Reporte de práctica 2 - Generador de funciones

1288433 - Pedro I. López [dreilopz@gmail.com]

4 de mayo de 2012

Maestro: Dr. Luis M. Torres Treviño
Materia: 483 PERCEPCIÓN (Laboratorio)
Dependencia: FIME UANL

1. Introducción

El ejercicio práctico consiste en implementar un generador de funciones basándose en la implementación de la calculadora analógica de la práctica 1. Al igual que con la calculadora analógica, las entradas del sistema son 2 señales analógicas con un rango de de 0-5 V (potenciómetros rotacionales) y 2 señales digitales (DIP switch). La salida del sistema se mide directamente en el pin de salida analógica del `USB6008` (`USB6008.AO0`).

2. Descripción

El generador de funciones se implementará con un `USB6008`, una PC y otros componentes eléctricos montados en un protoboard, con un programa desarrollado en lenguaje MATLAB. Las funciones que se pueden generar son *triangular*, *trapezoidal*, *sigmoidal* y *gaussiana*. Tales funciones fueron implementadas en MATLAB (ver Listing 1).

Los 2 potenciómetros rotacionales servirán para proporcionar 2 parámetros a las funciones triangular, trapezoidal y guaussiana. En el caso de las funciones triangular y trapezoidal requieren más de 2 parámetros, tales parámetros serán constantes (ver código en Listings 4). La función sigmoidal no necesita ningún parámetros adicional.

Los 2 switches sirven para seleccionar la función que se desea general en `USB6008.AO0`.

La NI USB-6008 (`USB6008`) es una interfaz de adquisición y salida de datos para PC con conexión USB. Los voltajes de los potenciómetros se envían a los canales analógicos de entrada `USB6008.AI0` y `USB6008.AI1`, y la salida se detecta en `USB6008.AO0`. Los estados de los switches que seleccionan la función generada se leen con `USB6008.P0.0` y `USB6008.P0.1`.

El software se desarrolló en MATLAB, utilizando el *Data Acquisition Toolbox* para obtener acceso a la API de la `USB6008`: *NIDAQ-mx*. La interfaz del programa es una línea de comandos interactiva.

Componentes utilizados:

- 2 potenciómetros de 10KΩ
- 1 DIP switch
- 1 NI USB6008
- Alambre
- Conectores
- 1 osciloscopio
- 1 fuente de poder de 5 V.

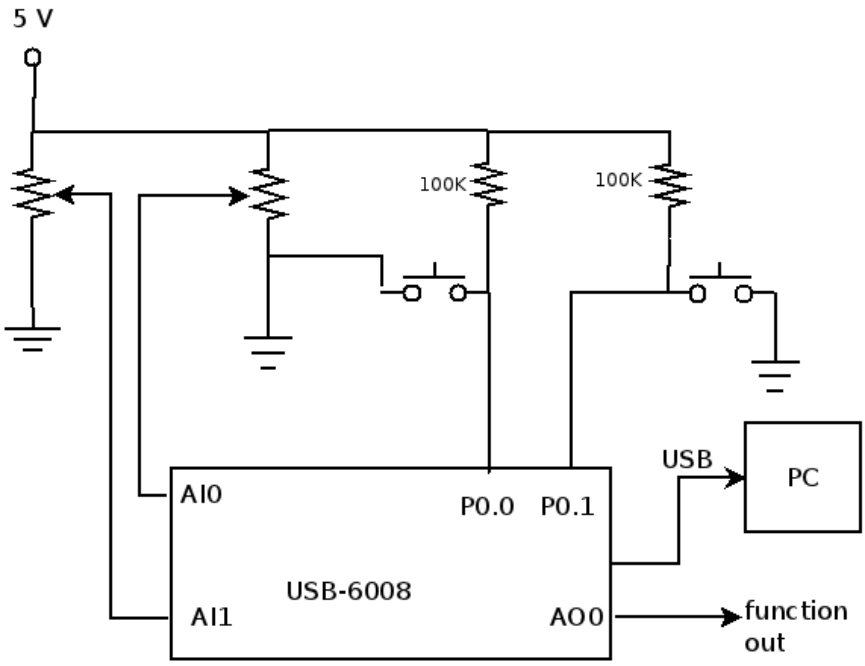

Figura 1: Diagrama de sistema.

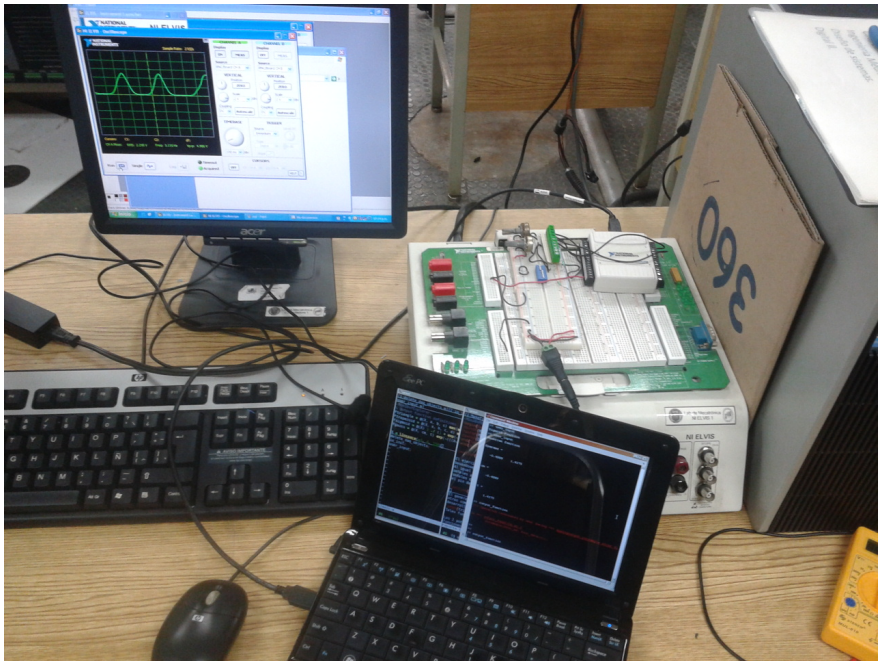

Figura 2: Foto del sistema global.

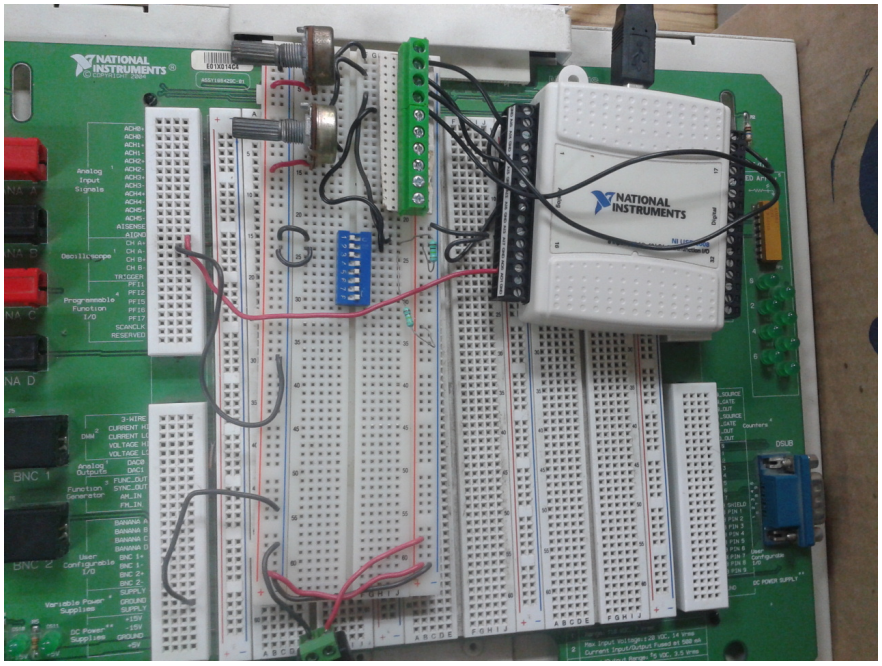

Figura 3: Circuito conectado al NI ELVIS para utilizar el osciloscopio virtual.

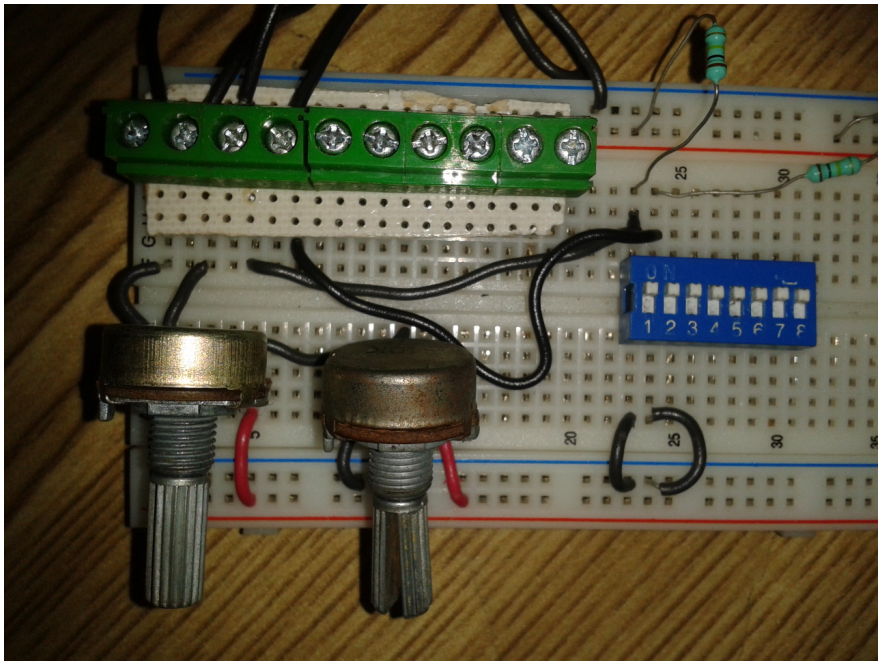

Figura 4: Entradas de sistema.

3. Resultados

Las funciones se generan correctamente. El sistema está implementado de manera que la actualización de los valores de los potenciómetros y de la función seleccionada no se hace de manera continua, sino que se tiene que detener la generación de la función. En otras palabras, los pasos para utilizar este sistema son:

1. Ejecutar MATLAB y cambiar el directorio actual al de la práctica.
2. Ejecutar `p2main`.
3. Seleccionar la función con el DIP switch y modificar las señales de entrada de los potenciómetros.
4. Ejecutar `user_input` y observar la línea de comandos para ver la función seleccionada y los parámetros configurados.
5. Repetir los 2 pasos anteriores hasta obtener la configuración y función a generar deseada.
6. Ejecutar `output_function`, la función se esta generando y se puede visualizar con el osciloscopio.
7. Para detener la generación presionar CTRL+C.

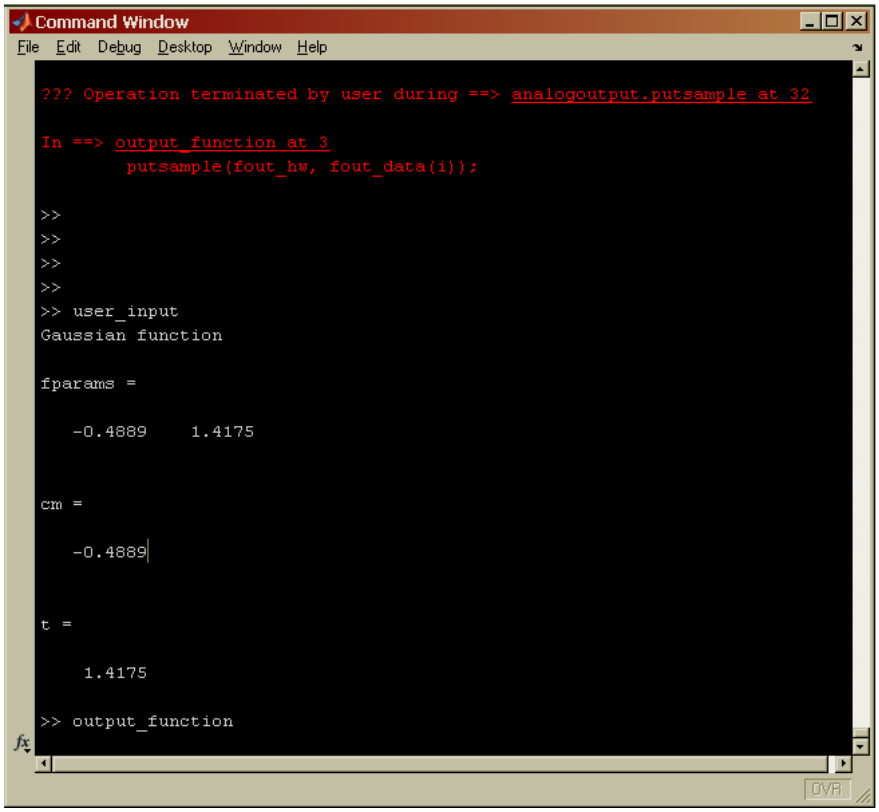

Figura 5: Programa del generador de funciones generando una función gaussiana.

4. Pruebas

Las pruebas se realizaron en el salón de laboratorio con la presencia y aprobación del instructor. Se muestran imágenes de las señales en el osciloscopio, las imágenes son fotografías ya que fue imposible copiar las capturas de pantalla de la estación de trabajo por fallas técnicas (los puertos USB no funcionaban y no se tenían privilegios de administrador para obtener las imágenes por medio del adaptador de red).

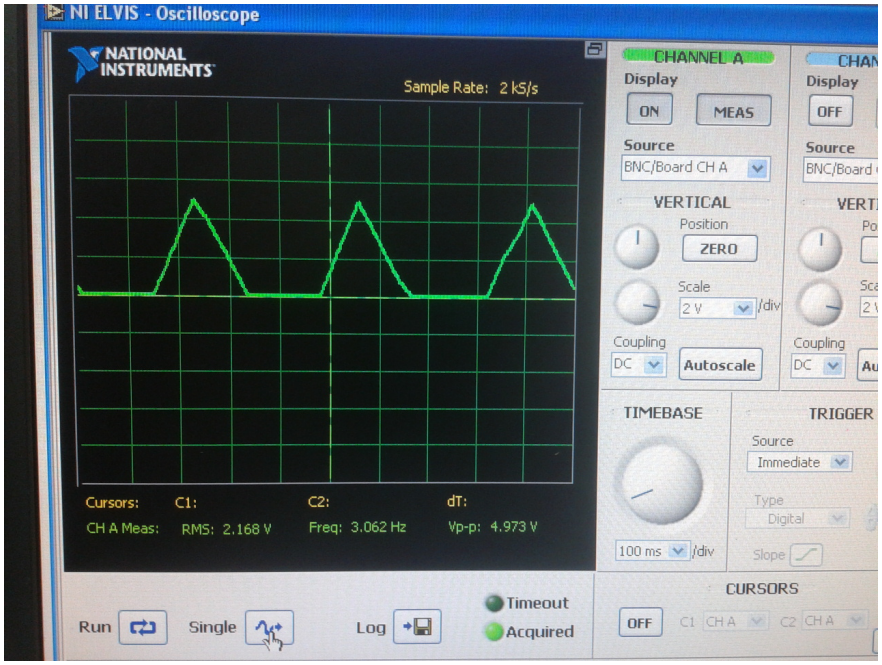

Figura 6: Generando función triangular con parámetros $a = -0{,}5215$, $b = 1{,}4827$ y $c = 4$

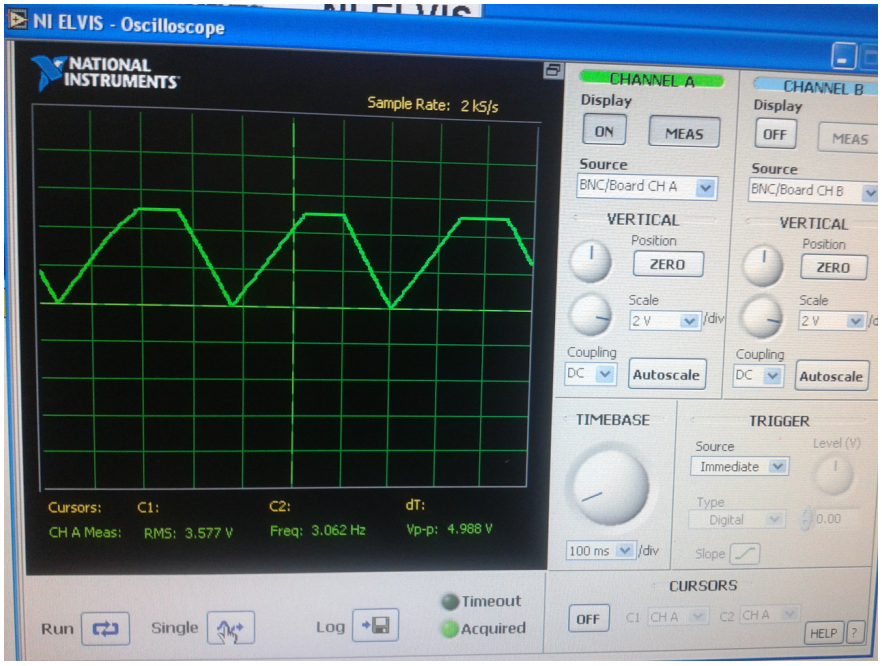

Figura 7: Generando función trapezoidal con parámetros $a = -4$, $b = -0{,}4564$, $c = 1{,}4664$ y $d = 4$

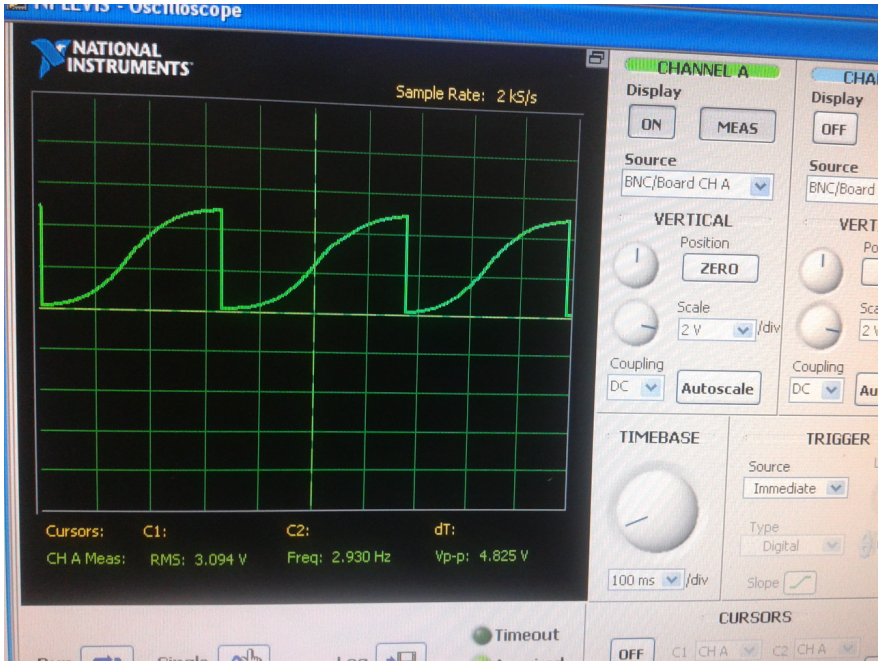

Figura 8: Generando función sigmoidal.

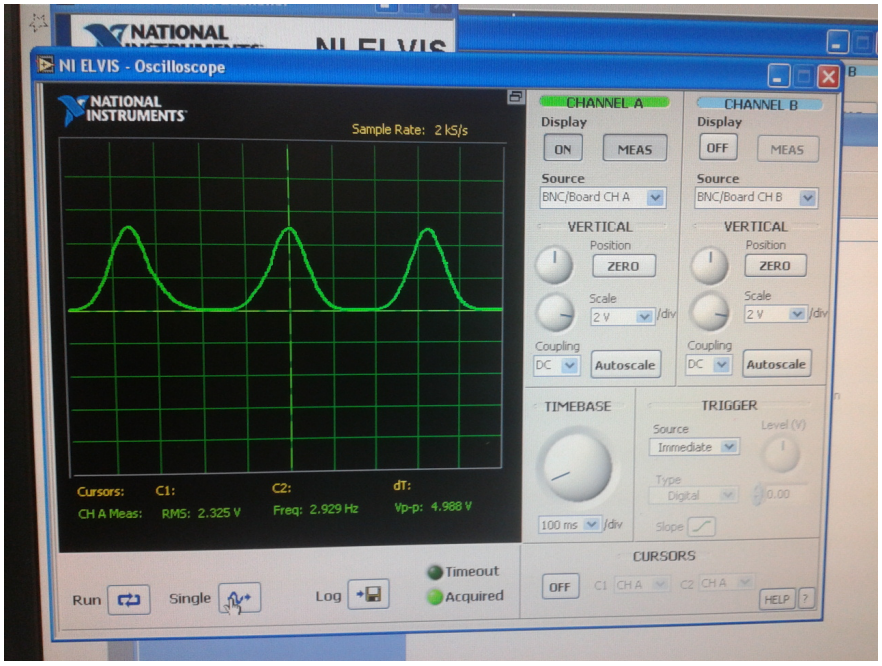

Figura 9: Generando función gaussiana con parámetros $cm = -0{,}4889$ y $t = 1{,}4175$.

5. Código

Listing 1: `p2main`: Script principal de la calculadora.

```
% Output function definitions.
ftriangle = @(X, a, b, c) max(min((X-a)/(b-a), (c-X)/(c-b)), 0);
ftrapezoid = @(X, a, b, c, d) max(min(min((X-a)/(b-a),1),(d-X)/(d-c)),0);
fsigmoid = @(X) 1 ./ (1 + exp(-X));
fgauss = @(X, cm, t) exp(-(((X-cm)./t).^2));

X = linspace(-4, 4, 150);
delete_daq_objects;
io_init;
user_input;
```

Listing 2: `delete_daq_objects`: Script que elimina objetos de dispositivos de adquisición de datos

```
try
    delete(fparams_hw);
    delete(fout_hw);
    delete(fselect_hw);
    clear('fparams_hw', 'fout_hw', 'fselect_hw');
end
```

Listing 3: `io_init`: Inicializando el hardware.

```
DAQ_DEVICE_NAME = 'usb6008';
ADAPTOR = daqhwinfo('nidaq');

delete_daq_objects;

fparams_hw = analoginput(ADAPTOR.AdaptorName, DAQ_DEVICE_NAME);
fparams_hw.SampleRate = 1000;
fparams_hw.InputType = 'SingleEnded';
fparams_hw.TriggerType = 'Immediate';
fparams_hw.SamplesPerTrigger = 2;
addchannel(fparams_hw, 0:1);

fout_hw = analogoutput(ADAPTOR.AdaptorName, DAQ_DEVICE_NAME);
addchannel(fout_hw, 0);
fout_hw.SampleRate = 150;
fout_hw.TriggerType = 'Immediate';

fselect_hw = digitalio(ADAPTOR.AdaptorName, DAQ_DEVICE_NAME);
addline(fselect_hw, 0:1, 'in');
```

Referencias

[1] *MATLAB R2010a documentation*. The MathWorks, Inc.

[2] *NI USB-6008/6009 - User guide and specifications*. 2004-2008 National Instruments Corporation.

Listing 4: `user_input`: Inicializando el hardware.

```
start(fparams_hw);
fparams = getdata(fparams_hw);
fparams = fparams(1,:);

% Scale analog input.
for i = 1:size(fparams, 2)
    if fparams(i) < 0
        fparams(i) = 0.0;
    elseif fparams(i) > 5.0
        fparams(i) = 5.0;
    end
end

% Select function.
fselect = bin2dec(num2str(logical(getvalue(fselect_hw))));
switch fselect
    case 0
        disp('Triangle function');
        fparams = (fparams * (8/5)) - 4
        a = fparams(1)
        b = fparams(2)
        c = 4
        fout_data = 5 * ftriangle(X, a, b, c);
    case 1
        disp('Trapezoid function');
        fparams = (fparams * (8/5)) - 4
        a = -4
        b = fparams(1)
        c = fparams(2)
        d = 4
        fout_data = 5 * ftrapezoid(X, a, b, c, d);
    case 2
        disp('Sigmoid function');
        fout_data = 5 * fsigmoid(X);
    case 3
        disp('Gaussian function');
        fparams = (fparams * (8/5)) - 4
        cm = fparams(1)
        t = fparams(2)
        fout_data = 5 * fgauss(X, cm, t);
end
```

Listing 5: `output_function`: Inicializando el hardware.

```
while true
    for i = 1:size(fout_data, 2)
        putsample(fout_hw, fout_data(i));
    end
end
```