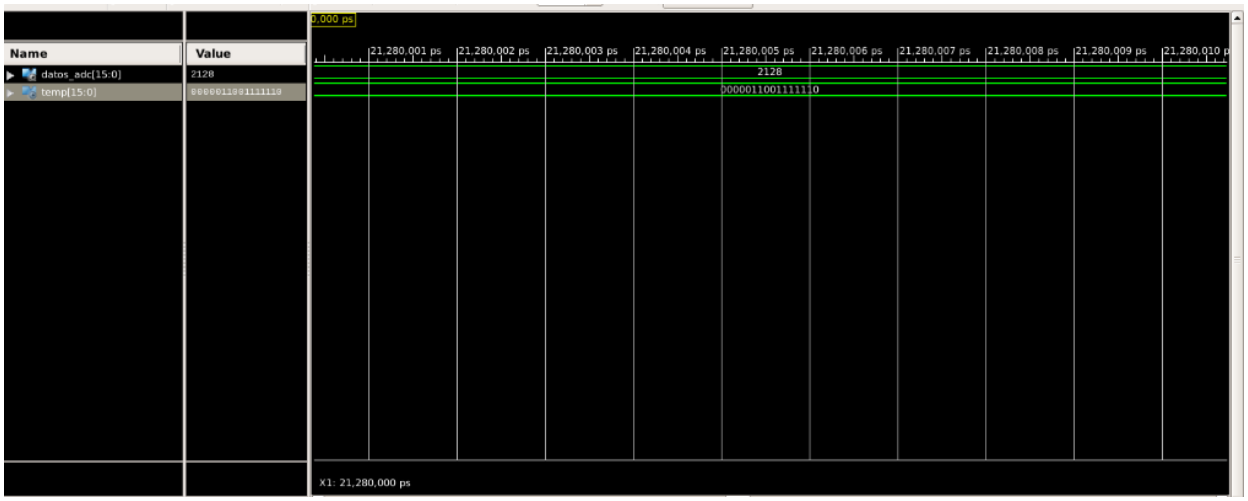
2. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

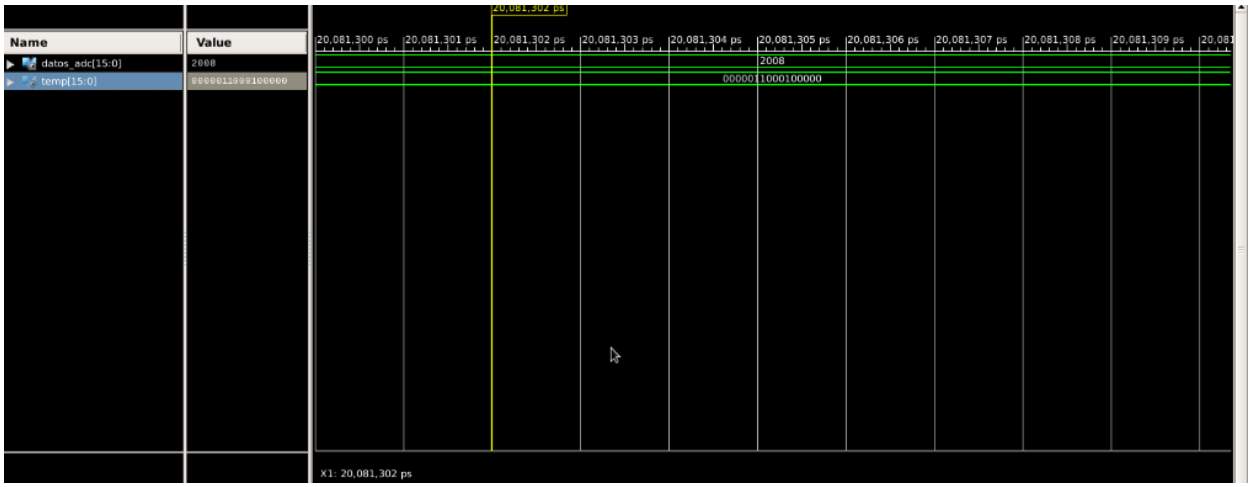
```
----------------------------------------------------------------------------
-------
-- Conversin de datos del ADC en temperatura
-- El sensor proporciona una tensin de 100 mV por C
----------------------------------------------------------------------------
-------

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

entity ADC_a_TEMP is
    Port ( DATOS_ADC : in  STD_LOGIC_VECTOR (15 downto 0);     -- Datos del
convertidor A/D
           TEMP      : out  STD_LOGIC_VECTOR (15 downto 0));   -- Salida
temperatura en punto fijo con 6 bits decimales
end ADC_a_TEMP;

architecture a_ADC_a_TEMP of ADC_a_TEMP is

constant FILTRO   : unsigned (15 downto 0):="0000111111111000";--Tras
realizar la operación pertinente determinamos este valor del filtro
signal datos      : unsigned (15 downto 0):=(others=>'0'); -- Datos ledos
del conversor y filtrados
signal datosx32   : unsigned (31 downto 0):=(others=>'0');  --  Datos
multiplicados por 32
signal datosx16   : unsigned (31 downto 0):=(others=>'0');  --  Datos
multiplicados por 16
signal datosx2    : unsigned (31 downto 0):=(others=>'0');  --  Datos
multiplicados por 2
signal datosx50   : unsigned (31 downto 0):=(others=>'0');  --  Datos
multiplicados por 50

begin

datos<= unsigned(DATOS_ADC)  and FILTRO;  -- datos de entrada filtrados

-- Para pasar a C se trata de multiplicar por 50 y dividir entre 4096

-- Para multiplicar datos*50, hacemos: datos*32+datos*16+datos*2

datosx32 <= "00000000000" & datos & "00000" ; -- Usamos esta operación para
concatenar el nº de ceros que necesitemos por cada lado, en este caso 5 por
la derecha (2^5 = 32) y 11 por la izquierda para completar los 16
```

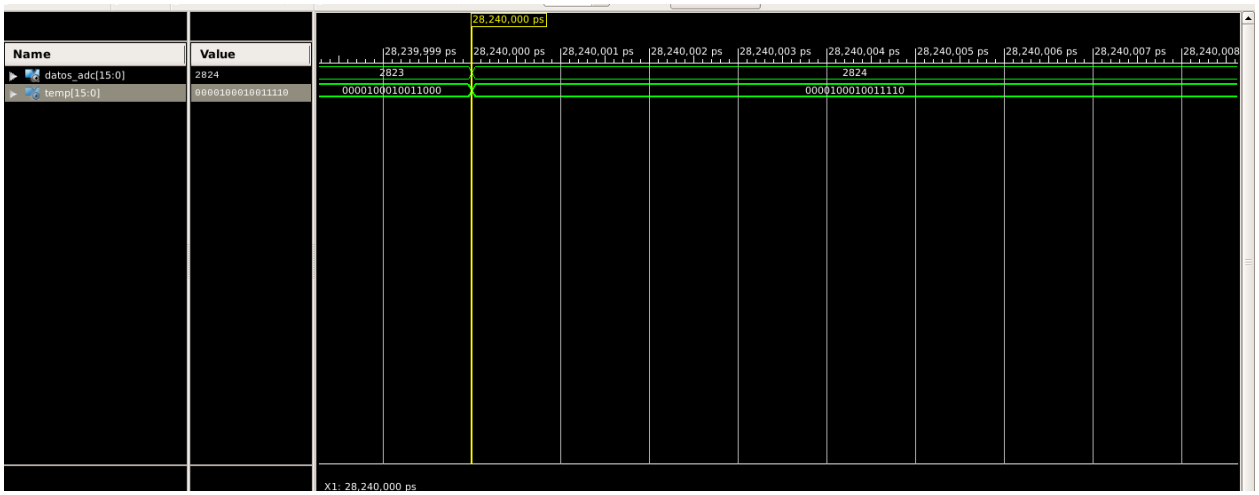
```
datosx16 <= "000000000000" & datos & "0000" ; -
datosx2 <= "000000000000000" & datos & "0" ;

datosx50<= datosx32 + datosx16 + datosx2;--Realizamos la operación que nos
indica el enunciado

-- Ahora tomamos los bits correspondientes a 10 enteros y 6 decimales

TEMP<=STD_LOGIC_VECTOR(datosx50(21 downto 6));--Convertimos la señal y
cogemos los bits que nos interesan, siguiendo lo que nos dice el enunciado

end a_ADC_a_TEMP;
```

3. Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.






Name
Value
datos_adc[15:0]
2008
temp[15:0]
0000011000100000
20,081,300 ps
20,081,301 ps
20,081,302 ps
20,081,303 ps
20,081,304 ps
20,081,305 ps
20,081,306 ps
20,081,307 ps
20,081,308 ps
20,081,309 ps
2008
0000011000100000
X1: 20,081,302 ps


Name
Value
datos_adc[15:0]
2824
temp[15:0]
0000100010011110
28,240,000 ps
28,239,999 ps
28,240,000 ps
28,240,001 ps
28,240,002 ps
28,240,003 ps
28,240,004 ps
28,240,005 ps
28,240,006 ps
28,240,007 ps
28,240,008
2823
2824
0000100010011000
0000100010011110
X1: 28,240,000 ps

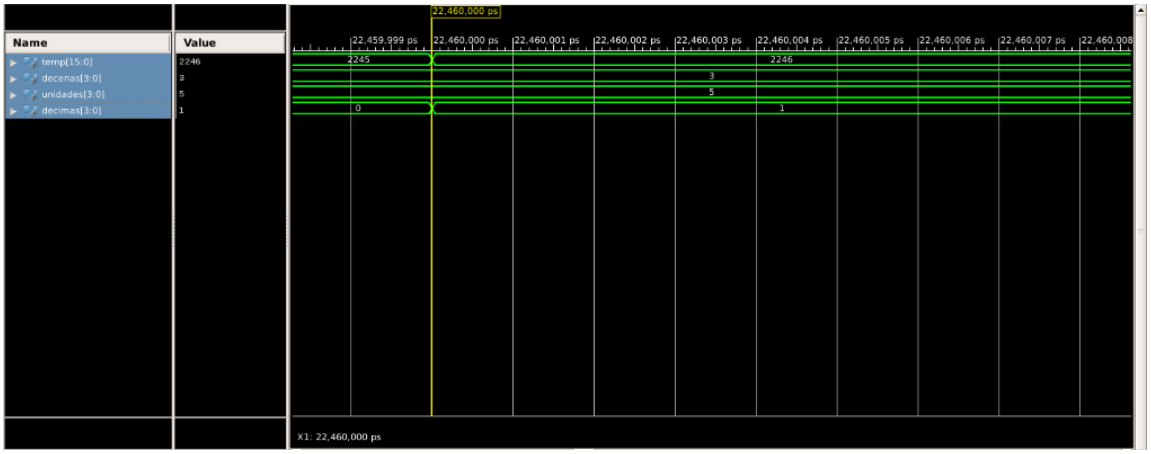
MODULO 9: TEMP a BCD

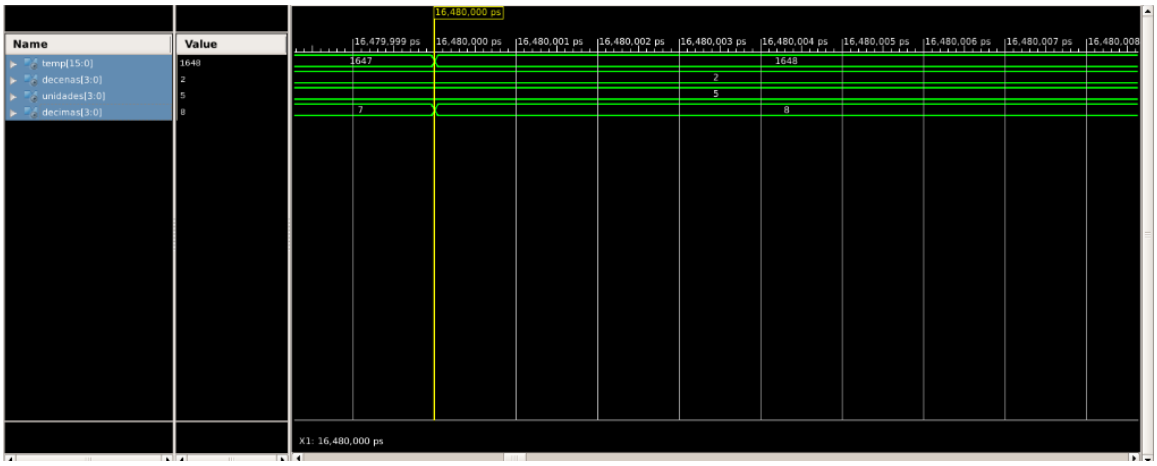
1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes, no se evalúa el código sin comentarios.**

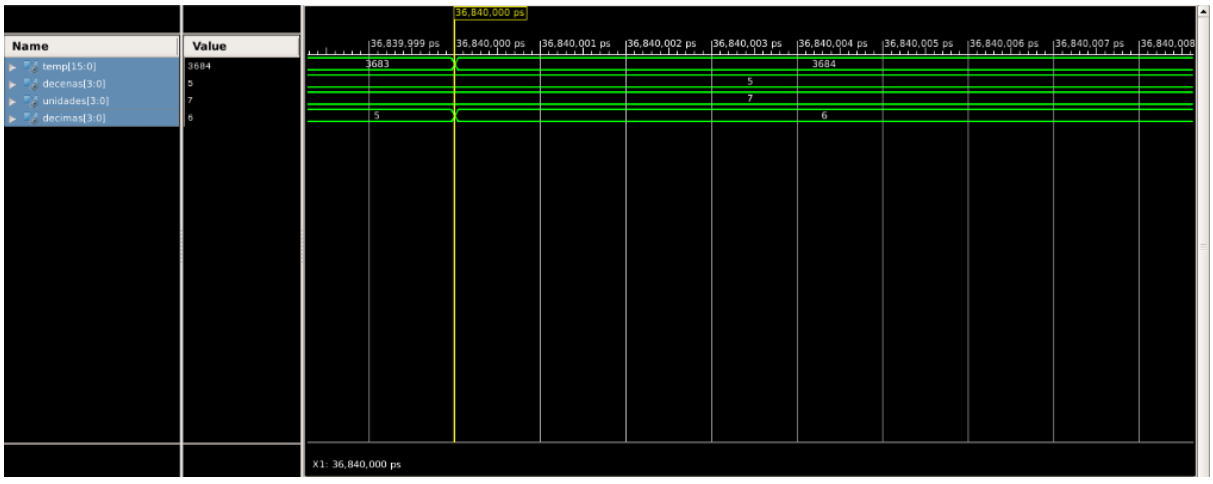
```
----------------------------------------------------------------------------
-------
-- Convierte 16 bits en punto fijo con 10 enteros y 6 decimales en BCD
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity TEMP_a_BCD is
    Port ( TEMP     : in  STD_LOGIC_VECTOR (15 downto 0);  -- Temperatura
con 6 bits decimales
           DECENAS  : out  STD_LOGIC_VECTOR (3 downto 0);  -- Decenas en
BCD
           UNIDADES : out  STD_LOGIC_VECTOR (3 downto 0);  -- Unidades en
BCD
          DECIMAS  : out  STD_LOGIC_VECTOR (3 downto 0)); -- Dcimas en BCD
end TEMP_a_BCD;

architecture a_TEMP_a_BCD of TEMP_a_BCD is

signal ENT : unsigned (9 downto 0):="0000000000";  -- Parte entera de la
temperatura
signal DEC : unsigned (5 downto 0):="000000";      -- Parte decimal de la
temperatura
signal s_unid : unsigned (9 downto 0):="0000000000";        -- unidades de
la temperatura

begin

-- Separamos parte entera y parte decimal
 ENT <= unsigned(STD_LOGIC_VECTOR(TEMP(15 downto 6)));--Cogemos los bits
que nos interesan de la señal TEMP y la transformamos en tipo unsigned
 DEC          <=          unsigned(STD_LOGIC_VECTOR(TEMP(5          downto
0)));
--Para asignar los valores que nos interesan seguimos las tablas
proporcionadas en el enunciado para determinar los intervalos
--requeridos, para definir las memorias ROM usamos la estructura when else
estipulada en el enunciado, siguiendo la misma metodología para las demás
ROM

    DECENAS <=
                "1001" when ENT >= 90 else
                "1000" when ENT >= 80 else
                "0111" when ENT >= 70 else
```

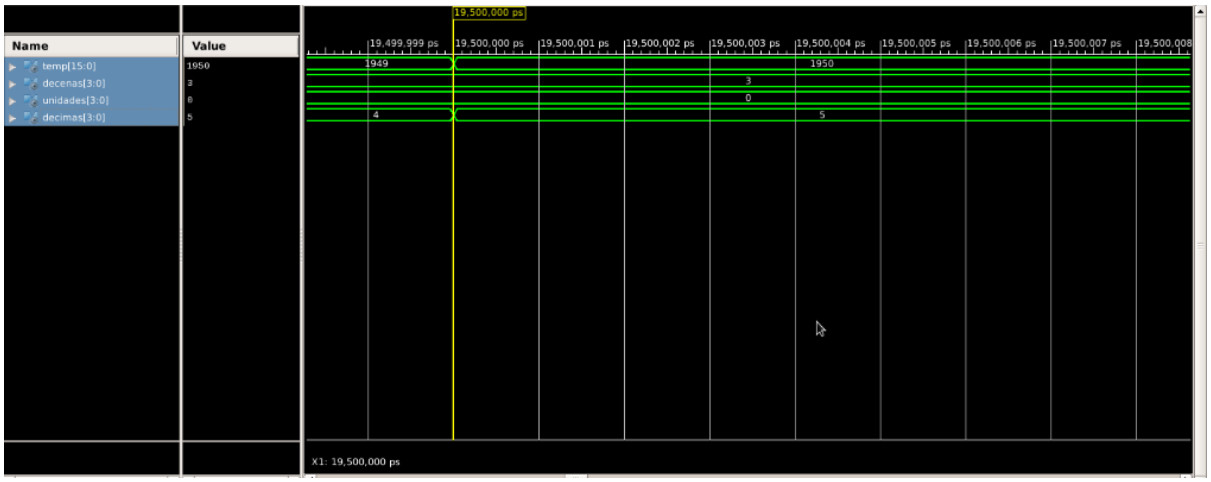
```
                "0110" when ENT >= 60 else
                "0101" when ENT >= 50 else
                "0100" when ENT >= 40 else
                "0011" when ENT >= 30 else
                "0010" when ENT >= 20 else
                "0001" when ENT >= 10 else
                 "0000"; -- Valor por defecto en caso de rango no esperado
/// others => 0 ;

-- Las unidades se determinan restando las decenas al valor de la
temperatura.

    s_unid <=
            ENT - 90 when ENT >= 90 else
            ENT -   80 when ENT >= 80 else
            ENT -   70 when ENT >= 70 else
            ENT -   60 when ENT >= 60 else
            ENT -   50 when ENT >= 50 else
            ENT -   40 when ENT >= 40 else
            ENT -   30 when ENT >= 30 else
            ENT -   20 when ENT >= 20 else
            ENT -   10 when ENT >= 10 else
            ENT;


    DECIMAS <=
                "0000" when DEC < 6 else
                "0001" when DEC < 12 else
                "0010" when DEC < 18 else
                "0011" when DEC < 24 else
                "0100" when DEC < 30 else
                "0101" when DEC < 36 else
                "0110" when DEC < 42 else
                "0111" when DEC < 48 else
                "1000" when DEC < 54 else
                "1001" ;

UNIDADES<= STD_LOGIC_VECTOR(s_unid( 3 downto 0));  --Transformamos la señal
y nos quedamos con los bits que nos interesan, en este caso, los 4 bits
menos significativos de s_unid

end a_TEMP_a_BCD;
```

2. Adjunte 4 capturas de pantalla de la simulación donde puedan apreciarse los tiempos y valores indicados en la tabla.








Name
Value
temp[15:0]
1950
decenas[3:0]
3
unidades[3:0]
0
decimas[3:0]
5
19,500,000 ps
19,499,999 ps
19,500,000 ps
19,500,001 ps
19,500,002 ps
19,500,003 ps
19,500,004 ps
19,500,005 ps
19,500,006 ps
19,500,007 ps
19,500,008
1949
1950
3
0
4
5
X1: 19,500,000 ps

MÓDULO 10: Circuito digital completo

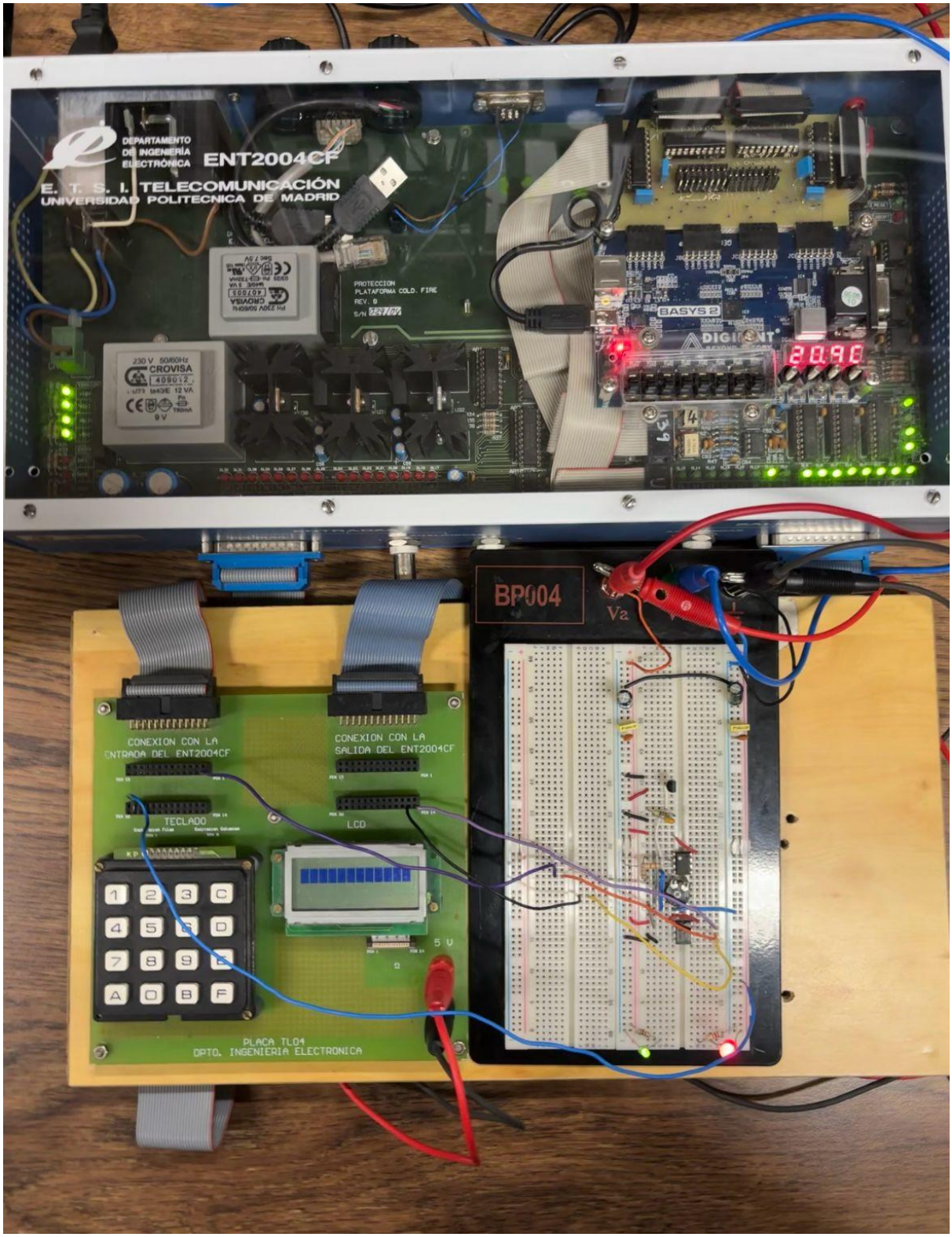
1. Detalle el código VHDL debidamente comentado. **Los comentarios son muy importantes. No se evalúa el código si no está comentado.**

```
----------------------------------------------------------------------------
-------
-- Termmetro basado en el LM35 y ADC MCP3201
--
----------------------------------------------------------------------------
-------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity termometro is
    Port ( CLK_50MHz : in  STD_LOGIC;               -- Reloj del sistema
           SPI_DATA : in  STD_LOGIC;                -- Entrada de datos del
puerto SPI
                     SPI_CLK : out  STD_LOGIC;                -- Salida de
reloj del puerto SPI
           SPI_CS : out  STD_LOGIC;                   -- Salida chip select
del puerto SPI
           AN : out  STD_LOGIC_VECTOR (3 downto 0); -- Salida de seleccin
de los displays
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);    -- Salida para los
segmentos de los displays
           DP : out  STD_LOGIC);                    -- Salida para el punto
decimal de los displays
end termometro;

architecture a_termometro of termometro is

component div_reloj is
    Port ( CLK_50MHz : in  STD_LOGIC;       -- Entrada reloj de la FPGA 50
MHz
           CLK       : out  STD_LOGIC);     -- Salida reloj a 1 KHz
end component;

   -- OTROS COMPONENTES

component aut_control is
    Port ( CLK        : in  STD_LOGIC;                         -- reloj del
sistema
           SPI_DATA   : in  STD_LOGIC;                           -- entrada
de datos del puerto SPI
                     SPI_CLK    : out  STD_LOGIC;                       --
Salida de reloj del puerto SPI
```

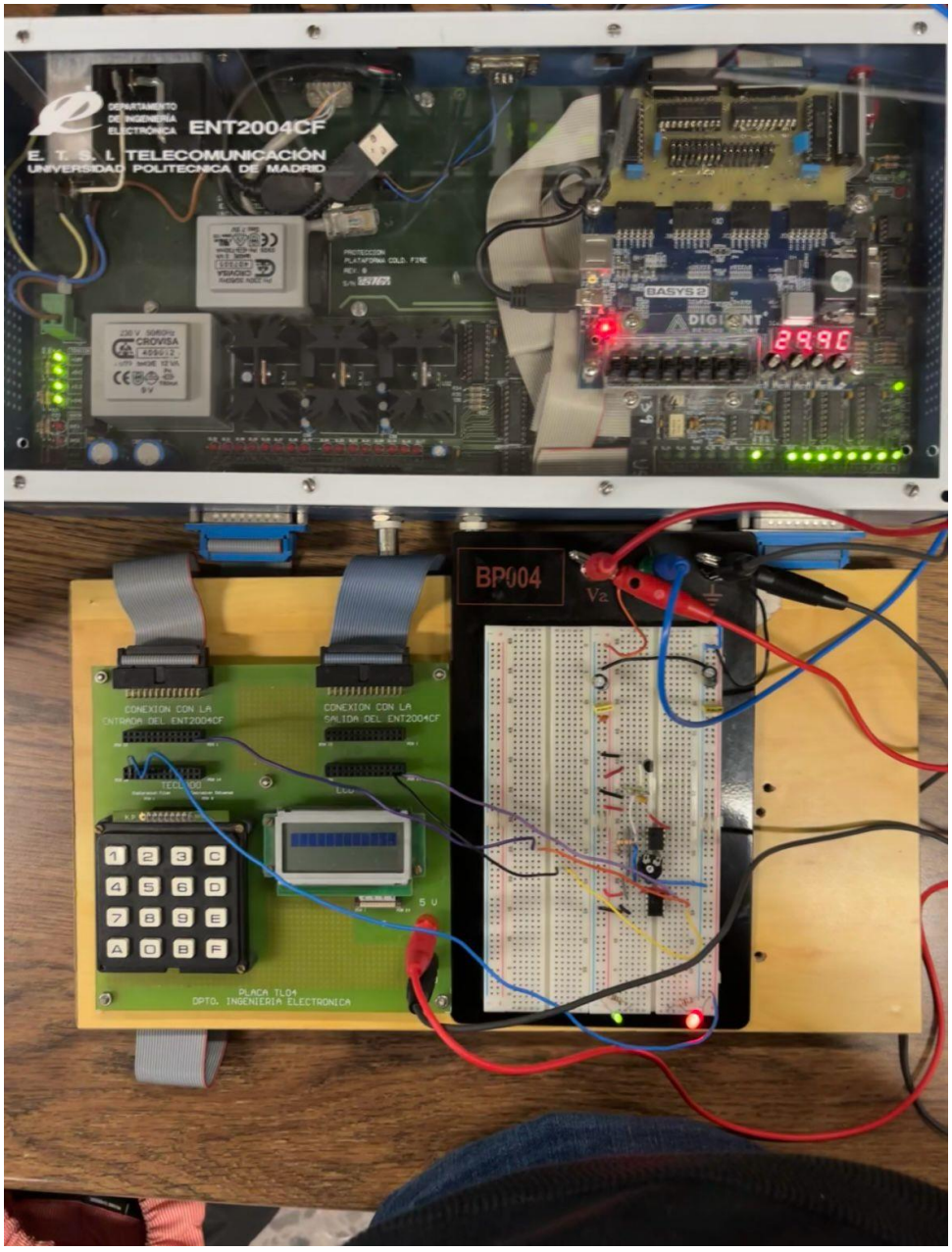
```
           SPI_CS      : out   STD_LOGIC;                              -- Chip
Select del puerto SPI
                    FIN_CONV   : out  STD_LOGIC;                        --
Fin de conversin A/D
                    DATOS_ADC  : out  STD_LOGIC_VECTOR (15 downto 0));  --
Dato ledo del ADC
end component;

component ADC_a_TEMP is
    Port ( DATOS_ADC : in  STD_LOGIC_VECTOR (15 downto 0);      -- Datos del
convertidor A/D
           TEMP       : out  STD_LOGIC_VECTOR (15 downto 0));    -- Salida
temperatura en punto fijo con 6 bits decimales
end component;

component TEMP_a_BCD is
    Port ( TEMP      : in  STD_LOGIC_VECTOR (15 downto 0);  -- Temperatura
con 6 bits decimales
           DECENAS   : out   STD_LOGIC_VECTOR (3 downto 0);   -- Decenas en
BCD
           UNIDADES : out   STD_LOGIC_VECTOR (3 downto 0);   -- Unidades en
BCD
           DECIMAS  : out  STD_LOGIC_VECTOR (3 downto 0)); -- Dcimas en BCD
end component;

component visualizacion is
    Port ( ENABLE : in STD_LOGIC;                              -- Entrada de
ENABLE
              DECENAS : in   STD_LOGIC_VECTOR (3 downto 0);     -- Entrada
decenas
              UNIDADES : in   STD_LOGIC_VECTOR (3 downto 0);    -- Entrada
unidades
              DECIMAS : in   STD_LOGIC_VECTOR (3 downto 0);     -- Entrada
dcimas
           CLK     : in  STD_LOGIC;                    -- Entrada de reloj
           SEG7 : out  STD_LOGIC_VECTOR (0 to 6);       -- Salida para los
displays
                    DP   : out STD_LOGIC;                          -- Salida
punto decimal
           AN     :  out    STD_LOGIC_VECTOR  (3  downto  0));  --  Activacin
individual
end component;

      -- POSIBLES SEALES NECESARIAS
signal C0 : STD_LOGIC;
signal  T0  :  STD_LOGIC_VECTOR(15  downto  0);--señal  que  conecta  ADC_a_TEMP
con TEMP_a_BCD
signal T1 : STD_LOGIC_VECTOR(3 downto 0);--señal que conecta DECENAS con E3
signal T2 : STD_LOGIC_VECTOR(3 downto 0);--Señal que conecta UNIDADES con
E2
signal T3 : STD_LOGIC_VECTOR(3 downto 0);--Señal que conecta DECIMAS con E1
```

```
signal A2 : STD_LOGIC_VECTOR(15 downto 0);--Señal que conecta DATOS_ADC del
autómata con ADC_a_TEMP
signal A1 : STD_LOGIC;--FIN_CONV con ENABLE

begin

U1 : div_reloj
      port map (CLK_50MHz => CLK_50MHz,
                           CLK => C0);

U2 : aut_control
      port map(SPI_CS => SPI_CS,
                           SPI_CLK => SPI_CLK,
                           SPI_DATA => SPI_DATA,
                           FIN_CONV => A1,
                           DATOS_ADC => A2,
                           CLK => C0);

U3 : ADC_a_TEMP
      port map (DATOS_ADC =>A2 ,
                           TEMP => T0);

U4 : TEMP_a_BCD
      port map (TEMP => T0,
                           DECENAS => T1,
                           UNIDADES => T2,
                           DECIMAS => T3);

U5 : Visualizacion
      port map (ENABLE => A1,
                           DECENAS => T1,
                           UNIDADES => T2,
                           DECIMAS => T3,
                           CLK => C0,
                           SEG7 => SEG7,
                           AN => AN,
                           DP => DP);


   -- OTRAS INTERCONEXIONES

end a_termometro;
```

2. Adjunte una fotografía del circuito conectado a la FPGA y donde se lea correctamente la temperatura ambiente.



3. A continuación caliente el sensor con los dedos y adjunte otra fotografía donde se vea el circuito conectado a la FPGA y se lea correctamente la temperatura.



4. Incluya el código del archivo de asociaciones empleado para la síntesis.

```
# Reloj principal del sistema
NET "CLK_50MHz" LOC = "M6"; # Se�al de reloj del sistema de 50 MHz
```

```
# Conexiones de los DISPLAYS
NET "SEG7<0>" LOC = "L14"; # se�al = CA
NET "SEG7<1>" LOC = "H12"; # Se�al = CB
NET "SEG7<2>" LOC = "N14"; # Se�al = CC
NET "SEG7<3>" LOC = "N11"; # Se�al = CD
NET "SEG7<4>" LOC = "P12"; # Se�al = CE
NET "SEG7<5>" LOC = "L13"; # Se�al = CF
NET "SEG7<6>" LOC = "M12"; # Se�al = CG
NET "DP" LOC = "N13"; # Punto decimal

# Se�ales de activaci�n de los displays
NET "AN<0>" LOC = "F12"; # Activaci�n del display 0 = AN0
NET "AN<1>" LOC = "J12"; # Activaci�n del display 1 = AN1
NET "AN<2>" LOC = "M13"; # Activaci�n del display 2 = AN2
NET "AN<3>" LOC = "K14"; # Activaci�n del display 3 = AN3

#Entradas y salidas
NET "SPI_DATA" LOC="B2"; # Entrada de datos
NET "SPI_CS" LOC="B9";  # Salida de Chip Select
NET "SPI_CLK" LOC="A9";   # Salida de reloj
```