# Anexos

# Anexo 1.- Código fuente

Como se dijo antes el código C está compuesto por tres archivos, que se añaden a continuación. Al realizar la compilación se tiene el resultado que se muestra en la figura:

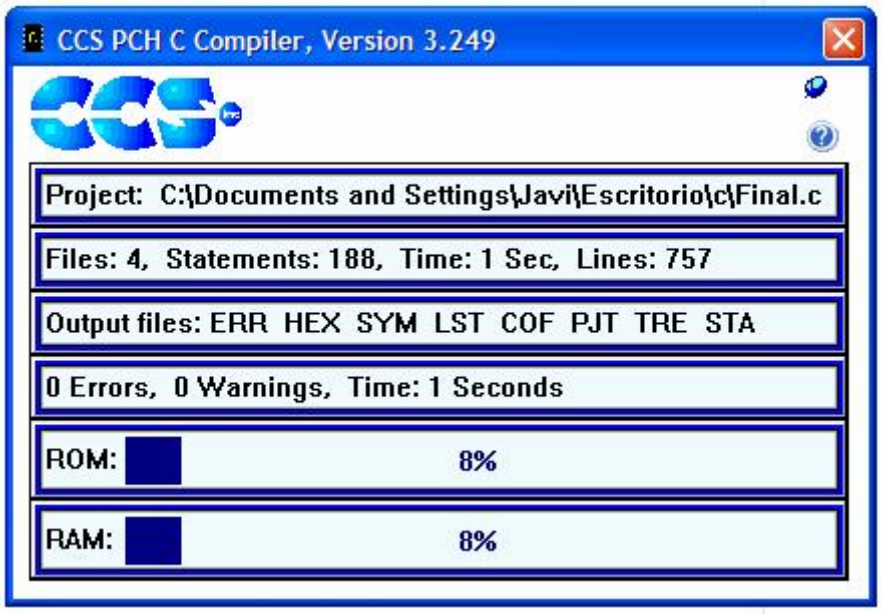


**Ilustración 87: Resultado de la compilación**

```
*************************************************************************************
*       Archivo Final.c                                                             *
*************************************************************************************

#include "C:\Documents and Settings\Javi\Escritorio\Final.h"
#include "C:\Documents and Settings\Javi\Escritorio\constantes.h"


int i;           // Declaracion de todas las variables
int j=0;         // Seran visibles en todo el programa
int m=0;
int m2=0;
int1 a, b;
signed int dif01=0;
signed int dif23=0;
int16 med01=0;
int16 med23=0;
int16 media=0;
int16 media2=0;
int16 Amax, Amax2, Amin, Amin2;
int16 muestras[NMUESTRAS];          // letra j
int16 medias[NMEDIAS];              // letra m
int16 medias2[NMEDIAS];             // letra m2
int16 t1=0;
int16 t2=0;
int16 result=0;


#int_CCP1
CCP1_isr()
{
  t2=get_timer1();
  result=t2-t1;
  muestras[j]=result;
  j++;

  disable_interrupts(INT_CCP1);

  output_high(INH);                 // Deshab tras recibir la interrupcion
  output_low(CTRL_SW_TX0);          // Deshab los switches de una dir
  output_low(CTRL_SW_TX1);
  output_low(CTRL_SW_RX0);
  output_low(CTRL_SW_RX1);
  output_low(CTRL_SW_TX2);          // Deshab los switches de la otra dir
  output_low(CTRL_SW_TX3);
  output_low(CTRL_SW_RX2);
  output_low(CTRL_SW_RX3);
}
```

```
void main()
{

  setup_adc_ports(NO_ANALOGS);
  setup_adc(ADC_OFF);
  setup_psp(PSP_DISABLED);
  setup_spi(FALSE);
  setup_wdt(WDT_OFF);
  setup_timer_0(RTCC_INTERNAL);
  setup_timer_1(T1_INTERNAL|T1_DIV_BY_1);
  setup_timer_2(T2_DISABLED,0,1);
  setup_timer_3(T3_DISABLED|T3_DIV_BY_1);
  setup_ccp1(CCP_CAPTURE_RE);
  setup_comparator(NC_NC_NC_NC);
  setup_vref(FALSE);
  enable_interrupts(INT_CCP1);
  enable_interrupts(GLOBAL);
  setup_low_volt_detect(FALSE);
  setup_oscillator(False);



  // CONDICIONES INICIALES

  output_low(TX_PULSOS);
  output_low(TX_OSC);

  output_high(INH);              // Al principio deshabilitamos también el
                                 // multiplexor de recepción
  output_low(MUX_A);
  output_high(NMUX_A);
  a=0;                           // Se correspondera siempre con MUX_A

  output_low(CTRL_SW_TX0);     // Deshab los switches de la dir01
  output_low(CTRL_SW_TX1);
  output_low(CTRL_SW_RX0);
  output_low(CTRL_SW_RX1);

  output_low(MUX_B);
  b=0;

  output_low(CTRL_SW_TX2);      // Deshab los switches de la dir23
  output_low(CTRL_SW_TX3);
  output_low(CTRL_SW_RX2);
  output_low(CTRL_SW_RX3);
```

```
while(1)          // Bucle permanente
{
  m=0;
  m2=0;

  while(m<NMEDIAS && m2<NMEDIAS)     // se obtiene la media total
  {
    if(j==NMUESTRAS)                 // obtener media de las muestras
    {

      // la media se obtiene despreciando la primera de las muestras y
      // los valores mayores y menores del conjunto de muestras

      Amax=MIN_16BIT;                // constantes de max y min, para que
      Amax2=MIN_16BIT;               // cambien seguro
      Amin=MAX_16BIT;
      Amin2=MAX_16BIT;

                  // Bloque que toma NMUESTRAS muestras
      j=1;        // para que el while no tome en cuenta la muestra 0

      while(j<NMUESTRAS)
      {
        if(Amax<muestras[j])
        {
          Amax2=Amax;
          Amax=muestras[j];
        }
        else if(Amax2<muestras[j])
                  Amax2=muestras[j];

        if(Amin>muestras[j])
        {
          Amin2=Amin;
          Amin=muestras[j];
        }
        else if(Amin2>muestras[j])  Amin2=muestras[j];

        j++;
      }

      // se han tomado NMUESTRAS muestras ya


      // se va a sacar una media, segun sentido, no importa direccion,
      // solo se mira MUX_A
```

```
  if(a)    // obtener el valor m2 del vector medias2
  {
    medias2[m2]=0;
    for(j=1;j<NMUESTRAS;j++)
        medias2[m2]=medias2[m2]+muestras[j];
    j=0;
    medias2[m2]=medias2[m2]-Amin-Amax-Amin2-Amax2;
    medias2[m2]=medias2[m2]/(NMUESTRAS-5);
    m2++;
  }
  else     // obtener el valor m del vector medias
  {
    medias[m]=0;
    for(j=1;j<NMUESTRAS;j++)
        medias[m]=medias[m]+muestras[j];
    j=0;
    medias[m]=medias[m]-Amin-Amax-Amin2-Amax2;
    medias[m]=medias[m]/(NMUESTRAS-5);
    m++;
  }
}
i=1;

if(b)
{
  if(a) output_high(CTRL_SW_TX3);
  else  output_high(CTRL_SW_TX2);
}
else     // En este caso siempre
{
  if(a) output_high(CTRL_SW_TX1);   // Switch para pulsos
  else  output_high(CTRL_SW_TX0);
}

delay_us(DELAY_TX);     // Espera antes de dar los pulsos

t1=get_timer1();

while(i<=NPULSOS)
{
  output_high(TX_PULSOS);
  delay_cycles(NCICLOS1);
  delay_us(N_US);
  output_low(TX_PULSOS);
  delay_us(N_US-1);
  delay_cycles(NCICLOS2);
  i=i+1;
}
```

```
   delay_us(DELAY_RX1);          // Espera un poco antes de habilitar
                                 // la recepcion

   if(b)
   {
     if(a) output_high(CTRL_SW_RX2);
     else  output_high(CTRL_SW_RX3);
   }
   else    // En este caso siempre
   {
     if(a) output_high(CTRL_SW_RX0);   // Switch para pulsos
     else  output_high(CTRL_SW_RX1);
   }

   output_low(INH);              // Habilit el mux

   delay_us(DELAY_RX2);    // Se espera tambien antes de habilitar
                           // las interrupciones de nuevo
   clear_interrupt(INT_CCP1);
   enable_interrupts(INT_CCP1);

   delay_ms(N_MS);         // En este tiempo se espera recibir
}                          // la interrupción



// Despues de obtener las NMUESTRAS se procede a su análisis
// para obtener al final una media final

if(a)
{
   m2=0;
   Amax=medias2[0];
   Amin=medias2[0];
   while(m2<NMEDIAS)
   {
     if (Amax<medias2[m2])  Amax=medias2[m2];
     if (Amin>medias2[m2])  Amin=medias2[m2];
     m2++;
   }
   media2=0;

   for(m2=0;m2<NMEDIAS;m2++) media2=media2+medias2[m2];
   media2=media2-Amin-Amax;
   media2=media2/(NMEDIAS-2);
}
```

```
else
{
  m=0;
  Amax=medias[0];
  Amin=medias[0];
  while(m<NMEDIAS)
  {
    if (Amax<medias[m])  Amax=medias[m];
    if (Amin>medias[m])  Amin=medias[m];
    m++;
  }
  media=0;
  for(m=0;m<NMEDIAS;m++)
          media=media+medias[m];
  media=media-Amin-Amax;
  media=media/(NMEDIAS-2);
}



// Ahora se debe proceder a conmutar los multiplexores y obtener la
// media de una direccion si es necesario

if(a)    // conmutar a y b
{
  output_low(MUX_A);
  output_high(NMUX_A);
  a=0;

  if(b)    // segun b esta en la direccion 01 ó 23
  {
    dif23=(media-media2)/2;
    med23=(media+media2)/2;

    output_low(MUX_B);
    b=0;

    putc(dif01);
    putc(make8(med01,1));
    putc(make8(med01,0));

    putc(dif23);
    putc(make8(med23,1));
    putc(make8(med23,0));
  }
```

```
      else
      {
        dif01=(media-media2)/2;
        med01=(media+media2)/2;

        output_high(MUX_B);
        b=1;
      }

    }
    else    // conmutar solo a
    {
      output_high(MUX_A);
      output_low(NMUX_A);
      a=1;
    }
  }
}
```

```
*********************************************************************************************
*        Archivo Final.h                                                                    *
*********************************************************************************************

#include <18F458.h>
#device ICD=TRUE
#device adc=8

#FUSES NOWDT                    //No Watch Dog Timer
#FUSES WDT128                   //Watch Dog Timer uses 1:128 Postscale
#FUSES EC_IO                    //External clock
#FUSES NOPROTECT                //Code not protected from reading
#FUSES NOOSCSEN                 //Oscillator switching is disabled, main
                                //oscillator is source
#FUSES BROWNOUT                 //Reset when brownout detected
#FUSES BORV20                   //Brownout reset at 2.0V
#FUSES NOPUT                    //No Power Up Timer
#FUSES NOCPD                    //No EE protection
#FUSES STVREN                   //Stack full/underflow will cause reset
#FUSES DEBUG                    //Debug mode for use with ICD
#FUSES LVP                      //Low Voltage Programming on B3(PIC16)
                                //or B5(PIC18)
#FUSES NOWRT                    //Program memory not write protected
#FUSES NOWRTD                   //Data EEPROM not write protected
#FUSES NOWRTB                   //Boot block not write protected
#FUSES NOCPB                    //No Boot Block code protection
#FUSES NOWRTC                   //configuration not registers write
                                //protected
#FUSES NOEBTR                   //Memory not protected from table reads
#FUSES NOEBTRB                  //Boot block not protected from table reads

#use delay(clock=40000000)
#define MUX_A   PIN_B4
#define NMUX_A  PIN_A0          // MUX_A negado, para el MUX RX
#define MUX_B   PIN_A1
#define CTRL_SW_TX1   PIN_A2
#define CTRL_SW_TX3   PIN_A3
#define TX_PULSOS   PIN_D7
#define TX_OSC      PIN_D2
#define CTRL_SW_TX2   PIN_A5
#define CTRL_SW_RX2   PIN_D5
#define CTRL_SW_RX3   PIN_C0
#define CTRL_SW_RX1   PIN_C1
#define CTRL_SW_TX0   PIN_E0
#define INH   PIN_E1
#define CTRL_SW_RX0   PIN_E2
#use rs232(baud=9600,parity=N,xmit=PIN_C6,rcv=PIN_C7,bits=8)
```

```
*****************************************************************************************
*        Archivo Constantes.h                                                           *
*****************************************************************************************

// En este archivo se muestran los distintos valores constantes
// que se tienen en el codigo
// Para una modificación, es más fácil realizarla cambiando solamente
// el valor de la constante sin que haga falta conocer los puntos donde se
usa


#define NMUESTRAS     13      // Muestras directas. Minimo 5
#define NMEDIAS        8      // Muestras directas. Minimo 3
#define NVALORES       8      // De ellos se hará la media

#define NPULSOS        2
#define N_MS           3      // Tiempo entre pulsos (en milisegundos)

#define DELAY_TX      31      // Numero de us de espera para la
#define DELAY_RX1     127     // transmisión y para la recepción
#define DELAY_RX2     155


#define MAX_16BIT   65535
#define MIN_16BIT       0


// definen espacio entre pulsos
// valores para frecuencia de 40 KHz

#define N_US           12
#define NCICLOS1        2
#define NCICLOS2        6

#define UMBRAL         10
```

# Anexo 2.- Placa pcb

A continuación se muestran las huellas que se quieren dejar sobre la placa para realizar todas las conexiones. Al ser necesarias un gran número de conexiones se ha rutado por las superficies de arriba (TOP) y por debajo (BOTTOM) de la placa.

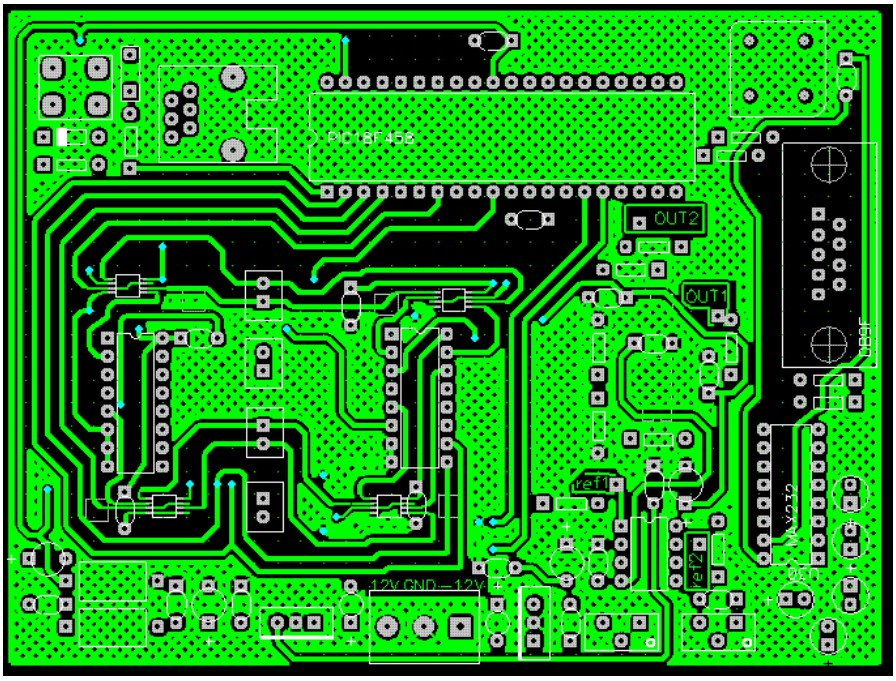


**Ilustración 88: TOP de la placa pcb**

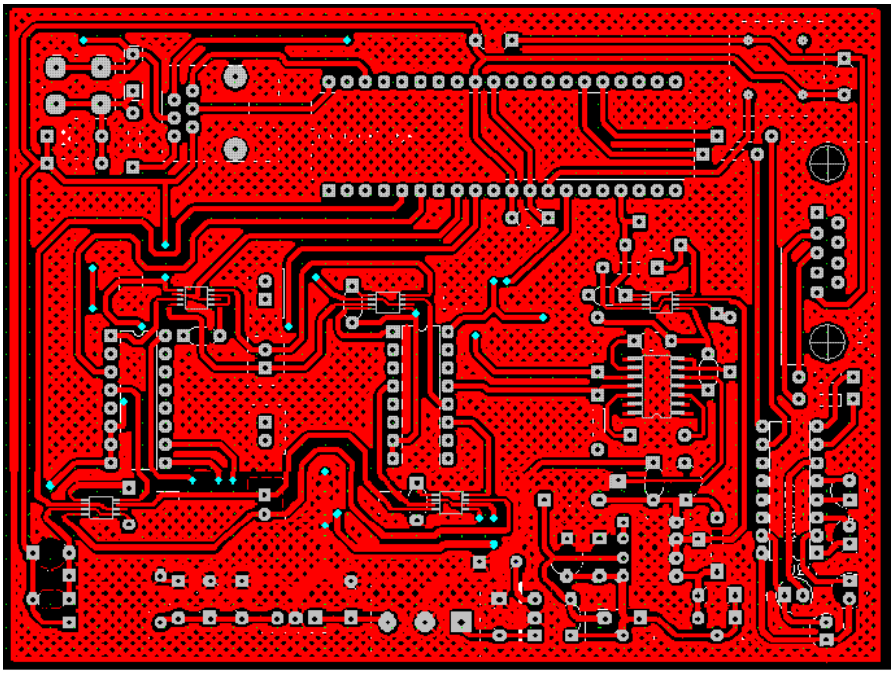

**Ilustración 89: BOTTOM de la placa pcb**