Referencia para la App *Lock-in+PID Harmonic* en *RedPitaya*

Laboratorio 4

`Lock-in+PID Harmonic` es una App para el entorno de Red Pitaya – STEMLab 125-14 (cuya documentación está disponible en `https://redpitaya.readthedocs.io/`). STEMLab consiste en un dispositivo embebido que integra electrónica de E/S (dos ADC y dos DACs de 14 bits para ± 1 V y 125 MSa/s) vinculada a una capa de electrónica programable FPGA y a un procesador que corre un sistema operativo Linux. Las aplicaciones diseñadas para ese entorno implementan la electrónica de procesamiento de la E/S y ofrecen una interfaz web que permite controlar los parámetros de la App y visualizar datos. Esta interfaz es suministrada a través de un servidor web que corre dentro del propio dispositivo, por lo que para acceder a cualquier aplicación es necesario conocer la dirección de red del mismo.

En particular, la aplicación `Lock-in+PID Harmonic` implementa la electrónica de adquisición de un amplificador Lock-in, emulando parcialmente el diseño del Stanford SR830. Tambien incorpora otras características, como amplificadores PID, un control de barrido, etc. Se puede descargar de: `https://marceluda.github.io/rp_lock-in_pid/Derivated/`.

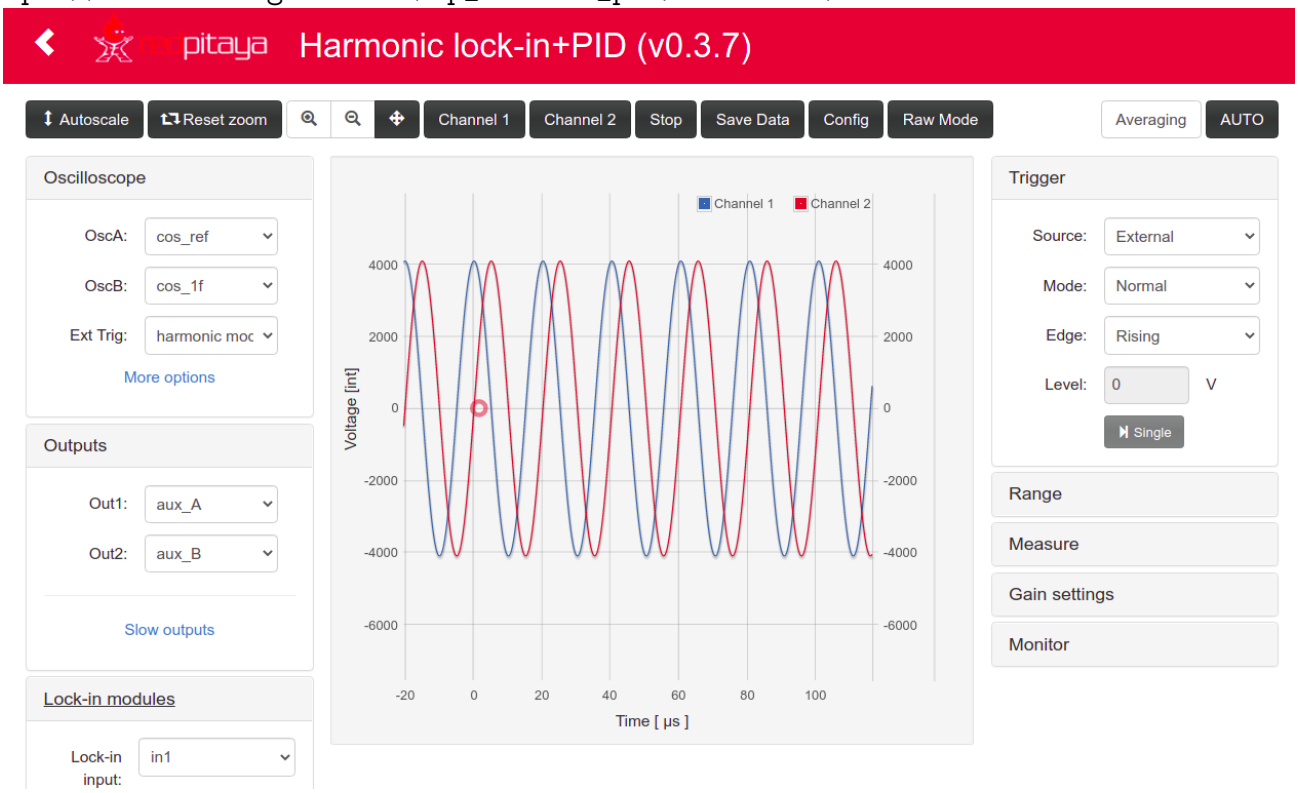

Figura 1: Interfaz web de `Lock-in+PID Harmonic`

1. Modo de operación

La aplicación contiene una serie de señales que pueden ser redireccionadas a distintos "instrumentos" o a salidas DAC usando de-multiplexores.

Estas son las señales más relevantes:

- **in1**,**in2**: Entrada de los ADC 1 y 2.
- **out1**,**out2**: Salida de los DAC 1 y 2. Se les puede asignar multiples señales.
- **oscA**,**oscB**: Señales de entrada del osciloscopio. Son las que se ven en la interfaz web.
- **signal**: Señal de entrada del amplificador lock-in. Puede ser configurada para ser **in1**, **in2** o **in1-in2**.
- **cos_ref**,**sin_ref**: Son señales armónicas en cuadratura que se usan para la demodulación lock-in.
- **cos_1d**,**cos_2f**,**cos_3f**: señales armónicas con la misma frecuencia (1f), el doble (2f) o el triple (3f) que **cos_ref**. Pueden ser desfasadas respecto a las de referencia condifurando el parámetro de fase.
- **Xo**: Es la demodulación lock-in en fase con **cos_ref**.
- **Yo**: Es la demodulación lock-in en cuadratura con **cos_ref** (en fase con **sin_ref**)

La interfaz web de la App permite seleccionar la fuente de cualquiera de estas señales mediante listas desplegables. Los diferentes instrumentos del diseño pueden ser controlados mediante parámetros, cuyos valores son números enteros.

Algunos de los parámetros mas relevantes:

- **oscA_sw**,**oscB_sw**: Controla qué señales son las fuentes de **oscA** y **oscB**.
- **trig_sw**: Controla de donde proviene el trigger externo del osciloscopio (solo para visualización web).
- **signal_sw**: Controla la fuente de **signal_i**, la señal de entrada del amplificador Lock-in.
- **out1_sw**,**out2_sw**: Controla qué señales saldrán por los DAC.
- **sg_amp0**: Controla el valor de amplificación de las señales demoduladas **Xo** e **Yo** que se ven el el osciloscopio. El valor de amplificación será: $x2^{\texttt{sg\_amp0}}$
- **lpf_F0**: Controla el filtro pasabajos de **Xo** e **Yo**.
- **gen_mod_phase**: Controla la fase entre **cos_ref** y **cos_1f**. Su valor puede ir entre 0 y 2519.
- **gen_mod_hp**: Controla el periodo de las señales de modulación. Puede ir entre 0 y 16383. Los detalles de cómo funciona este parámetro están en la documentación web de un App similar a esta: `https://marceluda.github.io/rp_lock-in_pid/TheApp/instruments/instruments_05_modulation/`.
- **aux_A**,**aux_B**: Son valores constantes que pueden ser asignados a las salidas. Pueden ser entre -8192 y 8191. Si se los asigna de una de las salidas, la escala en volt es: $8192 \ \texttt{int} \equiv 1$ Volt.

2. Control remoto mediante Python

La App puede ser controlada de forma remota mediante un canal de comunicación HTTP. Se provee una clase de Python que facilita el acceso a leer y configurar parámetros, así como realizar adquisición de datos. La librería se llama **control_finn**. A continuación se muestra un ejemplo de uso:

```
import numpy as np
from matplotlib import pyplot as plt
from time import sleep,time

from control_finn import RedPitayaApp, reg_labels

# Creación del objeto rp para controlar la App de lock-in
rp  = RedPitayaApp('http://rp-f00a3b.local/lock_in+pid_harmonic/')

# "rp-f00a3b.local" debe ser reemplazado por la dirección
# correspondiente a la RedPitaya de cada usuario
```

A continuación se muestra un ejemplo de cómo acceder a los parámetros y valores de las señales con este objeto:

```
# Ejemplo de escritura de parámetros
rp.lock.aux_A          = -100
rp.lock.gen_mod_hp     = 4
rp.lock.gen_mod_phase = 630

# ejemplo de lectura
print(f"El valor actual de aux_A es: {rp.lock.aux_A}")
print(f"El valor actual de Xo es: {rp.lock.Xo_28}")
```

Debido a restricciones de la electrónica programable, tanto los valores de los parámetros como de las señales están en números enteros, cuya conversión a Volt, segundos o Hz dependen de escalas e implementaciones que no se tratarán en detalle aquí. Para facilitar el proceso de configuración y adquisición se incorporaron algunas funciones de ayuda, que se tratarán a continuación.

3. Funciones para facilitar el control remoto y adquisición

El siguiente conjunto de funciones permite controlar los parámetros de los intrumentos mediante números enteros, pero obtener a la vez cuál es el valor fijado en el parámetro en unidades físicas.

rp.set_lpf(tau_exponent)

Función para configurar los filtros pasabajos. Devuelve el valor de tiempo fijado. El tiempo carcterístico del pasabajos será: $\tau = 2^{14+\texttt{tau\_exponent}} \cdot 8\,\text{ns}$.

Ejemplo:

```
tau = rp.set_lpf(15, order=2)  # LPF de 262.144 us de orden 2
```

rp.get_lpf()

Devuelve el tiempo característico y el orden del filtro pasabajos actual.

Ejemplo:

```
tau, order = rp.get_lpf()
```

rp.get_XY()

Devuelve los valores demodulados **X** e **Y**, en Volts.

Ejemplo:

```
X, Y = rp.get_XY()
```

rp.set_freq(val)

Configura el periodo/frecuencia de las funciones **cos_ref** y **sin_ref**. `val=0` corresponde a la frecuencia más alta. Devuelve la frecuencia configurada en Hz.

Ejemplo:

```
frecuencia = rp.set_freq(8) # configura la frecuencia en 5511.46 Hz
```

rp.set_modulation_amplitud(val,ch=."out1")

Configura que a la señal asignada a **out1** se le sume una señal armónica de amplitud **val** (escala **8219 int**$\equiv$ 0,5 V). Devuelve la amplitud fijada en Volts.

Ejemplo:

```
A = rp.set_modulation_amplitud( 4096) # modulación con amplitud de 0.25 Volts
```

rp.set_phase(val)

Configura la relación de fase entre **cos_ref** y **cos_1f**. Escala: 2520 `int` $\equiv 360°$. Devuelve el valor fijado en grados.

Ejemplo:

```
fase = rp.set_phase( 630 ) # fase 90°
```