Implementación de controladores PID fraccionales en las plataformas STM32-Discovery y Arduino a partir de SIMULINK/MATLAB: parte I

JESÚS ADOLFO FLORES-ORDEÑANA[1], ISRAEL CERÓN-MORALES[1], CARLOS MUÑIZ-MONTERO[1], LUIS ABRAHAM SÁNCHEZ-GASPARIANO[2], AND ESTEBAN TLELO-CUATLE[3]

[1] *Universidad Politécnica de Puebla, 3er Carril del Ejido "Serrano" S/N, San Mateo Cuanalá, Juan C. Bonilla, Puebla, México. C.P. 72640*
[2] *Benemérita Universidad Autónoma de Puebla, Jardines de San Manuel, Puebla, México. C.P. 72570*
[3] *Instituto Nacional de Astrofísica Óptica y Electrónica, Luis Enrique Erro No. 1, Tonantzintla, Puebla, México. C.P. 72840, jesus.flores@uppuebla.edu.mx, israel.ceron@uppuebla.edu.mx*

Compiled 28 de noviembre de 2017

This work presents two methodologies for the direct implementation, from the block diagram in SIMULINK to reconfigurable hardware, of fractional order PID controllers. These methodologies do not require programming in C code and detailed knowledge of the target hardware. Two examples are provided. In the first one, the speed of a direct current motor is controlled by using a STM32-Discovery development board. In the second, a temperature control with Arduino Uno microcontroller is carried out. The correspondence between the simulation results and the experimental ones validates these methodologies.

OCIS codes: (140.3490) Lasers, distributed feedback; (060.2420) Fibers, polarization-maintaining;(060.3735) Fiber Bragg gratings.

http://dx.doi.org/10.1364/ao.XX.XXXXXX

RESUMEN

Este trabajo se presentan dos metodologías para la implementación en hardware reconfigurable de controladores PID de orden fraccional a partir de diagramas comportamentales realizados en SIMULINK/MATLAB. Con estas metodologías no se requiere de un conocimiento detallado del hardware ni de programación en lenguaje C. Se proporcionan dos ejemplos. En el primero se controla la velocidad de un motor de corriente directa a partir de una tarjeta de desarrollo STM32-Discovery. En el segundo se realiza un control de temperatura con microcontrolador Arduino Uno. La correspondencia entre los resultados de simulación y experimentales validan estas metodologías.

1. INTRODUCCIÓN

La estrategia de control más utilizada en el sector industrial es el control PID, estimándose su utilización hasta en un 90 % de los casos prácticos [1]. Este control realiza acciones proporcionales (P), integral (I) y derivativa (D) a la señal de error de los sistemas en lazo cerrado para satisfacer los requerimientos impuestos en la respuesta transitoria y en estado estable. Además, existe una generalización de orden fraccional de estos controladores (conocida como control ($PI^{\lambda}D^{\mu}$, la cual presenta un desempeño superior [1],[2]. Por ejemplo, es más tolerantes a la presencia de perturbaciones y variaciones en la ganancia de la planta y alcanza la respuesta deseada en menor tiempo [3]. Estas ventajas son consecuencia de la incorporación de dos grados de libertad adicionales: el orden de la derivada μ y el orden de la integral . Los cinco grados de libertad resultantes permiten establecer hasta cinco restricciones de diseño a la respuesta del sistema. Una elección típica de estas restricciones consiste en establecer restricciones al sistema en el margen de fase, margen de ganancia, rechazo a perturbaciones, rechazo al ruido de alta frecuencia e incorporar robustez con respecto a variaciones en la ganancia de la planta [4].

Para satisfacer estas restricciones se utilizan algoritmos de sintonización de los grados de libertad [5]. Desafortunadamente, a pesar de la existencia de herramientas de cálculo numérico especializadas en la sintonización de estos parámetros, tales como el toolbox de MATLAB "Fomcon" [6] o las funciones de MATLAB "fmincon" e "invfreqs" [7], la inexistencia de elementos de circuito con respuesta de orden fraccional ha dificultado la penetración del controlador $PI^{\lambda}D^{\mu}$ en la industria [8]. Se pueden citar solo algunos ejemplos, principalmente académicos. En [8] y [9], se reportaron realizaciones analógicas a partir de funciones de aproximación [1], sin embargo, el hardware resultante es difícil de implementar debido a la utilización de resistencias y capacitores de valores no comerciales e inclusive de convertidores negativos de impedancia. Las realizaciones analógicas con hardware reconfigurable son aún más escasas [10]. También se han propuesto realizaciones digitales con la tarjeta controladora dDS1104 R&D (programable con Simulink) [11] o la tarjeta de adquisición de datos PCL-818H de la compañía PC-LabCard, con su propio toolbox para MATLAB [12].

Considerando esta problemática, en este trabajo se proponen dos metodologías para la implementación digital de controladores $PI^{\lambda}D^{\mu}$ en hardware reconfigurable a partir de diagramas comportamentales realizados en SIMULINK/MATLAB, la primera en la tarjeta de desarrollo STM32F4-Discovery con los programas Simulink-MATLAB R2016a, STM32CubeMX 4.18.0 y

Keil μVision 5, y la segunda en microcontroladores Arduino Uno con los programas Simulink-MATLAB, Arduino (IDE) 1.6.13 y el soporte para Arduino de Simulink/MATLAB. Con el uso de estas metodologías no se requiere de un conocimiento detallado del hardware ni de programación en lenguaje C, lo que podría facilitar la penetración de este controlador en la industria. Para ello, se proporcionan dos ejemplos. En el primero se controla la velocidad de un motor de corriente directa a partir de una tarjeta de desarrollo STM32-Discovery. En el segundo se realiza un control de temperatura con microcontrolador Arduino Uno

2. CONTROLADOR

Una de las definiciones más empleadas para el cálculo de derivadas e integrales fraccionales es la definición de Riemann-Liouville, la cual establece [1].

$$\mathfrak{D}_t^a f(t) = \frac{1}{\Gamma(m-a)} \left(\frac{d}{dt}\right)^m \int_\infty^t \frac{f(\tau)}{(t-\tau)^{a-m+1} d\tau} \quad (1)$$

donde $\alpha \in \mathbb{R}, m-1 < \alpha < m, m \in \mathbb{N} y \Gamma(\bullet)$ es la función Gamma. Para $\alpha > 0, \alpha < 0 y \alpha = 0$ se tienen respectivamente la derivada fraccional, la integral fraccional y la función identidad. Al calcular la transformada de Laplace de (1) con condiciones iniciales iguales a cero resulta

$$\mathfrak{L}\{\mathfrak{D}_t^\alpha f(t)\} = s^{\pm\alpha} F(s) \quad (2)$$

A partir de (2) es posible generalizar el controlador PID al controlador fraccional $PI^\lambda D^\mu$ de la Figura 1, el cual realiza acciones integral y derivativa de órdenes

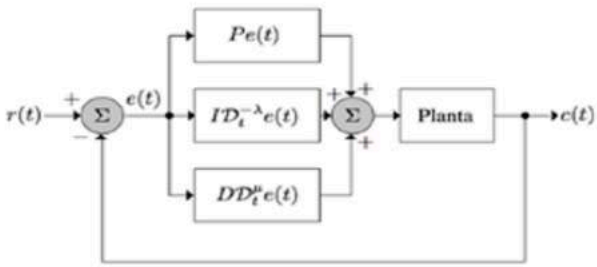

Fig. 1. Control PID de orden fraccional.

fraccionales $\lambda \in \mathbb{R}^+ y \mu \in \mathbb{R}^+$ de la señal de error e(t), además de la acción proporcional. La función de transferencia del controlador se expresa como [5][2]

$$C(s) = \frac{U(s)}{E(s)} = P + \frac{1}{s^\lambda} + Ds^\mu, (\lambda, \mu > 0) \quad (3)$$

3. HARDWARE Y SOFTWARE REQUERIDOS

Para la implementación del controlador $PI^\lambda D^\mu$ con la tarjeta de desarrollo STM32F4-Discovery, se utilizan los programas informáticos Simulink-MATLAB R2016a, STM32CubeMX 4.18.0 y Keil μVision 5. El STM32F4-Discovery permite desarrollar fácilmente aplicaciones con un microcontrolador ARM Cortex-M4 de 32-bits [13]. Éste microcontrolador tiene una memoria Flash de 1-Mbyte, RAM de 192-Kbytes, USB ST-LINK con capacidad de re-enumeración y tres interfaces diferentes; puerto de depuración, puerto de comunicación virtual y almacenamiento masivo. Además, la tarjeta cuenta con salida de alimentación de 3V y 5V, acelerómetro de tres ejes, sensor de audio omnidireccional, micrófono digital, ocho LED's, dos botones pulsadores, software de desarrollo libre y completo el cual incluye una gran variedad de ejemplos, así como librerías estándar. El STM32CubeMX es una herramienta de configuración de software gráfico que permite la generación de código de inicialización C utilizando asistentes gráficos. Posee un conjunto de componentes de middleware (RTOS, USB, TCP / IP y gráficos) [14], reuniendo en un solo paquete todos los componentes de software embebido necesarios para desarrollar una aplicación en microcontroladores de la familia Microelectronics. Estos componentes son portátiles, no sólo dentro de la serie STM32F4, sino también con otras series STM32. El driver STM32CubeF4 es totalmente compatible con el generador de código STM32CubeMX [14]. Por su parte, Keil μVision 5 es la última versión de desarrollo de software para una amplia gama de dispositivos basados en microcontroladores ARM Cortex-M. Incluye un depurador en el IDE, compilador de C/C++, componentes esenciales de middleware e incluye todos los componentes necesarios para crear, construir y depurar aplicaciones [15].

Para la implementación del controlador $PI^\lambda D^\mu$ con el Arduino Uno se utilizan los programas Simulink-MATLAB R2016a, Arduino (IDE) 1.6.13 y el soporte para Arduino de Simulink/MATLAB (será necesario contar con una cuenta de MathWorks). El Arduino Uno es una placa con microcontrolador Atmel y circuitería de soporte con reguladores de tensión, puerto USB con adaptador USB-Serie para programar el microcontrolador desde una computadora personal y para comunicación con el propio chip. Posee 14 pines de entrada/salida para señales digitales de 0V a 5V. Dispone de 6 entradas analógicas, 6 salidas para generación de señales PWM de 8 bits, comunicaciones SPI e I^2C, memoria FLASH de 32KB, EEPROM de 1KB y frecuencia de reloj de 16MHz.4.

4. METODOLOGÍA

Las metodologías para la implementación del controlador PID fraccional en la tarjeta de desarrollo STM32-Discovery y en el Arduino Uno se describen a continuación.

a) Metodología con el STM32F4-Discovery

Instalación del software:

1. Instale MATLAB R2016a o superior.
2. Instale STM32CubeMX 4.18.0 (disponible en [14]), el driver del STM32F4-Discovery (disponible en [13]), stm32-MAT/TARGET (disponible en [14]). Entonces, desde la barra de tareas de MATLAB, en la opción Set Path, seleccione "Add with subfolders" y busque en el directorio de instalación la carpeta "STM32-MATH". Esto habilita el usos de la librería "Microelectronics" en SIMULINK.
3. Instale Keil μVision 5 (disponible en [15])
4. Una vez instalado Keil μVision 5 inícielo y proceda a descargar las librerías para la STM32F4-Discovery. Para ello, de clic en el icono "Pack Installer".

Implementación:

1. Genere un nuevo proyecto en STM32CubeMX, declarando los puertos a utilizar; habilite la frecuencia interna del oscilador de la tarjeta y el depurador en modo serial (Vea la Figura 5).
2. Construya en SIMULINK el modelo del sistema que desee implementar. Por ejemplo, en este trabajo se realiza el controlador $PI^\lambda D^\mu$ y, para fines didácticos, se incorpora también el modelo discretizado de la planta. Agregue los

bloques de ST Microelectronics que sirven para habilitar el MCU (Micro-controller Unit), para declarar los puertos de entrada/salida, y para habilitar el convertidor digital-analógico (DAC) que permitirá visualizar la respuesta en osciloscopio. Simule para verificar el funcionamiento (Vea la Figura 6).
3. Si la simulación es satisfactoria, genere el código en lenguaje C desde SIMULINK con el botón "Build Model". Posteriormente, proceda a abrir con Keil μVision 5 el archivo resultante con extensión .uvprojx.
4. Por defecto Keil reconoce el modelo de la tarjeta que se está usando, de lo contrario selecciónelo manualmente dando clic en el botón "Options for Target".
5. Desde "Options for Target" seleccione el depurador "ST-Link Debugger".// 6. En la opcion "Programming Algorithm" elija "STM32F4xx Flash".// 7. Proceda a compilar el programa. De no existir errores descargárguelo en la tarjeta con el botón "Download".
8. Finalmente, tome lectura de la salida del DAC para compararla con los resultados de simulación.

b) Metodología con el Arduino Uno

Instalación del software:
1. Instale MATLAB R2016a o superior.
2. Instale Arduino IDE 1.6.13 (disponible en [¿]).
3. Instale el complemento de Arduino para Simulink/MATLAB. Para ello, desde MATLAB de click en el botón "Add-Ons", seleccione la opción "Get Hardware Support Packages", seleccione "Internet (recommended)".
4. Instale el complemento de Arduino para Simulink/MATLAB. Para ello, desde MATLAB de click en el botón "Add-Ons", seleccione la opción "Get Hardware Support Packages", seleccione "Internet (recommended)".

Las metodologías propuestas se validan con dos ejemplos desarrollados paso a paso, uno para la tarjeta STM32-Discovery y el otro para el Arduino Uno. En el Ejemplo 1 se implementa un controlador $PI^{\lambda}D^{\mu}$ para el control de velocidad de un motor de corriente directa. En el Ejemplo 2 se incorpora un controlador $PI^{\lambda}D^{\mu}$ a un sistema de control de temperatura

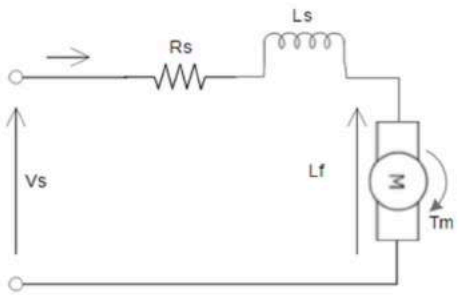

Fig. 2. Motor de corriente directa con excitación independiente.

5. EJEMPLO 1: CONTROL DE VELOCIDAD DE UN MOTOR DE CD.

El diagrama esquemático de un motor de corriente directa con excitación independiente se muestra en la Figura 2. La Tabla 1 presenta la lista de símbolos y abreviaturas correspondientes.

Cuadro 1.

Abrv.	Parametro
Rs	Resistencia de la armadura
Ls	Inductancia de la armadura
Vs	Voltaje aplicado
Lf	Corriente de campo
m	Motor
Tm	Torque del motor
Ti	Torque 1

La función de transferencia de este motor se obtuvo de manera experimental en [8], resultando

$$Gp(s) = \frac{1{,}23e^{0{,}0118s}}{0{,}32s+1} \quad \textbf{(4)}$$

que corresponde a un sistema de orden uno con un polo en s=32.25rad/s y tiempo muerto de 0.017 segundos. La respuesta al escalón en lazo cerrado de este sistema es lenta (tiempo de establecimiento de t_s=0.2s), no presenta oscilaciones y tiene error en estado estacionario e_{ss}=24 %. Mediante aproximación de Páde de (4) obtenemos

$$Gp(s) = \frac{-2{,}5s+1}{12{,}5s^2+7{,}5s+1} \quad \textbf{(5)}$$

Para disminuir t_s y reducir e_{ss} se incorpora un controlador $PI^{\lambda}D^{\mu}$ diseñado a partir de las siguientes restricciones de diseño: Ganancia unitaria en la frecuencia de cruce ω_c, margen de fase ϕ_{PM}= 60°, rechazo al ruido en alta frecuencia de al menos 20dB y rechazo a perturbaciones de al menos 20dB. La sintonización de los parámetros P, I, D, λ y μ para satisfacer estas restricciones se realizó en [16] con la función fmincon de MATLAB, obteniéndose

$$G_c(s) = -0{,}54202 + \frac{1{,}1277}{s^{0{,}6}} + 0{,}24801s^{0{,}2} \quad \textbf{(6)}$$

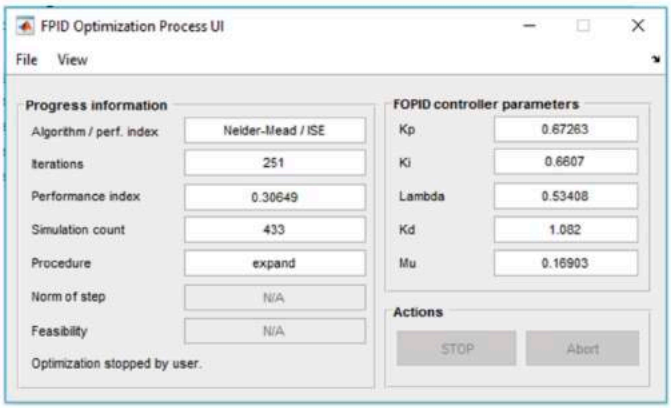

Fig. 3. Optimización del controlador $PI^{\lambda}D^{\mu}$.

Además, $G_C(s)$ se puede aproximar con MATLAB a una función de transferencia de orden entero a partir del método de aproximación Crone [17] mediante
kp=-0.54202;

```
ki=1.1277;
kd=0.24801;
lan=0.6, u=0.2;
cc=nipid(kp,kd,u,ki,lan,[2 200],2,'crone')
gc = c2d(cc,0.1)
```

obteniéndose

$$G_C(s) \approx \frac{0{,}00506s^3 + 0{,}1087s^2 - 12{,}06s + 113{,}6}{0{,}0637s^3 + 5{,}223s^2 + 101{,}5s} \quad (7)$$

Posteriormente, se calcula la transformada Z de (7) con la función "*c2d*" de MATLAB, aproximación bilineal y un periodo de muestreo de 0.1 segundos, resultando

$$G_C(Z) = \frac{0{,}59932z^3 - 0{,}352z^2 + 0{,}00966z + 0{,}0001588}{z^3 - 1{,}049z^2 + 0{,}04887z - 0{,}0000964} \quad (8)$$

De forma similar se calcula la transformada Z de la planta (5) resultando

$$Gp(Z) = \frac{0{,}1732z + 0{,}1488}{z^2 - 0{,}738z} \quad (9)$$

El sistema (9) sin compensación y con el compensador (8) en lazo cerrado se muestra en la Figura 4(a). De acuerdo con la Figura 4(b), con el sistema no compensado el error en estado estacionario para una entrada escalón unitario es de 45 %, con un tiempo de elevación de 0.08s, mientras que el sistema compensado presenta un tiempo de elevación de 0.08s, sin error apreciable en estado estacionario y con sobretiro de 12.2 %.

Siguiendo la metodología de la Sección 4 y luego de instalar el software requerido en un equipo con sistema operativo Windows 10 y procesador AMD A8, se genera un nuevo proyecto con la selección de puertos indicada en la Figura 5 (Paso 1). Como se aprecia los puertos RCC_OSC_IN y RCC_OSC_OUT corresponden a la señal interna de reloj y se habilitan con la opción "Crystal/Ceramic Resonator". Los puertos DAC_OUT1 (entrada escalón) y DAC_OUT2 (salida del sistema) se habilitan con la opción "DAC", mientras que los puertos SYS_JTMS-SWDIO y SYS_JTCK-SWMCU quedan habilitados con la opción "SYS" (puertos del debugger). Posteriormente (Paso 2), se agregan en SIMULINK al sistema de la Figura 4(a) los bloques de ST Microelectronics desde el "Library Browser". Entonces, se realizan las conexiones que se aprecian en la Figura 6, se simula y se siguen los paso 3 a 8 de la metodología.

La caracterización del sistema se obtuvo con el osciloscopio HDO6000 de Teledyne-Lecroy. La señal escalón de entrada de 1V de amplitud se realizó desde la tarjeta STM32-Discovery. Con las opciones de medición del osciloscopio se obtuvo un sobretiro del 12 % y un tiempo de elevación de 0.09s.

6. CONCLUSIONES

Se presentaron dos metodologías para la implementación directa desde SIMULINK de controladores PID de orden fraccional en tarjetas de desarrollo STM32-Discovery y Arduino uno. Se observó que los resultados obtenidos corresponden a las respuestas obtenidas por simulación. Con estas metodologías se espera que este tipo de controladores adquieran mayor aceptación en la industria. Estas metodologías son aplicables también al desarrollo de otros tipos de sistemas, tales como sistemas de adquisición y procesamiento de señales o sistemas caóticos, por mencionar algunos. El uso de hardware reconfigurable permite el desarrollo en poco tiempo de prototipos y el ajuste en sitio de los parámetros de control o del controlador mismo.

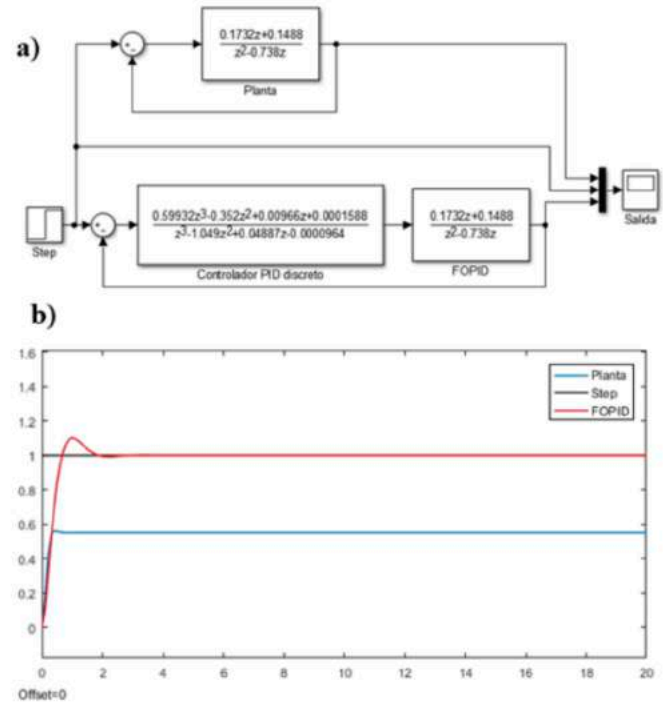

Fig. 4. Simulación del motor de CD. (a) Diagrama a bloques de la planta sin control y con control, (b) respuesta al escalón de la planta con y sin control.

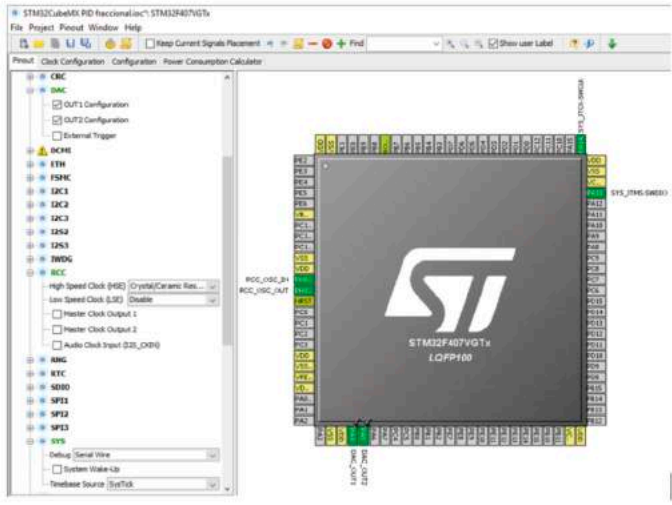

Fig. 5. Inicializador de software embebido.

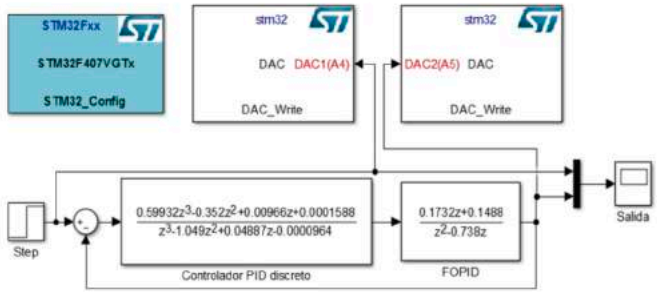

Fig. 6. Sistema de la Fig. 4(b) con bloques de la librería ST Microelectronics en SIMULINK.

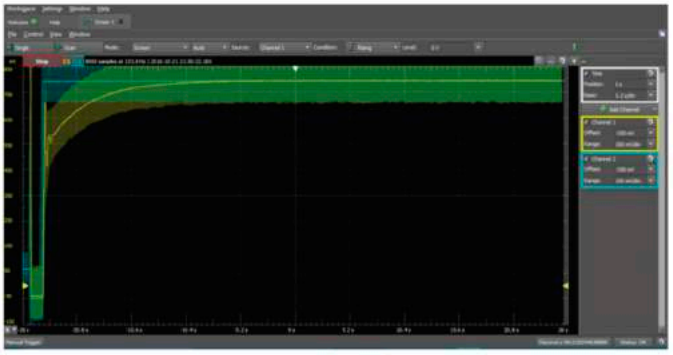

Fig. 7. Respuesta en un osciloscopio.

7. REFERENCES

REFERENCIAS

1. C. Muñiz-Montero, L. V. García-Jiménez, L. A. Sánchez-Gaspariano, C. Sánchez-López, V. R. González-Díaz, and E. Tlelo-Cuautle, Nonlinear Dynamics pp. 1–16 (2017).
2. I. Polubny, IEEE Trans. Automatic Control **44**, 208 (1999).
3. K. J. Åström and T. Hägglund, Control Engineering Practice **9**, 1163 (2001).
4. T. Aleksei, P. Eduard, and B. Juri, "A flexible matlab tool for optimal fractional-order pid controller design subject to specifications," in "Control Conference (CCC), 2012 31st Chinese," (IEEE, 2012), pp. 4698–4703.
5. C. Monje, B. Vinagre, Y. Chen, V. Feliu, P. Lanusse, and J. Sabatier, "Proposals for fractional piλdμ tuning," in "Proceedings of The First IFAC Symposium on Fractional Differentiation and its Applications (FDA04)," (2004).
6. FOMCON, "Último acceso octubre 2016," urlhttp://fomcon.net/fomcon-toolbox/overview/ (2016).
7. MathWorks, "Último acceso noviembre 2016," urlhttps://www.mathworks.com/help/signal/ref/invfreqs.html (2016).
8. A. Sandhya and R. S. M. Prameela, International Journal of Innovations in Engineering and Technology **4**, 534 (2016).
9. B. M. Vinagre, C. A. Monje, A. J. Calderón, and J. I. Suárez, Journal of Vibration and Control **13**, 1419 (2007).
10. D. Visan, I. Lita, and I. Cioc, "Temperature control system based on adaptive pid algorithm implemented in fpaa. electronics technology (isse)," in "2011 34th International Spring Seminar on, vol., no," , vol. 501 (2011), vol. 501, pp. 11–15.
11. A. Amirahmadi, M. Rafiei, K. Tehrani, G. Griva, and I. Batarseh, Journal of Power Electronics **15**, 160 (2015).
12. A. J. Calderón, B. M. Vinagre, and V. Feliu, Signal Processing **86**, 2803 (2006).
13. STMicroelectronics, "Último acceso octubre 2016," urlhttp://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-discovery-its/stm32f4discovery.html (2016).
14. STMicroelectronics, "Último acceso octubre 2016," urlhttp://www.st.com/content/st_com/en/products/deve-lopment-tools/software-development-tools/stm32-software devel opment-tools/stm32-configurators-and-code-generators/stm32cu bemx.html (2016).
15. ARMKEIL, "Último acceso octubre 2016," urlhttp://keil.com/ (2016).
16. C. A. Monje, A. J. Calderon, B. M. Vinagre, Y. Chen, and V. Feliu, Nonlinear Dynamics **38**, 369 (2014).
17. L. E. G. Jaimes, C. A. V. Ospina, and H. W. L. Espinosa, REVISTA POLITÉCNICA **10**, 65 (2015).