

Trabajo Práctico Final

Control Digital - MSE

Alumno: Carlos Herrera Trujillo

Repositorio: https://github.com/CarlosSHT/GitHub_MSE_CDI/tree/main/Avance02

I. Propuesta de Trabajo

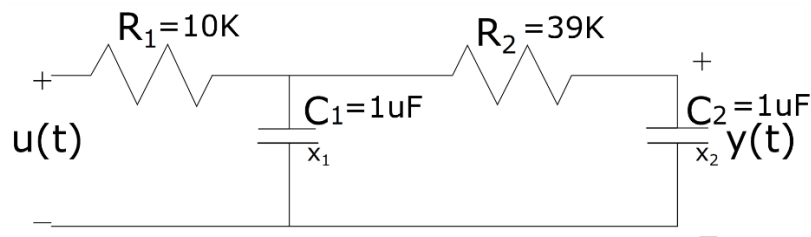


Figura 1: Circuito rc-rc

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.

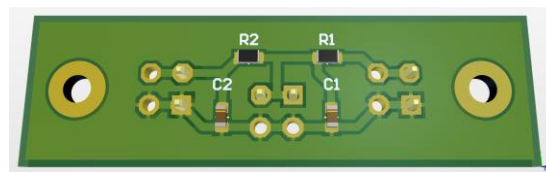


Figura 2: Diseño del circuito

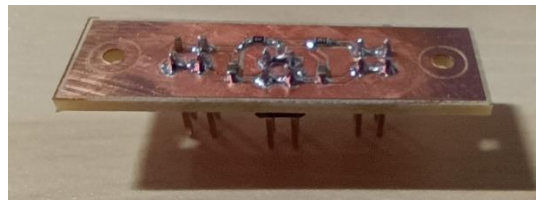


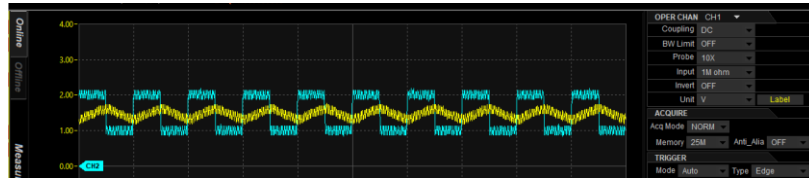
Figura 3: Circuito físico

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

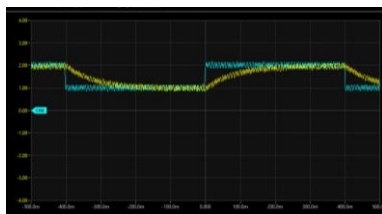
Frecuencia de oscilación 10 Hz (100ms)

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).

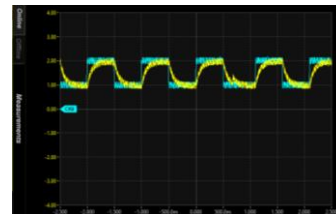


Resultados con frecuencia de onda cuadrada de 10 Hz

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.

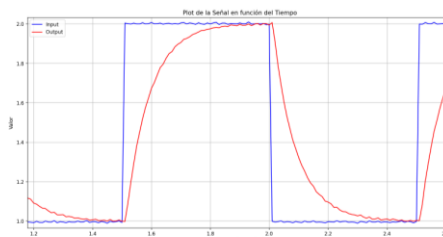


Frecuencia de oscilación 1.25 Hz (800ms)



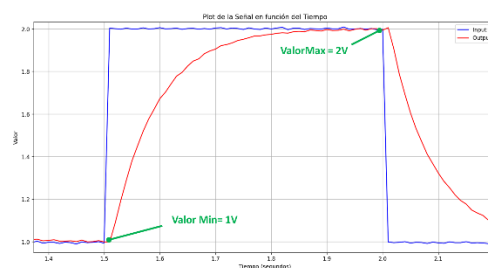
Frecuencia de oscilación 1 Hz (1000ms)

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

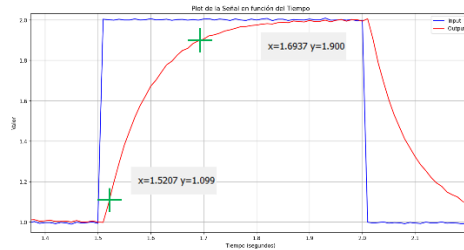
El tiempo de subida se calcula como el tiempo que tarda la señal en ir del 10 % al 90 % de su valor ($t_r = t_{90\%} - t_{10\%}$).



Gráfica en de la señal en valor mínimo y máximo

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

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



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

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

$$t_r = 1.6937 - 1.5207$$

$$t_r = 0.173 \text{ segundos}$$

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:

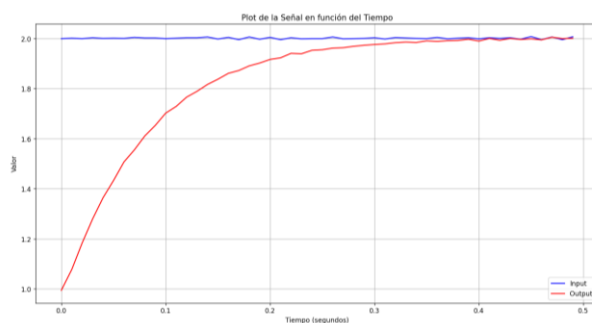
$$\frac{Y(s)}{U(s)} = \frac{1}{R_1 C_1 R_2 C_2 s^2 + [R_1 C_1 + R_1 C_2 + R_2 C_2]s + 1}$$

Se procedió a realizar la identificación con el algoritmo “offline” LS (“TPF_p2_Identificacion.m”) 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.

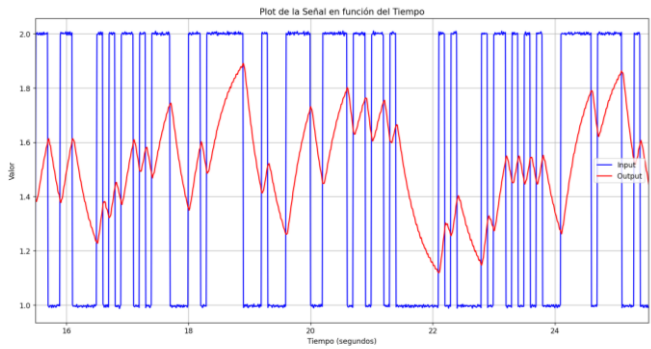
Señal PRBS:

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 obtuvieron los siguientes resultados

Periodo de muestreo proyectado PRBS: 0.0192222222222222
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 (nº 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
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	[-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

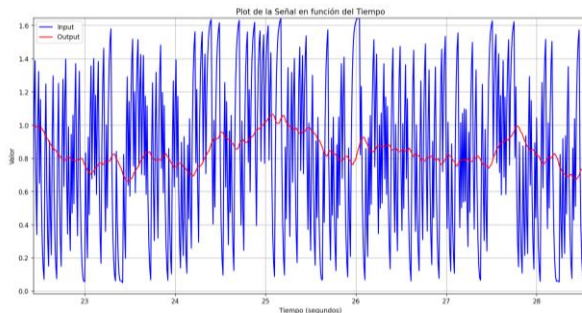
Utilizando una señal de entrada PRBS se obtuvieron errores de mayor valor, no obstante el numerador y denominador de orden 2 tienen un valor más cercano a lo calculado de forma teórica. Además de dicha señal PRBS se probaron otras variantes modificando la amplitud, duración del ensayo, frecuencia de muestreo. Sin embargo, los resultados obtenidos no tienen un valor de error bajo.

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

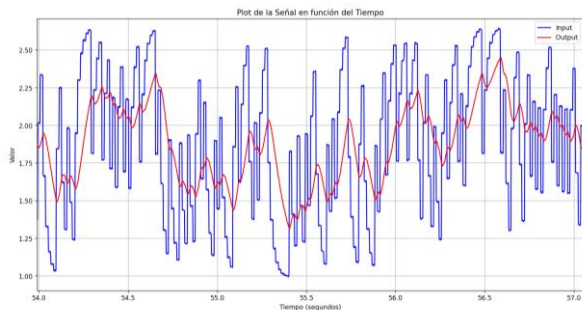
Podemos observar que los coeficientes de numerador y denominador de orden 2 también tienen valores cercanos a lo calculado teóricamente, sin embargo, su error sigue siendo alto.

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]

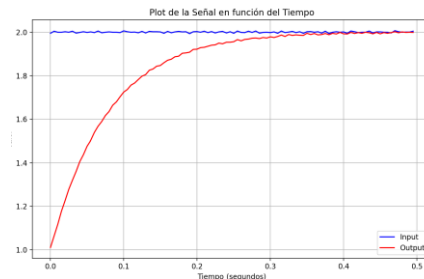
4. **Controlador PID:** Diseñar y aplicar un control PID para cumplir con las siguientes especificaciones:

Archivos: t05_ControladorPID_LA.py

- A. Tener error de estado estacionario nulo.
- B. Mejorar el tiempo de subida, t_r , 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

$$x1 = 0.0414$$

$$y1 = 1.034$$

$$x2 = 0.1352$$

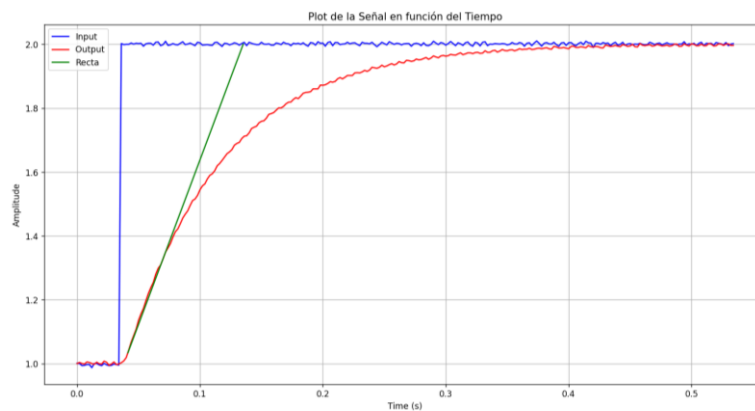
$$y2 = 2$$

Obteniendo de esta forma:

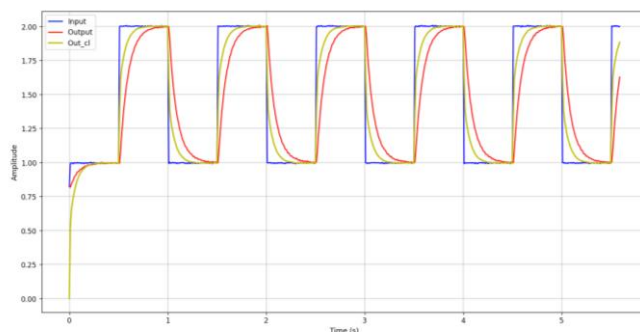
$$L = x1$$

$$T = x2 - x1$$

$$A = B = 2$$

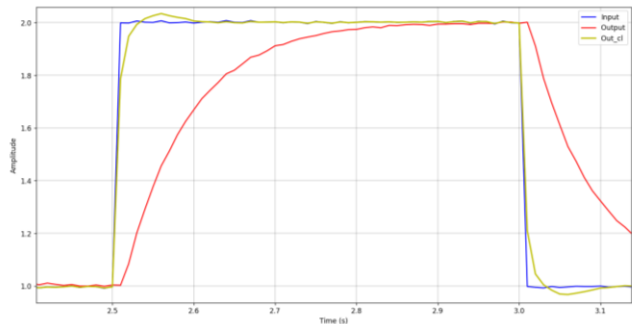


Una vez implementado el controlador PID se tiene el siguiente resultado, donde la señal controlada se muestra en color amarillo y tiene un valor máximo de 1.9234



El valor de K_p es
2.7188
El valor de K_i es
32.8362
El valor de K_d es
0.0563

Una vez sintonizado los valores de K_p , K_i y K_d 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 K_p es
16.313
El valor de K_i es
656.7248
El valor de K_d es
0.2701

Además el nuevo tiempo de subida es de 0.0165 segundos siendo una mejora de mas 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 no sea demasiado corto.

Frecuencia de muestreo mínima: Con 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.

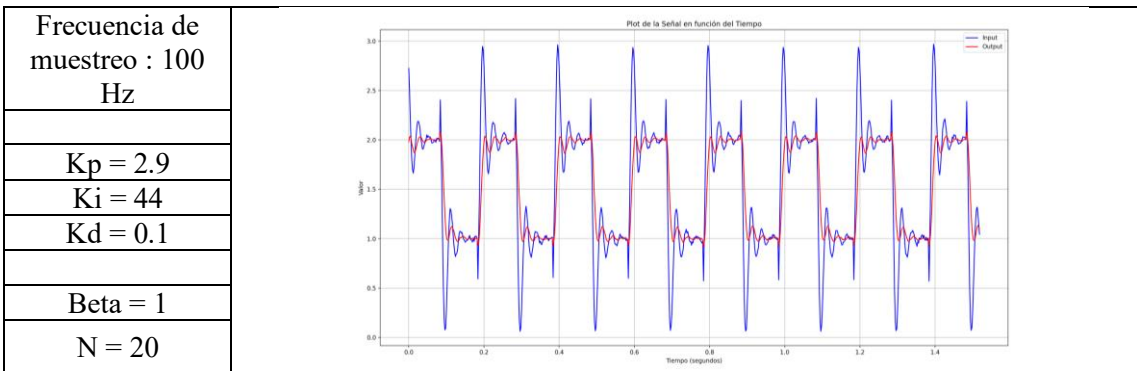
$$T(ADC) = 84 \text{ ciclos de reloj}$$

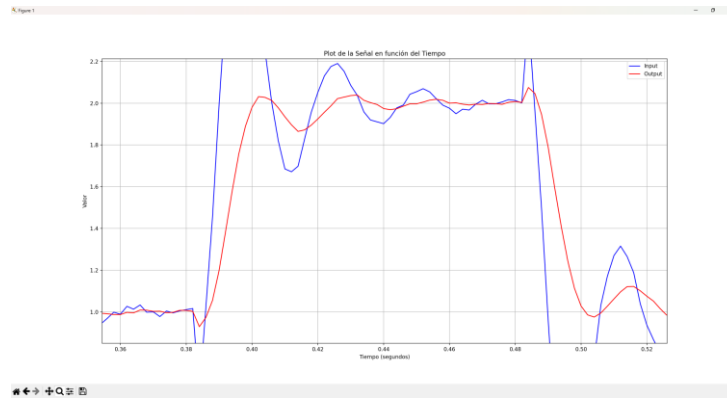
$$T(\text{Procesamiento}) = \text{Medido en ciclos de reloj}$$

$$T(DAC) = \text{microsegundos}$$

Resultados en microcontrolador

Se realizó una sintonización con valores de K_p , K_i y K_d





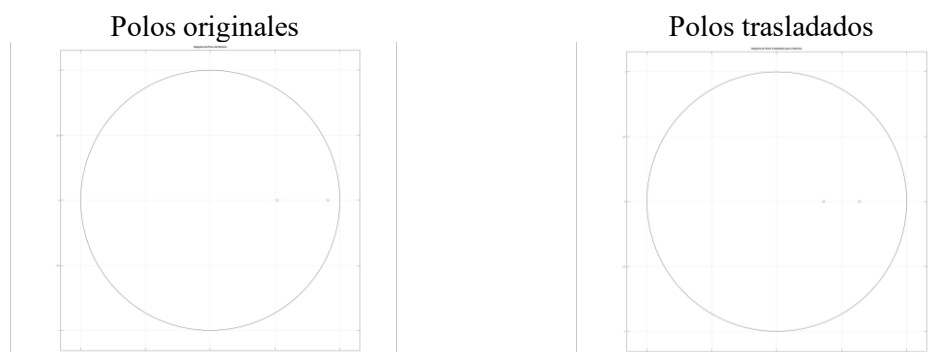
$$tr = 0.398 - 0.389 = 0.009 \Rightarrow 9 \text{ milisegundos}$$

6. Realimentación de estados:

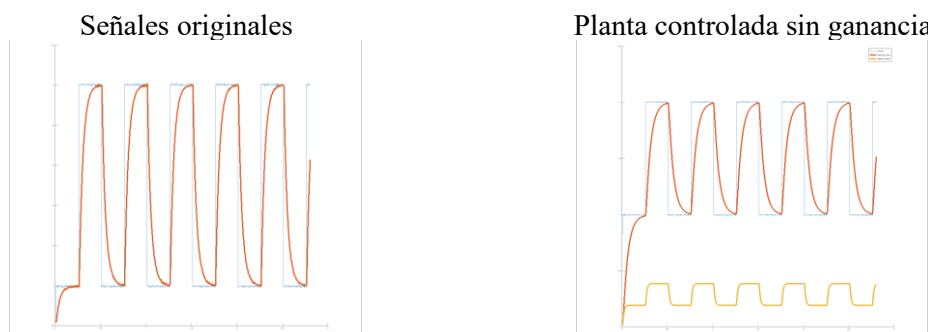
- Realizar una realimentación de estados para cumplir con las especificaciones 4b y 4c, midiendo x_1 y x_2 .

Debido a que no se pudo obtener la función transferencia de la planta de forma experimental, se aplica el algoritmo en octave “t06_PolePlacement.m” haciendo uso de la función de transferencia teórica y utilizando datos reales obtenidos a una frecuencia de 200 Hz.

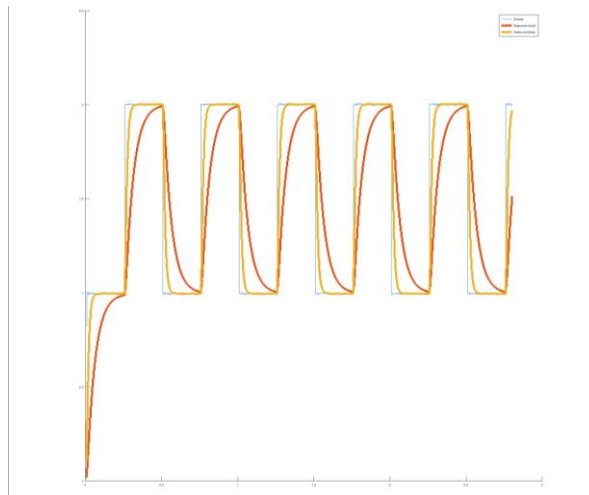
En primer lugar, se observa la ubicación original de los polos, de tal manera que para cumplir con que sean 30% más rápidos, se multiplica por 0.7.



En segundo lugar, una vez trasladados se observa el efecto que tiene sobre la planta, observándose en la izquierda la señal original y en la derecha la señal controlada, pero sin la ganancia requerida.



Posteriormente se realiza el calculo de K_0 para poder aplicar la ganancia y se obtiene lo siguiente, cumpliéndose los puntos 4b y 4c.



- Ahora medir solamente la salida $y = x_2$, realizar un observador y una realimentación de estados.

