Trabajo Práctico Final
Control Digital - MSE

Alumno: Carlos Herrera Trujillo

Repositorio: https://github.com/CarlosSHT/GitHub_MSE_CDI/tree/main/Entrega_Final

I. Propuesta de Trabajo

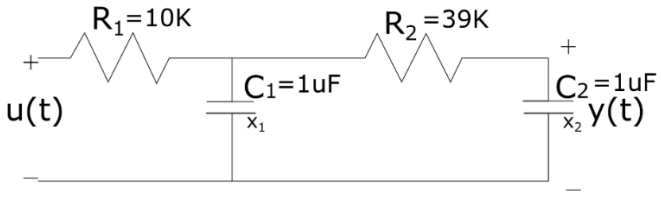
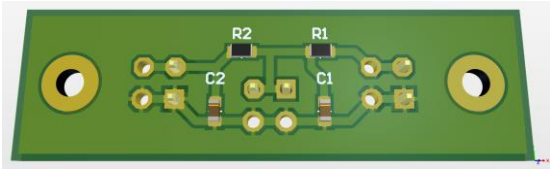


Figura 1: Circuito rc-rc

1. **Armado del circuito**: Para el armado del circuito los valores de resistencia y capacidad serán dados a cada alumno/a de manera particular.

   En la Figura 2 se observa el diseño del circuito rc-rc en Altium haciendo uso de los componentes disponibles. Posteriormente se fabricó para su uso, obteniendo el circuito físico mostrado en la Figura 3.



Diseño del circuito

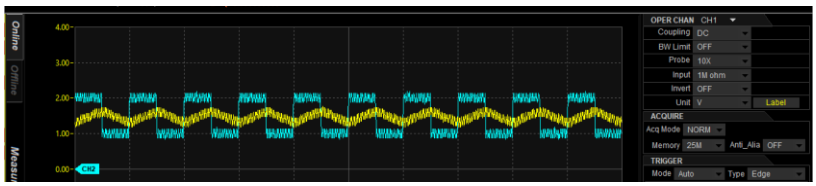
Circuito físico

2. **Análisis:** Generar una señal cuadrada que vaya de 1V a 2V con una frecuencia de 10Hz como señal de referencia (utilizar el DAC). **Sin aplicar control**, obtener la respuesta al escalón del sistema a lazo abierto. Medir el **tiempo de subida** (t_r) y mostrar los **gráficos** que se generan.

Archivos: `t01_UDPreceiver.py, t02_CSVsignalsplot.py`

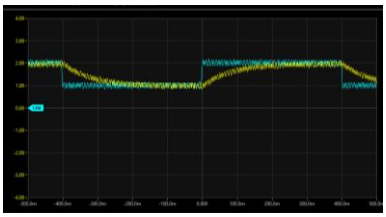
Frecuencia de oscilación 10 Hz (100ms)

Debido que a una frecuencia de 10 Hz no es posible obtener el tiempo de subida, se reducirá la frecuencia de oscilación de la señal de entrada a fin de poder observar mejor el tiempo de subida de la señal resultante “y(t)” (señal amarilla).

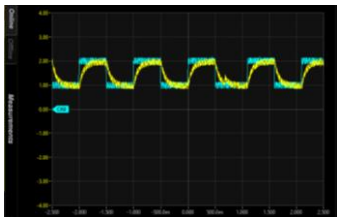


Resultados con frecuencia de onda cuadrada de 10 Hz

En las siguientes figuras se puede observar que a una frecuencia de oscilación de 1 Hz y de 1.25 Hz se puede visualizar el tiempo de subida, por tanto, para la presente pregunta se seleccionó la frecuencia de 1 Hz para la obtención del tiempo de subida.

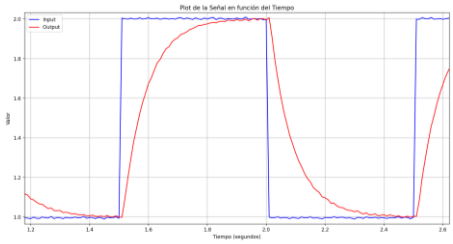


Frecuencia de oscilación 1.25 Hz (800ms)



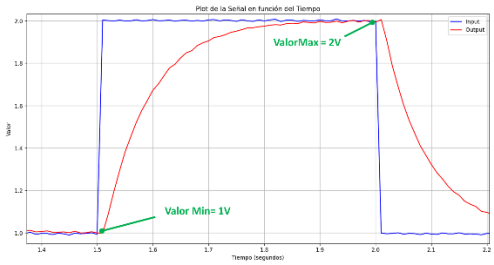
Frecuencia de oscilación 1 Hz (1000ms)

Además, a fin de tener una buena visualización, se utilizará una frecuencia de muestreo de 100 Hz en las dos entradas de ADC, una para la señal generada con el DAC y otro para la salida del circuito RCRC, ambos valores obtenidos, en formato enteros de 16 bits, se agruparán en paquetes para ser enviados por ethernet utilizando el protocolo UDP. Por último, se utilizará Python para la visualización de las señales en vivo.



Gráfica en Python

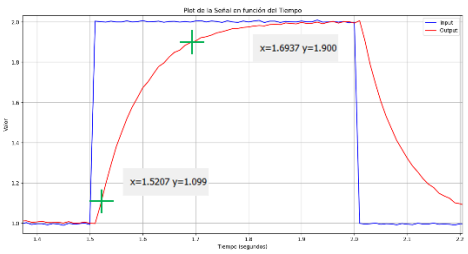
El tiempo de subida se calcula como el tiempo que tarda la señal en ir del 10 % al 90 % de su valor ($t_r = t_{90\%} - t_{10\%}$).



Gráfica en de la señal en valor mínimo y máximo

$valor_{90\%} = (2-1) \times 0.1 + 1 = 1.1$

$valor_{90\%} = (2-1) \times 0.9 + 1 = 1.90$



$t_{10\%} = 1.5207\ seg$

$t_{90\%} = 1.6937\ seg$

$$t_r = 1.6937 - 1.5207$$
$$t_r = 0.173\ segundos$$

3. **Identificación:** Determinar el valor de los polos del sistema mediante identificación, considerando al sistema como una caja gris.

$$G(s) = Y(s)/U(s)$$

Modelo matemático:

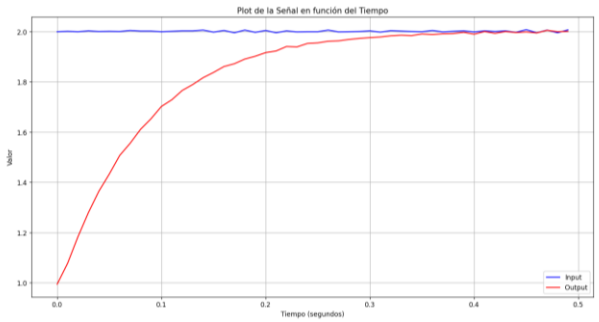
$$\frac{Y(s)}{U(s)} = \frac{1}{R_1C_1R_2C_2s^2 + [R_1C_1 + R_1C_2 + R_2C_2]s + 1}$$

Se procedió a realizar la identificación con el algoritmo "offline" LS **"TPF_p2_Identificacion.m"** y también la función **"systemIdentification"** de Matlab para los siguientes tipos de entrada:

- Señal Cuadrada
- PRBS
- Ruido RNG

Señal cuadrada:

Gráfica respuesta al escalón



Archivo:
TPFrcrcDatos_002.csv
Frecuencia de muestreo:
100 Hz
G(s) teórico:
$G(s) = \frac{1}{0.00039s^2 + 0.059s + 1}$
H(z) teórico:
$H(z) = \frac{0.08058z + 0.04887}{z^2 - 1.091z + 0.2203}$

Se obtuvieron los siguientes resultados:

Orden	Numerador (z)	Denominador (z)	Error
1	[0.415, -0.307]	[1,-0.893]	0.0683
2	[0.0912, -0.212,0.280]	[1,-0.479,-0.362]	3.158
3	[-0.096,-0.407,0.194,0.546]	[1,0.074,-0.756,-0.0813]	10.390

De acuerdo a los resultados obtenidos se desprende que la planta correspondería a orden 1, no obstante, se conoce que la planta es de orden 2. Por tanto, utilizar una señal correspondiente a un escalón para la identificación es insuficiente.

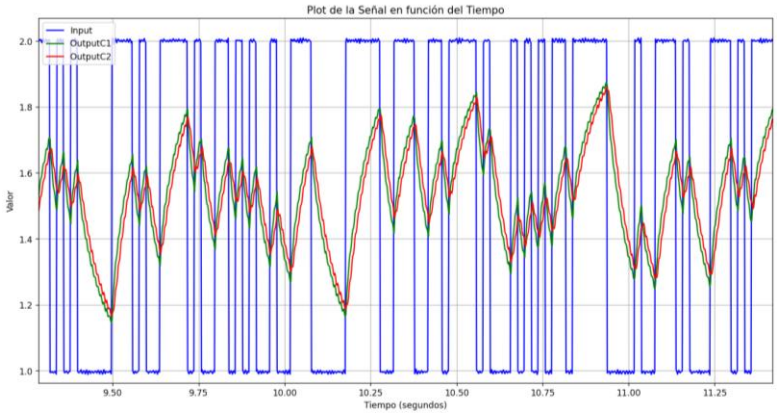
Señal PRBS:

De acuerdo a la teoría, se debe seleccionar una frecuencia de muestreo adecuada. En primer lugar, debe poderse muestrear de manera efectiva la señal en el tiempo de subida (0.173 segundos).
En segundo lugar, la frecuencia de muestreo del ADC debe ser lo suficientemente mayor para poder muestrear el periodo mínimo del PRBS.
Utilizando el algoritmo de Python (archivo "t03_PRBSgenerator.py") se generó una señal PRBS con las siguientes características:

```
Periodo de muestreo proyectado PRBS: 0.01922222222222222
Periodo de muestreo mínimo seleccionado PRBS: 0.02
Máximo número de muestras por PULSO: 10
Tiempo máximo de pulso 0.2 > tiempo de subida 0.173
Frecuencia de muestreo PRBS: 50.0
Longitud total de la secuencia (n° muestras): 1023
Tiempo total del ensayo: 20.46
```

Por tanto, la frecuencia de muestreo seleccionada será de 500 Hz.

Gráfica salida con entrada PRBS



Archivo:
TPFrcrcDatos_003.csv
Variable en C:
prbs_1
Frecuencia de muestreo:
500 Hz
G(s) teórico:
$G(s) = \frac{1}{0.00039s^2 + 0.059s + 1}$
H(z) teórico:
$H(z) = \frac{0.004644z + 0.004199}{z^2 - 1.73z + 0.7389}$

Se obtuvieron los siguientes resultados con el algoritmo LS:

Orden	Numerador (z)	Denominador (z)	Error
1	[-0.002, 0.023]	[1, -0.980]	16.6975
2	[-5.427e-04, 0.007, 0.012]	[1,-1.198, 0.216]	11.9850
3	[-8.260e-04,0.007, 0.008, 0.012]	[1,-1.012, 0.159, -0.121]	11.7885

Por otro lado, al utilizar la herramienta de Matlab se obtuvo el siguiente resultado al definir que el sistema cuenta con 2 polos y 1 cero:

```
From input "u1" to output "y1":
        4177
--------------------
s^2 + 328.8 s + 4179
```

Al normalizar la ecuación el resultado final es:

$$G(s) = \frac{1}{0.0002s^2 + 0.0787s + 1.0005}$$

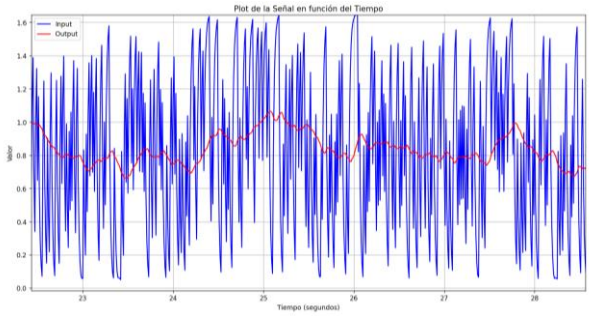
Utilizando una señal de entrada PRBS se obtuvieron errores de mayor valor, no obstante, el numerador y denominador de orden 2 tienen un valor más cercano a lo calculado de forma teórica. Además de dicha señal PRBS se probaron otras variantes modificando la amplitud, duración del ensayo, frecuencia de muestreo. Sin embargo, los resultados obtenidos no tienen un valor de error bajo.
De otro modo, al utilizar la función de Matlab se pudo obtener una función de transferencia con valores cercanos a la función teórica.

Señal NOISE:

Para el presente ensayo se utilizó el generador de código pseudoaleatorio que posee el módulo DAC del microcontrolador STM32F429.

Tomando muestras por aproximadamente 30 segundos a una frecuencia de muestreo de 500 Hz. Así mismo, la amplitud del ruido generado fue de 1.65 V aproximadamente.

Gráfica salida con entrada RNG



Archivo:
TPFrcrcDatos_004.csv
Frecuencia de muestreo:
500 Hz
G(s) teórico:
$G(s) = \frac{1}{0.00039s^2 + 0.059s + 1}$
H(z) teórico:
$H(z) = \frac{0.004644z + 0.004199}{z^2 - 1.73z + 0.7389}$

Se obtuvieron los siguientes resultados:

Orden	Numerador (z)	Denominador (z)	Error
1	[5.194e-05,0.012]	[1, -0.988]	43.3634
2	[1.634e-04, 0.007, 0.008]	[1, -1.280, 0.294]	14.0723
3	[1.257e-04, 0.007, 0.007, 0.006]	[1, -1.082, 0.154, -0.052]	12.9128

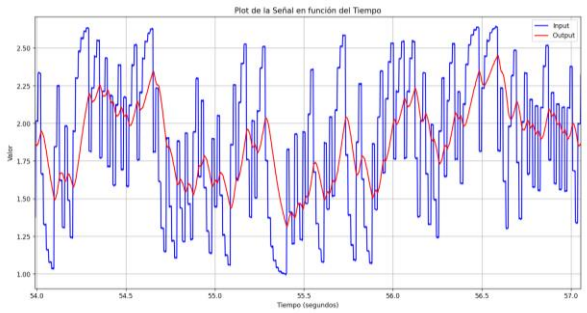
Podemos observar que los coeficientes de numerador y denominador de orden 2 también tienen valores cercanos a lo calculado teóricamente, sin embargo, su error sigue siendo alto.

Señal NOISE 2:

Debido a que la generación de la señal RNG del módulo DAC depende de los eventos de interrupción generados por un TIMER (Timer 7) se realizó una prueba adicional disminuyendo la frecuencia de actualización del *timer* a fin de obtener una señal escalonada variable.

Para el presente ensayo se realizó una grabación de la señal por 2 minutos. Además, entre las características de la señal se encuentra una amplitud de 1.65 V cuyo punto mínimo será de valor 1V.

Gráfica salida con ruido de entrada escalonada



Archivo:
TPFrcrcDatos_005.csv
Frecuencia de muestreo:
500 Hz
G(s) teórico:
$G(s) = \frac{1}{0.00039s^2 + 0.059s + 1}$
H(z) teórico:
$H(z) = \frac{0.004644z + 0.004199}{z^2 - 1.73z + 0.7389}$

El resultado obtenido de coeficientes fue:

Numerador: [0.0011 0.0076 0.0129]

Denominador: [1.0000 -1.0856 0.1072]

La función obtenida con la herramienta de Matlab es la siguiente

```
From input "u1" to output "y1":
          4637
--------------------
s^2 + 375.8 s + 4639
```

Normalizando la función de transferencia se obtiene:

$$G(s) = \frac{1}{0.000216s^2 + 0.08104s + 1.00043}$$

Con las identificaciones realizadas con las señales PRBS y NOISE2 y la herramienta de Matlab, se establece como función transferencia experimental la siguiente función:

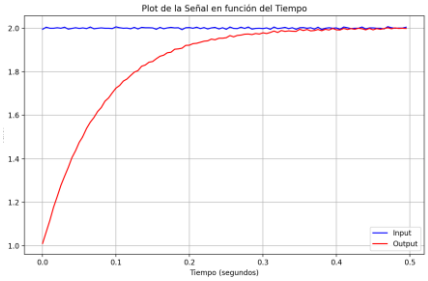
$$G(s) = \frac{1}{0.000216s^2 + 0.08104s + 1.00043}$$

4. **Controlador PID:** Diseñar y aplicar un control PID para cumplir con las siguientes especificaciones:

Archivos: `t05_ControladorPID_LA.py`

A. Tener error de estado estacionario nulo.
B. Mejorar el tiempo de subida, tr, un 30 % con respecto a la respuesta a lazo abierto.
C. Tener un sobrepico menor al 8 %.

Se utiliza el método respuesta al escalón Ziegler – Nichols, así mismo, los valores en formato decimal del ADC se convierten a voltaje (3.3Vref).
Dado la señal obtenida del microcontrolador, con muestreo en la salida del DAC, se procede a utilizar la sección que sería la respuesta al escalón.

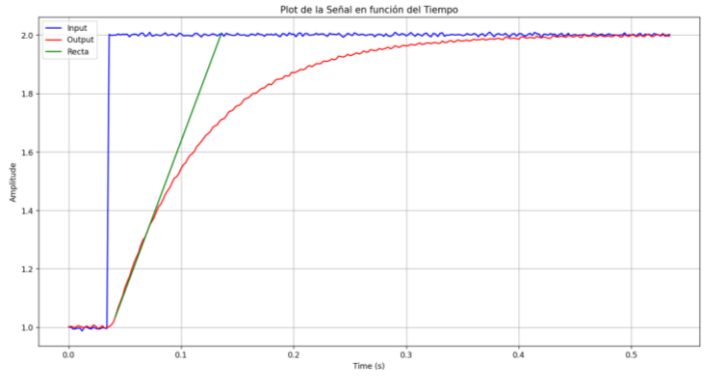


Se procede a dibujar la recta tangente con los valores de

$$x1 = 0.0414 \qquad y1 = 1.034$$
$$x2 = 0.1352 \qquad y2 = 2$$

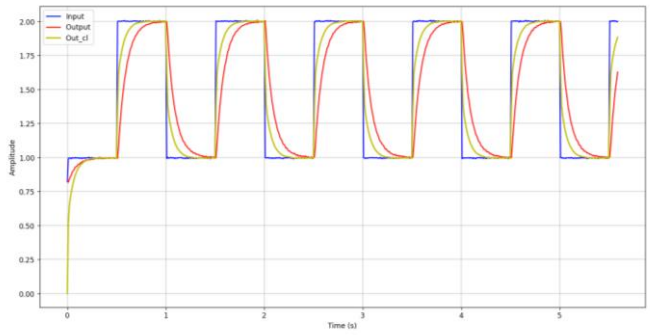
Obteniendo de esta forma:

$$L = x1$$
$$T = x2 - x1$$



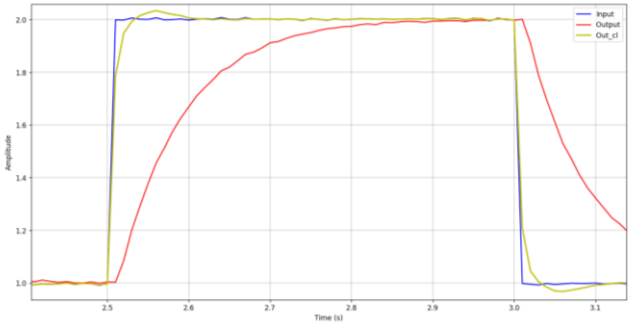
$$A = B = 2$$

Una vez implementado el controlador PID se tiene el siguiente resultado, donde la señal controlada se muestra en color amarillo y tiene un valor máximo de 1.9234



El valor de Kp es
2.7188
El valor de Ki es
32.8362
El valor de Kp es
0.0563

Una vez sintonizado los valores de Kp, Ki y Kd se obtiene la nueva señal controlada (amarillo), cuyo valor máximo que se obtiene es de 2.0337V (3.4% mayor), con un error de estado estacionario nulo.



El valor de Kp es
16.313
El valor de Ki es
656.7248
El valor de Kd es
0.2701

Además, el nuevo tiempo de subida es de 0.0165 segundos siendo una mejora de más de 90% con respecto al anterior tiempo de subida.

5. **Período de Muestreo:** Seleccionar un tiempo de muestreo que cumpla con todas las restricciones prácticas vistas en clase.

A fin de poder seleccionar un buen periodo de muestreo en el diseño del filtro PID, se tuvo en cuenta el tiempo de subida de la señal con el controlador PID utilizado anteriormente.

Frecuencia de muestreo mínima: Dado un tiempo de subida de 20 milisegundos podemos tener como mínimo una frecuencia de muestreo de 200 Hz (20 milisegundos / 4 muestras)

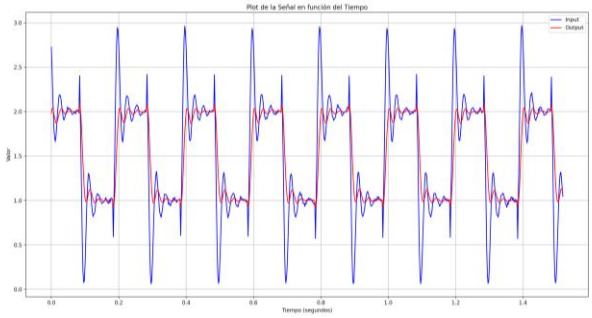
Frecuencia de muestreo máxima: Para lo cual se debe tener en cuenta el tiempo que demora en muestrear, el tiempo de procesamiento y el tiempo del conversor digital/analógico.
El periodo mínimo de muestreo será 20 veces la sumatoria de todos los tiempos mencionados.

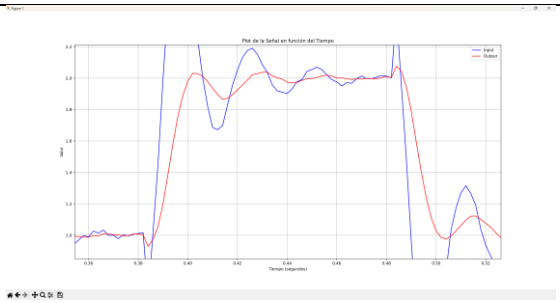
En el microcontrolador STM32 F429 una vez obtenido la suma de todos los retardos provocados, estos se suman y el periodo mínimo de muestreo será 20 veces el valor de dicha sumatoria.

$$T(ADC) = 84\ ciclos\ de\ reloj$$
$$T(Procesamiento) = Medido\ en\ ciclos\ de\ reloj$$
$$T(DAC) = microsegundos$$

Resultados en microcontrolador

Se realizó una sintonización con valores de Kp, Ki y Kd

Frecuencia de muestreo : 100 Hz	
Kp = 2.9	
Ki = 44	
Kd = 0.1	
Beta = 1	
N = 20	



tr = 0.398-0.389 = 0.009 => 9 milisegundos

6. Realimentación de estados:

- Realizar una realimentación de estados para cumplir con las especificaciones 4b y 4c, midiendo x1 y x2.

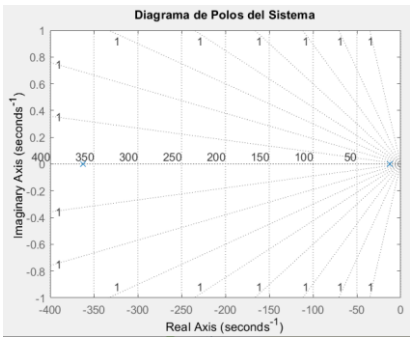
Proyecto C "c04_Avance02_TPFinal_SS"

Utilizando la función de transferencia identificada se procede a realizar el calculo de los polos, traslación de polos, matrices de espacio de estados, K y K0 (ganancia^{-1}).

$$G(s) = \frac{1}{0.000216s^2 + 0.08104s + 1.00043}$$

Así mismo, se utiliza una frecuencia de muestreo de 200 Hz. A fin de que la conversión analógico-digital no consuma tiempo de CPU del microcontrolador, se hizo uso del módulo ADC con DMA lo que permite implementar una frecuencia de muestreo alta.

Como primer paso es necesario obtener los polos originales del sistema, es así que se utiliza el algoritmo de MATLAB llamado "t06_PolePlacement_v2.m". A continuación, se puede visualizar la ubicación de los polos en H(s) y los valores de los mismos.



Polos en tiempo continuo:

(-362.4533 + 0j)
(-12.7781 + 0j)

Polos en tiempo discreto:

(0.1633+0j)
(0.9381+0j)

Polos trasladados 30%

(0.0948+0j)
(0.9203+0j)

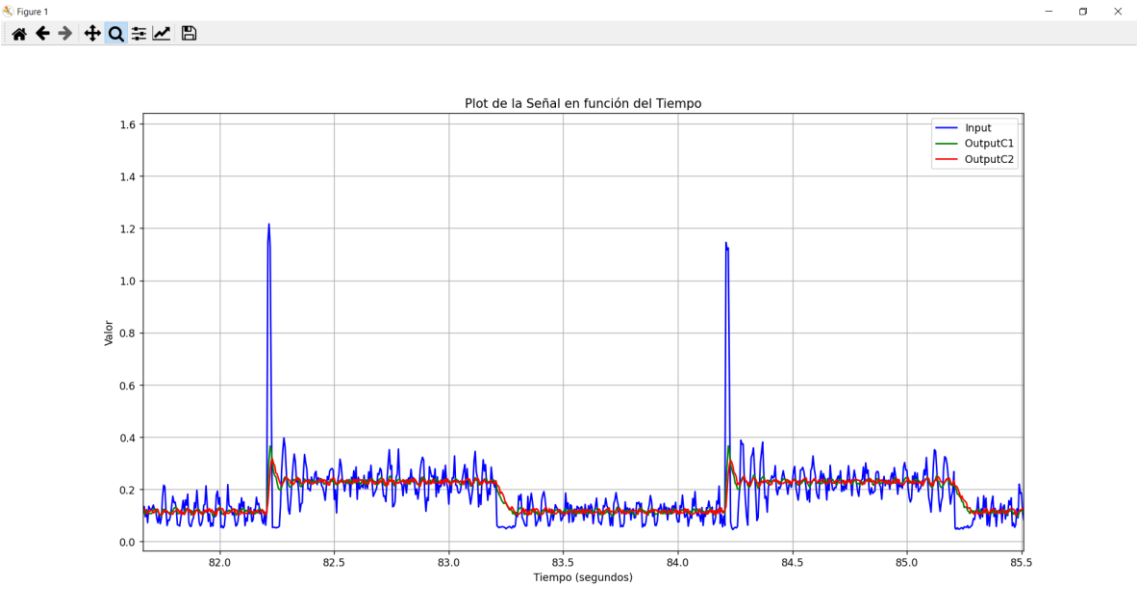
Una vez obtenido los polos se procede a calcular el valor de K con las matrices de espacio estado:

K = [4.1231 3.5563]

Así mismo, se calcula el valor de K0 a fin de poder aplicar la ganancia.

K0 = [1.3937]

El resultado obtenido requiere una mayor verificación del controlador debido a que, aunque posee la forma, se observa que la amplitud no es la requerida.



- Ahora medir solamente la salida y = x2, realizar un observador y una realimentación de estados.