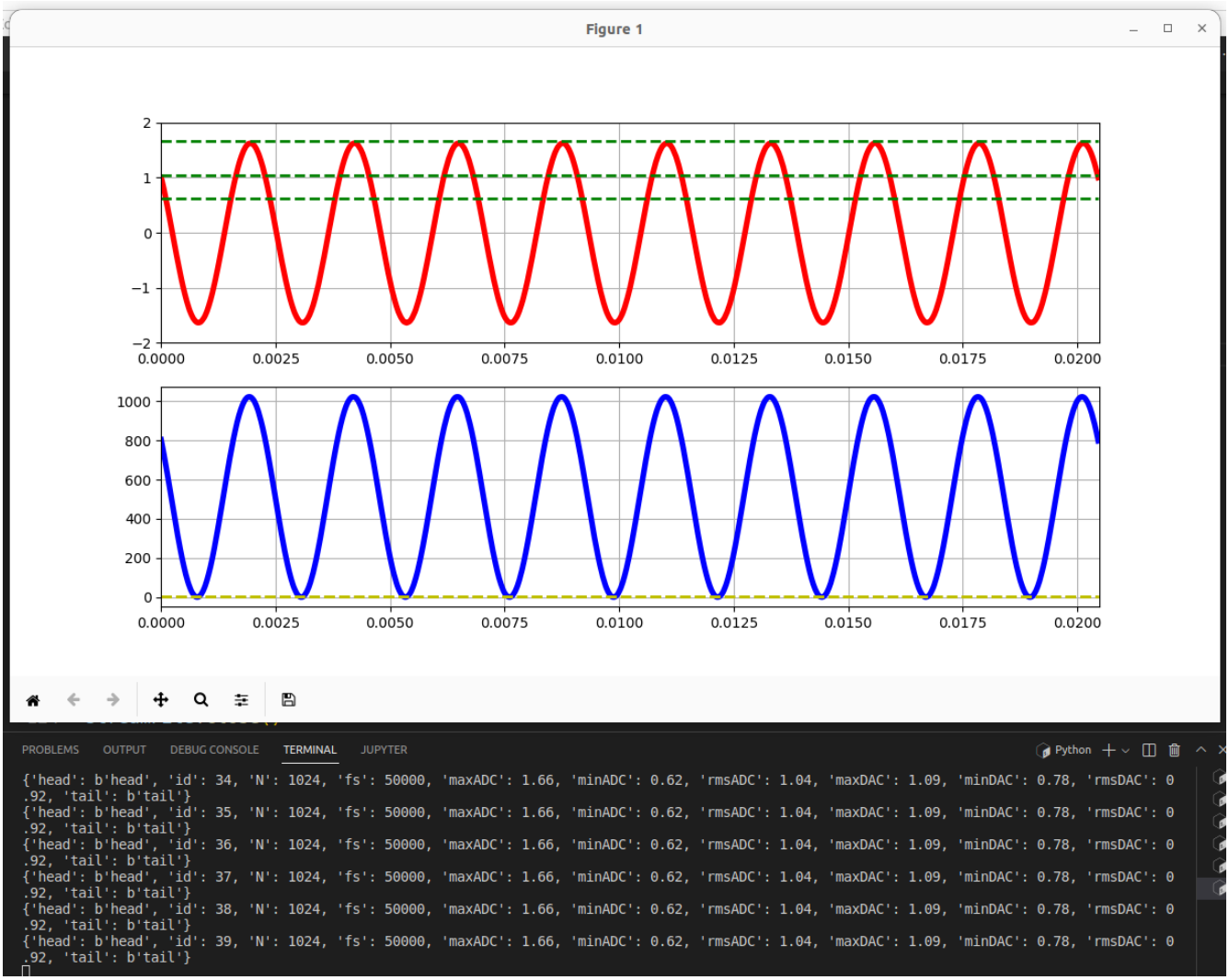
1) Genere con un tono de LA-440. Digitice con 10 y luego con 4 bits, envíe los datos a la PC y grafique:

1) Señal original con su máximo, mínimo y RMS

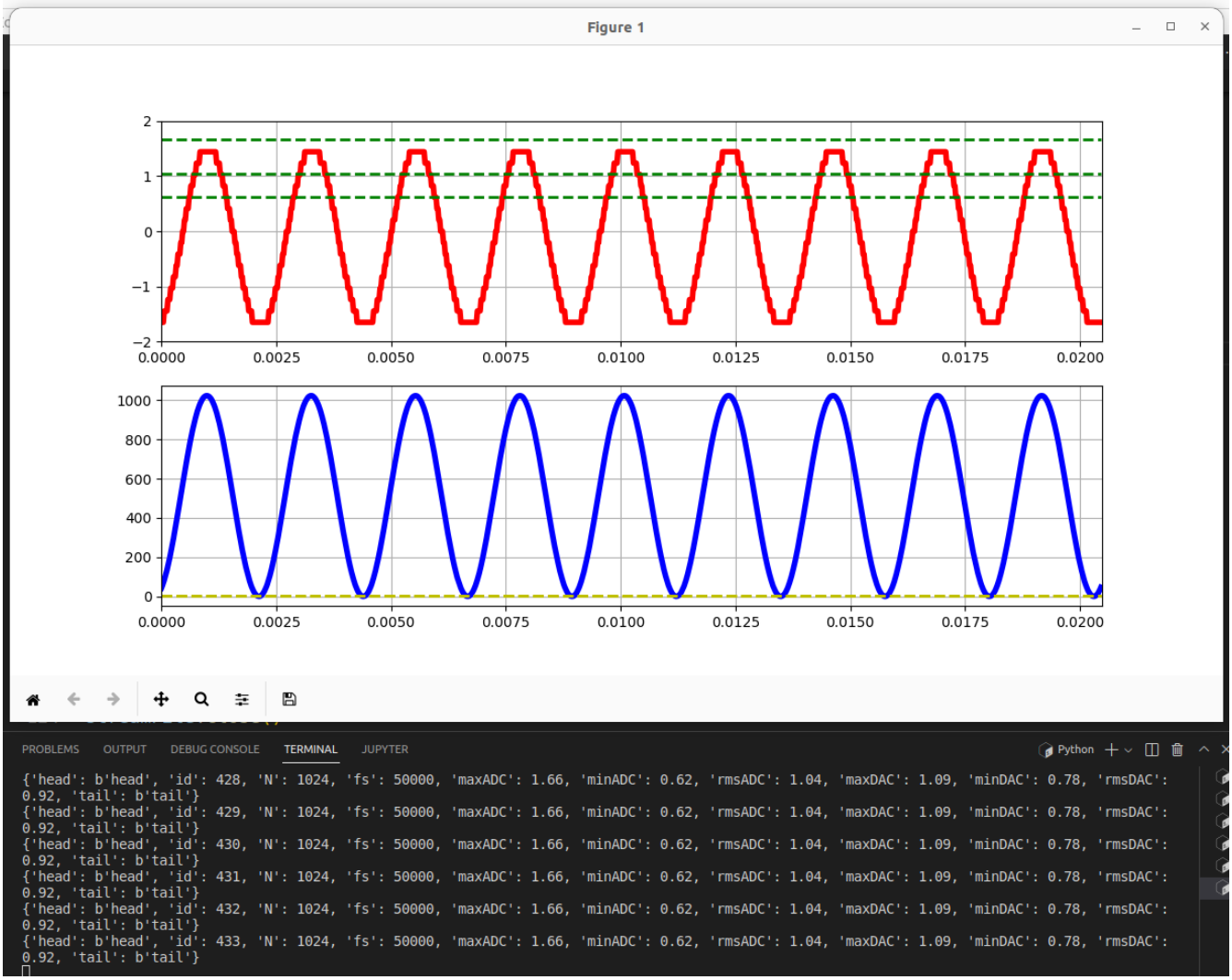
2) Señal adquirida con su máximo, mínimo y RMS.

Hay diferencias? a que se deben?. Se presentan diferencias debido a que a menor numero de bits disminuye la resolución, por lo que se pierde simetría en la señal.

Señal original y señal adquirida a 10 bits



Señal original y señal adquirida a 4 bits



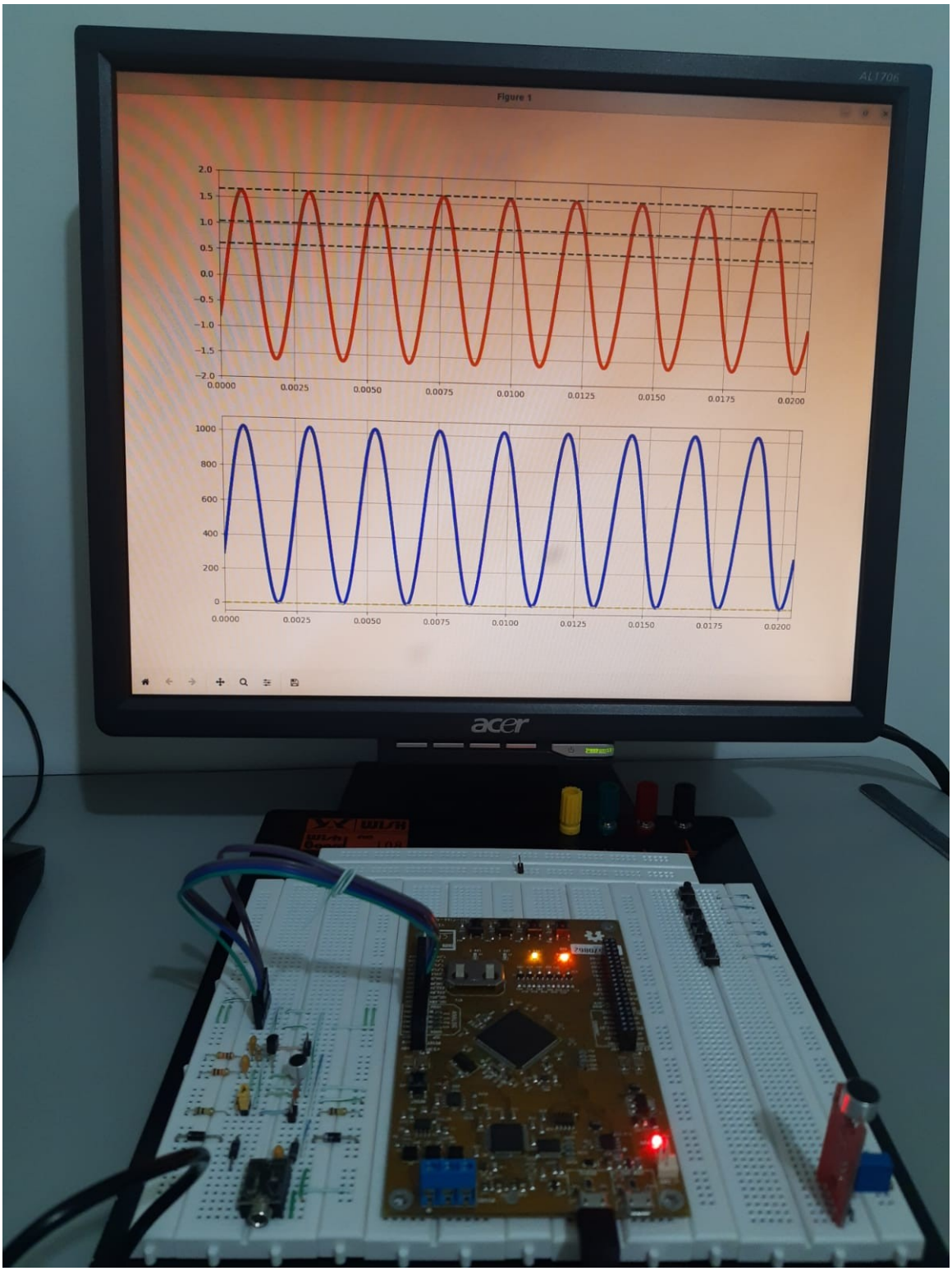
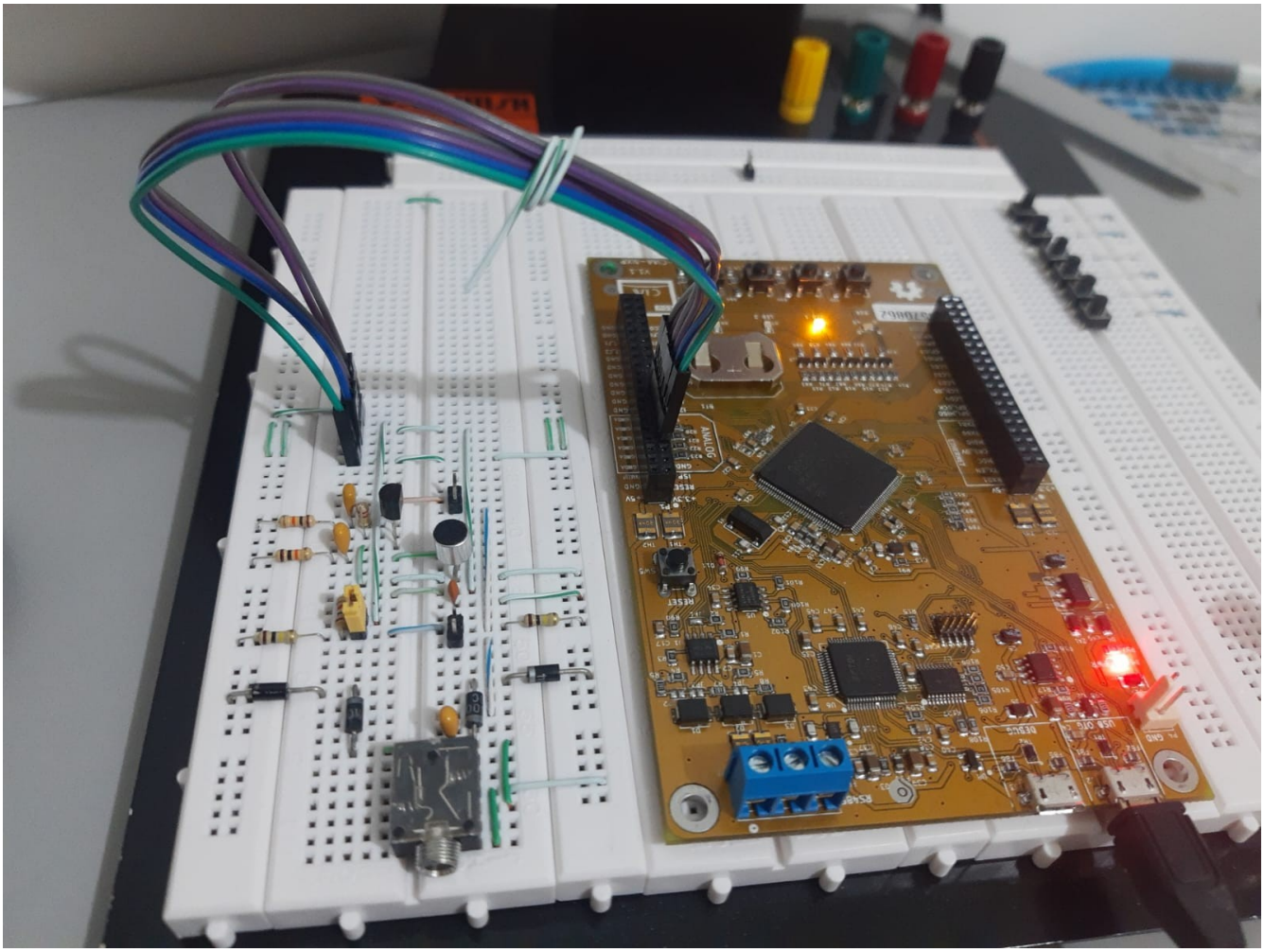


Figure 1
2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
0.0000
0.0025
0.0050
0.0075
0.0100
0.0125
0.0150
0.0175
0.0200
1000
800
600
400
200
0
acer



Los códigos se pueden verificar en siguiente enlace

https://github.com/jdalvaradocol/PSF_MSE.git

El archivo es *TP1_B.c* y el archivo en python es *visualize.py*

Código en C

```
/*==================================================================
 * Author: Jose David <alvaradomoreno.jd@gmail.com>
 * Date: 2022/07/25
 *==================================================================*/

/*=====[Inclusions of function dependencies]=========================*/

#include "TP1_B.h"
#include "sapi.h"
#include "arm_math.h"
#include "arm_const_structs.h"

#define BITS 10                                 // cantidad de bits usado para cuantizar
```

```
uint32_t tick   = 0   ;
uint16_t tone   = 440 ;


struct header_struct {
   char          pre[4];
   uint32_t id;
   uint16_t N;
   uint16_t fs ;
   q15_t         maxADC;
   q15_t         minADC;
   q15_t         rmsADC;
   q15_t         maxDAC;
   q15_t         minDAC;
   q15_t         rmsDAC;
   char          pos[4];
} __attribute__ ((packed));

struct header_struct header={"head",0,1024,50000,0,0,0,0,0,0,"tail"};

void trigger(int16_t threshold)
{
   while((adcRead(CH1)-512)>threshold)
            ;
   while((adcRead(CH1)-512)<threshold)
            ;
   return;
}

int main ( void )

{

         uint16_t sample = 0;
         uint32_t Index = 0;
         int16_t adc [ header.N          ];
         int16_t dac [ header.N          ];

         boardConfig          (                                          );
         uartConfig           ( UART_USB ,460800                         );
         adcConfig            ( ADC_ENABLE                               );
         dacConfig            ( DAC_ENABLE                               );
         cyclesCounterInit ( EDU_CIAA_NXP_CLOCK_SPEED );

         while(1)
         {

                  cyclesCounterReset();
```

```
                uartWriteByteArray ( UART_USB ,(uint8_t* )&dac[sample] ,sizeof(dac[0]) );      //
envia el sample ANTERIOR
                uartWriteByteArray ( UART_USB ,(uint8_t* )&adc[sample] ,sizeof(adc[0]) );      //
envia el sample ANTERIOR

                adc[sample]    = (((int16_t )adcRead(CH1)-512)>>(10-BITS))<<(6+10-BITS);
          // PISA el sample que se acaba de mandar con una nueva muestra

                float t = (tick)/((float)header.fs);
                tick++;

                int16_t dac_signal = 512 * arm_sin_f32 (t*tone*2*PI)+512;

                dac[sample] = dac_signal;

                dacWrite( DAC, dac_signal);             // tono

                if ( ++sample==header.N )
                {
                        gpioToggle ( LEDR );                                                    // este led
blinkea a fs/N

                        sample = 0;

                        arm_max_q15 ( (q15_t)(adc) ,header.N/2+1 ,&header.maxADC , &Index );
                        arm_min_q15 ( (q15_t)(adc) ,header.N/2+1 ,&header.minADC , &Index );
                        arm_rms_q15 ( (q15_t)(adc) ,header.N/2+1 ,&header.rmsADC );

                        arm_max_q15 ( (q15_t)(dac) ,header.N/2+1 ,&header.maxDAC ,&Index );
                        arm_min_q15 ( (q15_t)(dac) ,header.N/2+1 ,&header.minDAC ,&Index );
                        arm_rms_q15 ( (q15_t)(dac) ,header.N/2+1 ,&header.rmsDAC );

                        // trigger(2);

                        header.id++;
                        uartWriteByteArray ( UART_USB ,(uint8_t*)&header ,sizeof(struct
header_struct ));

                                adcRead(CH1); //why?? hay algun efecto minimo en el 1er sample..
puede ser por el blinkeo de los leds o algo que me corre 10 puntos el primer sample. Con esto se
resuelve.. habria que investigar el problema en detalle
                 }
                 gpioToggle ( LED1 );
                // este led blinkea a fs/2
                 while(cyclesCounterRead()< EDU_CIAA_NXP_CLOCK_SPEED/header.fs) // el clk
de la CIAA es 204000000
                        ;
  }
}
```

Codigo en Python

```
#!python3
import numpy as np
import matplotlib.pyplot as plt
from   matplotlib.animation import FuncAnimation
import os
import io
import serial
from fxpmath import Fxp

STREAM_FILE=("/dev/ttyUSB1","serial")
#STREAM_FILE=("log.bin","file")

header = {  "head": b"head", "id": 0, "N": 1024, "fs": 50000,
          "maxADC":0, "minADC":0, "rmsADC":0,
          "maxDAC":0, "minDAC":0, "rmsDAC":0,
          "tail":b"tail" }
fig    = plt.figure ( 1 )

adcAxe      = fig.add_subplot ( 2,1,1                              )
adcLn,      = plt.plot        ( [],[], 'r-', linewidth=4           )
maxadcLn, = plt.plot          ( [],[], 'g--', linewidth = 2, alpha = 1  )
minadcLn, = plt.plot          ( [],[], 'g--', linewidth = 2, alpha = 1  )
rmsadcLn, = plt.plot          ( [],[], 'g--', linewidth = 2, alpha = 1   )
adcAxe.grid                 ( True                                 )
adcAxe.set_ylim               ( -2 , 2                               )

dacAxe      = fig.add_subplot ( 2,1,2                              )
dacLn,      = plt.plot        ( [],[], 'b-', linewidth=4           )
maxdacLn, = plt.plot          ( [],[], 'y--', linewidth = 2, alpha = 1  )
mindacLn, = plt.plot          ( [],[], 'y--', linewidth = 2, alpha = 1  )
rmsdacLn, = plt.plot          ( [],[], 'y--', linewidth = 2, alpha = 1   )
dacAxe.grid                 ( True                                 )
dacAxe.set_ylim               ( -50 ,2**10 + 50                      )

def findHeader(f,h):
   find=False
   while(not find):
      data=bytearray(len(h["head"]))
      while data!=h["head"]:
         data+=f.read(1)
         data[:]=data[-4:]

      h["id"]       = readInt4File(f,4)
      h["N" ]       = readInt4File(f)
      h["fs"]       = readInt4File(f)
      h["maxADC"] = (readInt4File(f,sign = True)/2**14) # Se normaliza el valor del adc/2¹⁴
```

```
        h["minADC"] = ((readInt4File(f,sign = True))*1.65)/(2**6*512)
        h["rmsADC"] = ((readInt4File(f,sign = True))*1.65)/(2**6*512)
        h["maxDAC"] = ((readInt4File(f,sign = True))*1.65)/(2**6*512)
        h["minDAC"] = ((readInt4File(f,sign = True))*1.65)/(2**6*512)
        h["rmsDAC"] = ((readInt4File(f,sign = True))*1.65)/(2**6*512)

        data=bytearray(b'1234')
        for i in range(4):
            data+=f.read(1)
            data[:]=data[-4:]
        find = data==h["tail"]

    print({k:round(v,2) if isinstance(v,float) else v for k,v in h.items()})
    return
h["id"],h["N"],h["fs"],h["maxADC"],h["minADC"],h["rmsADC"],h["maxDAC"],h["minDAC"],h["rms
DAC"]

def readInt4File(f,size=2,sign=False):
    raw=f.read(1)
    while( len(raw) < size):
        raw+=f.read(1)
    return (int.from_bytes(raw,"little",signed=sign))

def flushStream(f,h):
    if(STREAM_FILE[1]=="serial"): #pregunto si estoy usando la bibioteca pyserial o un file
        f.flushInput()
    else:
        f.seek ( 2*h["N"],io.SEEK_END)

def readSamples(adc,dac,N,trigger=False,th=0):
    state="waitLow" if trigger else "sampling"
    i=0
    for t in range(N):
        sample_dac = (readInt4File(streamFile,sign = True)*1)/(1)
        sample     = (readInt4File(streamFile,sign = True)*1.65)/(2**6*512)
        state,nextI= {
                "waitLow" : lambda sample,i: ("waitHigh",0) if sample<th else ("waitLow" ,0),
                "waitHigh": lambda sample,i: ("sampling",0) if sample>th else ("waitHigh",0),
                "sampling": lambda sample,i: ("sampling",i+1)
                }[state](sample,i)
        adc[i] = sample
        dac[i] = sample_dac
        i=nextI

def update(t):
    global header
#    flushStream ( streamFile,header )
    id,N,fs,maxADC,minADC,rmsADC,maxDAC,minDAC,rmsDAC = findHeader
( streamFile,header )
```

```
    adc     = np.zeros(N)
    dac     = np.zeros(N).astype(complex)
    time    = np.arange(0,N/fs,1/fs)

    readSamples(adc,dac,N,False,0)

    adcAxe.set_xlim   ( 0    , N/fs   )
    adcLn.set_data    ( time , adc    )
    maxadcLn.set_data ( time , maxADC )
    minadcLn.set_data ( time , minADC )
    rmsadcLn.set_data ( time , rmsADC )

    dacAxe.set_xlim ( 0    ,N/fs )
    dacLn.set_data  ( time ,dac  )
    maxdacLn.set_data ( time , maxDAC )
    mindacLn.set_data ( time , minDAC )
    rmsdacLn.set_data ( time , rmsDAC )


    return adcLn, maxadcLn, minadcLn, rmsadcLn, dacLn, maxdacLn, mindacLn, rmsdacLn,

#seleccionar si usar la biblioteca pyserial o leer desde un archivo log.bin
if(STREAM_FILE[1]=="serial"):
    streamFile = serial.Serial(port=STREAM_FILE[0],baudrate=460800,timeout=None)
else:
    streamFile=open(STREAM_FILE[0],"rb",0)

ani=FuncAnimation(fig,update,10000,init_func=None,blit=True,interval=1,repeat=True)
plt.draw()
#plt.get_current_fig_manager().window.showMaximized() #para QT5
plt.show()
streamFile.close()
```