# MATERIA: INSTRUMENTACIÓN Y CONTROL

# PREINFORME

#### Integrantes Grupo:

Juan Doppler

Lucas Pavlov

Adrián Tichno

Fecha: 19 de septiembre de 2018

# TOMA DE DATOS DE COMPONENTES ELECTRÓNICOS CON PLACA DE AUDIO DE PC

## OBJETIVOS

El objetivo de este trabajo es realizar un programa de medición de datos en lenguaje PYTHON a partir de la placa de audio de una PC de escritorio con sistema operativo Windows y posteriormente medir con la misma algunas características de componentes electrónicos, en este caso la curva I-V de un diodo y la respuesta de un amplificador operacional a lazo abierto.

## COMPONENTES

En particular en este trabajo utilizamos una PC de escritorio con sistema operativo Windows XP y una placa de audio con salida de señal estéreo y entradas de micrófono y line-in. Se utilizaron conectores de audio estéreo preparados para conectar la placa de audio con un protoboard.

La salida de señal estéreo se utilizó para inyectar señal a los distintos circuitos a evaluar, y la entrada de line in se utilizó para tomar datos.

La programación se realizó en PYTHON. Para el manejo de audio se trabajó con Pyaudio. Para compartir el trabajo en el laboratorio se utilizaron cuentas de GITHUB, y se fueron realizando commits, primero a la cuenta de uno de los integrantes del grupo, y posteriormente a la grupal, para poder realizar cambios durante la semana.

Se comprobó el correcto funcionamiento del programa a través de mediciones de la salida y entrada de la placa de audio con un osciloscopio Tektronix TDS 10028 y un generador de funciones Tektronix AFG3021B.

### PLACA DE AUDIO

La placa de audio utilizada fue la de una PC de escritorio con sistema operativo Windows XP. En general este tipo de placas inyectan una señal digital a un conversor digital - analógico que convierte la señal digital en una analógica escalonada, la cual es filtrada con un pasabajos, eliminando en lo posible esos escalones y produciendo la salida analógica de audio de unos pocos volts. La frecuencia de salida está optimizada para el oído humano, es decir para reproducir fielmente frecuencias entre 20 a 20.000 Hz. Una buena placa de audio produce señales con menos de 0,3 dB de variación entre estas frecuencias. Los dispositivos tipo micrófonos o speakers tienen mayores variaciones que los estrictamente electrónicos (se consideran aceptables valores de 2 dB de variación en la respuesta de frecuencias).

Las placas en PC de escritorio poseen una entrada de micrófono y una de LINE-IN. Las entradas de micrófono suelen admitir muy poca señal, tienen un offset de continua destinada a alimentar el micrófono y una impedancia ajustada para ese equipamiento. Las de LINE –IN en cambio permiten la toma de datos en estéreo y están preparadas para admitir señales de audio de equipos estándar. Las mismas cuentan con un conversor analógico -digital que convierte la señal de entrada a una señal digital.

La placa de audio tiene una API propia que vincula el hardware con Windows.

Durante las mediciones trabajamos con sample rates de 44100, 96000 y 192000 Hz y 24 y 32 bits de profundidad. ((chequear- En el soft está así de acuerdo a lo que usamos, pero no anote como estaba en la API original e ignoro si en este caso la API manda o si el pyaudio puede cambiarlo.))

### PYAUDIO

Dado que la placa de audio trabaja con una API propia que la relaciona con Windows, y esta no es multiplataforma, buscamos una API hecha en Python que nos permitiera manejarla y a la vez permitir que el código utilizado sea multiplataforma. Luego de una breve búsqueda en internet nos inclinamos por PYAUDIO.

PyAudio es un desarrollo en Python que replica a PortAudio escrito en C. PortAudio fue pensada para simplificar el manejo multiplataforma. En la figura 1 mostramos un esquema de capas donde se observan los distintos niveles de manejo de audio.

PortAudio

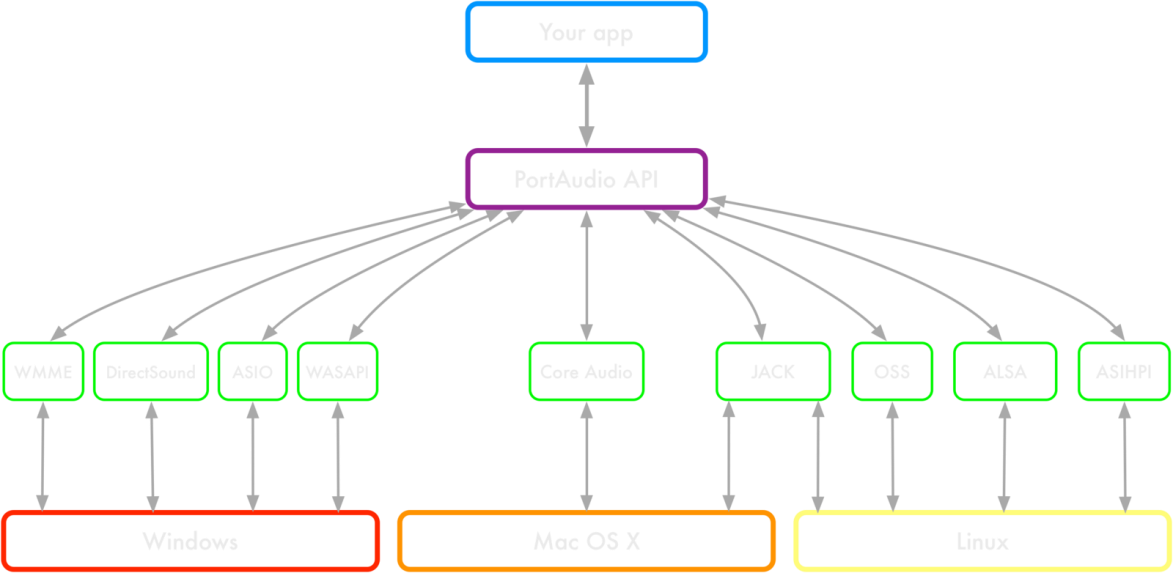


Figura 1: Esquema de capas de PortAudio. PyAudio es la implementación del mismo en Python.

Este esquema de capas, nos permitió trabajar de modo de poder probar el software en una Mac y en varios sistemas Windows y Linux, y a la vez trabajar en la PC de escritorio del laboratorio.

Posteriormente mostró tener un problema que encontramos durante las mediciones. Esencialmente la API se monta sobre la propia de la PC, por lo cual no se desactivan los amplificadores de la placa sino que estos están manejados por la API de Windows. En otras palabras, la intensidad de señal que se envíe por software por medio de PyAudio se montará por sobre los volúmenes de salida y grabación que estén fijados en Windows.

### INYECCIÓN DE SEÑAL Y TOMA DE DATOS

Para la salida de señal y la toma de señal se usaron conectores de audio minijack de 3.5 mm tipo TRS (Tip-ring-sleeve) de 3 sectores, similar al que se muestra en la Fig. 2. La punta y el anillo se usan para señal estéreo, y la funda para la tierra. En nuestro caso, para la PC utilizamos dos de estos conectores, uno de ellos para la salida de audio (salida de señal) y otro en el line-in, lo que permite tener salida en dos canales y entrada en dos canales. En el caso de las notebooks modernas, se utiliza el conector de 4 sectores, TRRS. Este conector tiene una salida estéreo y una entrada de micrófono. Si bien es posible utilizarlo, presenta desventajas en cuanto a que la señal de entrada no es estéreo (solo se puede medir una cantidad a la vez) y las impedancias de ese canal están preparadas para la toma de señal típica de un micrófono y no de una medición. El otro extremo del conector, eran simplemente cables que se conectaron a una protoboard para conformar los distintos circuitos.

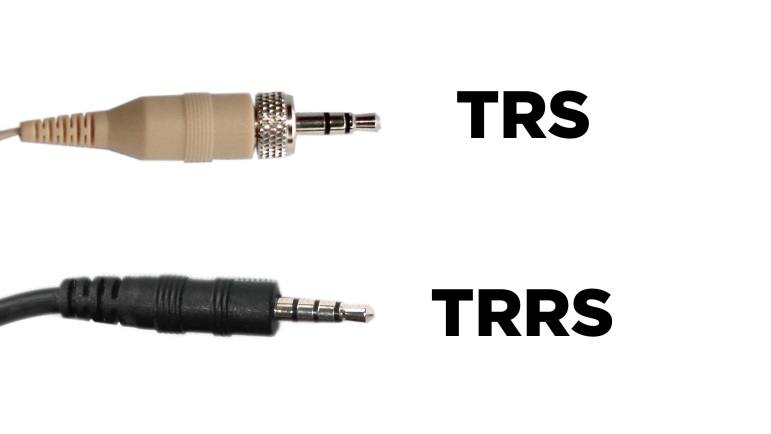


Fig 2: Conectores minijack 3.5 mm tipo TRS y TRRS

Los componentes de los circuitos electrónicos y conexiones empleados en cada etapa de las mediciones fueron montados en una protoboard de pruebas.

## DESARROLLO DEL TRABAJO

Los pasos del presente trabajo fueron los siguientes:

* Búsqueda de una API adecuada para mediar entre la placa de audio y PYTHON.
* Programación y primeras pruebas para generación de pulsos a través de la salida de la placa de audio.
* Programación y primeras pruebas para medición de pulsos a través de la entrada de la placa de audio.
* Programación con Callback para realizar una salida de señal y una medición de señal simultáneamente a través de la placa de audio.
* Preparación de distintas funciones para emplear durante las mediciones.
* Programación de la entrada en estéreo para tomar dos señales simultaneas.
* Medición de la respuesta en frecuencia de la placa de audio.
* Medición de la curva I-V de un diodo.
* Medición de la respuesta en frecuencia a lazo abierto de un amplificador operacional (en curso).

(Notas a agregar)

### BUSQUEDA DE UNA API ADECUADA PARA PROGRAMAR EN PYTHON

Realizamos una búsqueda en internet durante la clase de laboratorio de una API que nos permitiera enviar señal de audio y leer señal de una entrada. A partir de diversos comentarios que leímos y dado que encontramos software que nos permitía comenzar a medir rápidamente, nos decidimos por PyAudio. En un trabajo con mayor tiempo, sería recomendable analizar las ventajas técnicas de este contra otras posibles API, pero en lo que sigue utilizamos esta API. Las principales ventajas de PyAudio se comentaron en la introducción.

### PROGRAMACIÓN Y GENERACIÓN DE PULSOS

Durante las primeras pruebas se realizó un programa para enviar los datos a una salida, que fueron medidos conectando directamente la salida a un osciloscopio y luego a un par de parlantes externos.

Pudimos generar correctamente una señal. Observamos que no se podía generar una rampa, y que en la generación de un pulso cuadrado los bordes presentaban mucho ruido. Esto se debe a que la placa de audio cuenta con un filtro pasabajos regulado para entregar linealmente la intensidad dentro del rango audible de entre 20 Hz-20 kHz.

Una de las primeras cosas que observamos fue que la terminal de Python debía ser iniciada posteriormente a las conexiones y configuraciones que se hiciesen directamente en la placa de audio vía la API de Windows.

### PROGRAMACIÓN Y TOMA DE DATOS

Luego de obtener una primera medición de salida, programamos la toma de datos e inyectamos una señal sinusoidal de 300 mVpp a distintas frecuencias con la intención de ver la respuesta en frecuencias de la placa de audio.

En esta etapa de prueba pudimos ver que la señal caía a partir de los 18 KHz, lo cual indicaba que la señal ingresada tenía una etapa de filtrado como también la tenía la señal de salida. Entonces evaluamos que para realizar la determinación de la respuesta en frecuencias del conjunto completo, salida-entrada, debíamos realizar una medición simultánea entre ambas terminales.

Al comienzo de esta toma de datos, observamos que la señal de entrada del micrófono tenía un offset de 1.5V, lo que nos llevó a cambiar la configuración de la señal por una de LINE-IN como toma de datos.

### PROGRAMACIÓN CON CALLBACK

((REVISAR TODO ESTO CON CUIDADO POR FAVOR, que lo escribí un poco confuso))

El siguiente paso en el trabajo fue implementar una función de CALLBACK en el software de inyección de señal y toma de datos. Esta función es necesaria para que se pueda realizar la salida de señal desde la placa y medir asincrónicamente la señal de entrada. Inicialmente la implementamos sobre la entrada, para que avise cuando el buffer de la placa se llenaba y haga un append en la tira completa de datos. Posteriormente vimos que el método más adecuado era implementarla sobre la salida de señal, teniendo en cuenta que la frecuencia debía ser tal que repitiese la señal manteniendo continuidad en el buffer. Es decir que la primera parte de la señal del buffer despues de la llamada de callback sea continuidad de la última parte de la señal que tenía la última parte del buffer anterior, de modo tal que la señal de salida no presente saltos bruscos.

El tamaño del archivo de la entrada de señal se fija de antemano. Esto es un poco más eficiente que el modo contrario donde hay que agrandar el tamaño del vector de memoria en la entrada de señal cada vez que se agota el buffer de salida.

((No termino de entender del todo la función del cursor que escribió Hernan, me parece que es la que permite trabajar del modo que escribí más arriba, al entregar un cursor al callback que varía con las llenadas del buffer, pero si quieren explciarla mejor))

### FUNCIONES DE INYECCIÓN

(completar un poco más esto)

Se programaron distintas funciones para realizar las mediciones que necesitaríamos más adelante. Entre estas una función sinusoidal (que devuelve la frecuencia pedida teniendo en cuenta la frecuencia de sampleo de la placa de audio), una onda cuadrada y un barrido en frecuencias con variación de la intensidad de salida para realizar un barrido en frecuencias de la señal de la placa de audio.

Para graficar se utilizó una xx

### TOMA DE DATOS ESTEREO

Se observó que para la medición de la curva de I-V de un diodo, y para la medición de la curva a lazo abierto del op-amp, es ideal trabajar midiendo corriente y voltaje en diferentes puntos del circuito. Para eso se necesita tomar ambas mediciones de señal por un sendos canales de entrada. Se implementó la medición en estéreo del LINE-IN en el software de toma de datos. Verificamos que ambas señales estaban amplificadas y filtradas en correspondencia una de otra. Eso se debe a que la placa de audio toma el valor de la medición de un canal luego del siguiente ((Verificar esto¡¡, si es así deberíamos tener cuidado con la frecuencia de sampleo porque la debe tomar a la mitad para cada canal, es decir si usas 44100 estas reduciendo la calidad de la señal a 22000 para c/canal)) para el filtrado y la conversión de señal analógica a digital.

### MEDICIÓN DE LA RESPUESTA EN FRECUENCIAS DE LA PLACA DE AUDIO

((Aca se pueden agregar algunos de los graficos de la primera tanda de mediciones, onda el barrido6.dat +barrido 7.dat // Despues se agrega la curva final que hicimos con barridocondiciones diodo.dat que debería estar calibrada))

Para esta medición conectamos la salida de la placa de audio directamente con la entrada de la misma. La implementación de software fue tal que se enviaba la señal de inyección a diversas frecuencias espaciadas linealmente entre 1 y 50 Hz, también linealmente entre 15 kHz y 25 kHZ y logarítmicamente entre 50 Hz y 44 kHz variando la intensidad de señal de las unidades de software (amplitud pico a pico de la senoidal en Python) a los siguientes valores 0.1, 0.5, 1.0, 1.5, 2.0, 2.5 y 3.0 (llamaremos a estos valores en unidades arbitrarias Vsoft, expresados en valores pico a pico).

En una primer serie de barridos observamos que si la unidad relativa de volumen subía por sobre 2.0, además de la disminución de intensidad esperable con el incremento de frecuencias más allá de los 18 kHz, había una pérdida de ganancia adicional con respecto a los volúmenes menores. Esto indicaba que existía algún problema con la saturación de la señal de entrada.

Posteriormente cuando tratábamos de implementar la medición de la curva I-V del diodo nos dimos cuenta que dado que el PyAudio trabaja por sobre la API de Windows, esta última tiene el comando sobre la intensidad de salida y la amplificación de entrada, por lo que se deben conocer los valores en que esta está seteada para poder realizar una calibración correcta de la salida de señal y de la entrada de la respuesta de los circuitos que quisiéramos medir.

Después de varias pruebas decidimos que el método más conveniente para medir era amplificar al máximo el volumen de la placa de salida y poca amplificación a la entrada (speaker volume=100, record volume=7, playback volume=0). De esta manera, para valores de volumen de software de Vsoft=2 el sistema no distorsionaba y en su zona central de frecuencias tenía un valor medido de 2.6 Vpp en el osciloscopio, lo que permitió medir el rango completo de la curva I-V del diodo en directa.

Finalmente se tomó una curva completa de la respuesta en frecuencia del sistema para esos valores, que se utiizarían posteriormente en la curva I-V del diodo.

((Poner acá tabla con valores de calibración, gráfico medido y cuentas de calibración para poder luego utilizar en la curva IV del diodo))

### MEDICION DE LA CURVA I-V DE UN DIODO

El circuito esquemático utilizado para esta medición fue el siguiente

((diodo en directa, resistencia en serie de 68,6 KOhm, medido entre inyección de diodo y sobre resistencia a la vez))

Se implementó en software la toma de datos de la curva I-V para una frecuencia de señal sinusoidal de 1 kHz.

((no se si vale la pena poner las curvas con la intensidad de ganancia de grabación en 30 y 100, donde la señal satura)) -> quizás si valga la pena, para ilustrar que no pudimos medir en esas condiciones y comentar un poco las complicaciones que tuvimos

### MEDICIÓN DE LA CURVA DE RESPUESTA A LAZO ABIERTO DE UN OPAMP

Aún no realizamos esta medición. El software está preparado para realizarla luego de montar el circuito en pocos minutos.

## CONCLUSIONES

La placa de audio presentó una respuesta en frecuencias como esperábamos, con una zona de baja frecuencias filtrada, una meseta a partir de 10 Hz hasta prácticamente 17 kHz donde comienza a disminuir para llegar a 25 kHz con una caída de -23 dB. La variación entre puntas de la meseta no superó los xx dB.

El filtro de salida de la señal impide trabajar con señales sofisticadas que tengan altas frecuencias, como es el caso de una rampa. Esta es una limitante a tener en cuenta en caso que se utilice este tipo de equipos para medición.

Observamos que la terminal de Python debía ser iniciada posteriormente a las conexiones y configuraciones que se hiciesen directamente en la placa de audio via la API de Windows para que esos cambios tuviesen efecto en los programas ejecutados a través de la misma.

Durante las mediciones iniciales, observamos que la entrada de micrófono tiene un voltaje de continua de aproximadamente 1.5 V con respecto a tierra. Documentándonos encontramos que este voltaje se utiliza para la alimentación del micrófono.

En una primera medición de la entrada de señal a partir del LINE–IN, donde inyectamos la señal a través de un generador de funciones, observamos que la señal caía fuertemente a partir de frecuencias mayores a 18 KHz, con lo cual se concluye que la etapa de entrada tiene un filtro en frecuencias del mismo tipo que la de salida.

Luego de realizar diversas mediciones en frecuencia, con barridos que a la vez variaban la intensidad de la señal de alimentación, observamos que si variábamos el volumen de grabación y el de salida en la aplicación base de Windows, la señal cambiaba. Esto se debe a que PyAudio (PortAudio) trabajan en una capa superior a la API de Windows, manejando a esta.

Observamos que la señal de salida de nuestra PC en particular podía emplear toda la potencia sin distorsionar la señal, pero no se podía grabar con una señal alta en el LINE-IN ya que la señal se recortaba y distorsionaba. En ese punto comenzamos a trabajar con señal de máxima amplificación a la salida y baja amplificación a la entrada.