Realización electrónica de sistemas caóticos: Parte 3, en sistemas digitales

Francisco Antonio Rodríguez Cruz, César de Jesús Chacón Rendón, Angel Rodriguez-Liñan
Universidad Autónoma de Nuevo León, Facultad de Ingeniería Mecánica y Eléctrica,
Ave. Universidad S/N, Cd. Universitaria, San Nicolás de los Garza, N.L., México.
franciscoantoniordz@hotmail.com, ces.chre98@gmail.com, angel.rodriguezln@uanl.edu.mx

RESUMEN

En las primeras partes de este trabajo, se mostró la realización electrónica con circuitos analógicos de algunos sistemas caóticos continuos cuadráticos y lineales por tramos, mediante circuitos con amplificadores operacionales y otros componentes, así como la equivalencia de sus variables electrónicas con los modelos matemáticos establecidos. En esta tercera parte, se aplica discretización de sistemas dinámicos para la implementación de estos sistemas caóticos en la plataforma de código abierto Arduino, ofreciendo así simplicidad y versatilidad para aplicaciones digitales. Finalmente, se ilustran resultados de su comportamiento caótico y equivalencia numérica con los modelos matemáticos continuos.

PALABRAS CLAVE

Sistemas caóticos, discretización de Euler, arduino.

ABSTRACT

In the first parts of this work, the electronic realization with analog circuits of some continuous quadratic and piecewise linear chaotic systems was shown, using circuits with operational amplifiers and other components, as well as the equivalence of their electronic variables with the established mathematical models. In this third part, discretization of dynamic systems is applied for the implementation of these chaotic systems in the Arduino open-source platform, thus offering simplicity and versatility for digital applications. Finally, results of its chaotic behavior and numerical equivalence with continuous mathematical models are illustrated.

KEYWORDS

Chaotic systems, Euler's discretization, arduino.

INTRODUCCIÓN

En las primeras partes de este trabajo, se abordó la realización electrónica de sistemas no lineales con términos cuadráticos y lineales por tramos que exhiben comportamiento caótico, mediante circuitos analógicos con amplificadores operacionales y otros componentes, así como la equivalencia de sus variables electrónicas con los modelos matemáticos establecidos. Sin embargo, también es posible la implementación de sistemas caóticos en dispositivos digitales, como microcontroladores, procesadores de señales digitales (DSP, por sus siglas en inglés), FPGA (del inglés *field-programmable gate array*) u otros[1], sus implementaciones y aplicaciones han ido aumentando debido al auge actual de las redes digitales y los sistemas embebidos, como en telecomunicaciones[2-5].

En esta tercera parte, se aplica discretización de sistemas dinámicos para su implementación digital en la plataforma de código abierto Arduino, ofreciendo así simplicidad y versatilidad para aplicaciones digitales. Particularmente, se utiliza el método de discretización de Euler en la siguiente sección para ilustrar la discretización en el tiempo de los sistemas dinámicos, por ser uno de los métodos más simples mediante los cuales se obtiene su modelo en ecuaciones en diferencias. En la sección posterior, se muestran sencillos códigos de programación en Arduino con los cuales se puede implementar los sistemas caóticos, cuya salida son digitales. Luego, se presenta la adquisición de datos con la extensión PLX-DAQ, que permite exportar los datos de Arduino al paquete Microsoft Excel, generando archivos de datos fáciles de manipular y procesar con muchas aplicaciones y dispositivos digitales. Se ilustran resultados de su comportamiento caótico y equivalencia numérica con los modelos matemáticos continuos. Finalmente, se presentan algunas conclusiones.

DISCRETIZACIÓN DE SISTEMAS CAÓTICOS

Además de la realización con electrónica analógica, como se ha mostrado en las primeras partes de este trabajo, los sistemas caóticos también pueden embeberse en dispositivos digitales. Para ello, es necesaria la discretización en el tiempo de las ecuaciones dinámicas. Uno de los métodos de discretización más sencillos y conocidos es el método de Euler[1,4], el cual brinda la posibilidad a los dispositivos digitales de realizar las operaciones con menor complejidad. Aunque hay métodos más estables y precisos, resulta conveniente utilizarlo cuando no se necesita mucha exactitud de los valores calculados y se quiere reducir el esfuerzo computacional[4].

El método de discretización de Euler se explica de forma resumida a continuación:

Sea el modelo dinámico de un sistema caótico escrito como la función $\frac{dx(t)}{dt} = f\big(x(t)\big)$, donde x es el vector de variables de estado. Puesto que por definición $\frac{dx(t)}{dt} = \lim_{\Delta t \to 0} \frac{x(t+\Delta t)-x(t)}{\Delta t}$, si $t = k\Delta t$ con k=0,1,2,…N y Δt un pequeño incremento constante de tiempo, es fácil deducir que $x(k+1) \approx x(k) + \Delta t \cdot f\big(x(k)\big)$.

De esta manera, es posible calcular el valor en tiempo discreto de $x(k+1)$ a partir de su valor actual $x(k)$ y del modelo $f\big(x(k)\big)$. Iniciando en k=0 y repitiendo iterativamente este proceso hasta la cantidad N de muestras de tiempo que se deseen, se reproduce la trayectoria de los estados $x(k)$ en tiempo discreto del sistema dinámico.

Entonces, la forma discreta del sistema Lorenz adimensional[6] resulta en

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[\sigma(x_2(k) - x_1(k))] \\ x_2(k+1) = x_2(k) + \Delta t[\rho x_1(k) - x_2(k) - x_1(k)x_3(k)] \\ x_3(k+1) = x_3(k) + \Delta t[x_1(k)x_2(k) - \beta x_3(k)] \end{cases} \quad (1)$$

El sistema Lü adimensional[7] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[a(x_2(k) - x_1(k))] \\ x_2(k+1) = x_2(k) + \Delta t[-x_1(k)x_3(k) + cx_2(k)] \\ x_3(k+1) = x_3(k) + \Delta t[x_1(k)x_2(k) - bx_3(k)] \end{cases} \quad (2)$$

El sistema Chen adimensional[8] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[a(x_2(k) - x_1(k))] \\ x_2(k+1) = x_2(k) + \Delta t[(c-a)x_1(k) - x_1(k)x_3(k) + cx_2(k)] \\ x_3(k+1) = x_3(k) + \Delta t[x_1(k)x_2(k) - bx_3(k)] \end{cases} \quad (3)$$

El sistema Rössler adimensional[9] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[-x_2(k) - x_3(k)] \\ x_2(k+1) = x_2(k) + \Delta t[x_1(k) + ax_2(k)] \\ x_3(k+1) = x_3(k) + \Delta t\big[b + x_3(k)[x_1(k) - c]\big] \end{cases} \quad (4)$$

El sistema Liu-Chen adimensional[10] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[ax_1(k) - x_2(k)x_3(k)] \\ x_2(k+1) = x_2(k) + \Delta t[-bx_2(2) + x_1(k)x_3(k)] \\ x_3(k+1) = x_3(k) + \Delta t[-cx_3(k) + x_1(k)x_2(k)] \end{cases} \quad (5)$$

El sistema de Malasoma adimensional[11] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[x_2(k)] \\ x_2(k+1) = x_2(k) + \Delta t[x_3(k)] \\ x_3(k+1) = x_3(k) + \Delta t[-\alpha x_3(k) - x_1(k) + x_1(k)x_2(k)] \end{cases} \quad (6)$$

El sistema MACM adimensional[3] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[-ax_1(k) - bx_2(k)x_3(k)] \\ x_2(k+1) = x_2(k) + \Delta t[-x_1(k) + cx_2(k)] \\ x_3(k+1) = x_3(k) + \Delta t[d - {x_2}^2(k) - x_3(k)] \end{cases} \quad (7)$$

El sistema afín a Rössler adimensional[12] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[-\alpha x_1(k) - \beta x_2(k) - \lambda x_3(k)] \\ x_2(k+1) = x_2(k) + \Delta t[x_1(k) + \gamma x_2(k)] \\ x_3(k+1) = x_3(k) + \Delta t[-\eta(x_1(k)) - x_3(k)] \end{cases} \quad (8)$$

con $\eta(x_1(k)) = \begin{cases} 0, & x_1(k) \le 3 \\ -\mu(x_1(k) - 3), & x_1(k) > 3 \end{cases}$

El sistema de Chua adimensional[13] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t \cdot \alpha[x_2(k) - x_1(k) - \eta(k)] \\ x_2(k+1) = x_2(k) + \Delta t[x_1(k) - x_2(k) + x_3(k)] \\ x_3(k+1) = x_3(k) + \Delta t[-\beta x_2(k) - \gamma x_3(k)] \end{cases} \quad (9)$$

con $\eta(k) = \begin{cases} bx_1(k) - a + b, & x(k) < -1 \\ ax_1(k), & -1 \le x(k) \le 1 \\ bx_1(k) + a - b, & x(k) > 1 \end{cases}$

El circuito de Sprott adimensional[14] discretizado resulta:

$$\begin{cases} x_1(k+1) = x_1(k) + \Delta t[x_2(k)] \\ x_2(k+1) = x_2(k) + \Delta t[x_3(k)] \\ x_3(k+1) = x_3(k) + \Delta t[ax_3(k) - x_2(k) + \eta(x_1(k))] \end{cases} \quad (10)$$

con sus diferentes funciones como **a)** $\eta(x_1(k)) = b|x_1(k)| - c$, **b)** $\eta(x_1(k)) = -b\max(x_1(k), 0) + c$, **c)** $\eta(x_1(k)) = bx_1(k) - c\text{signo}(x_1(k))$ y **d)** $\eta(x_1(k)) = -bx_1(k) + c\text{signo}(x_1(k))$.

En esta parte del trabajo, se propone la implementación de estos sistemas caóticos discretizados en el dispositivo Arduino, ya que es una plataforma muy barata y simple de programar, para posibles aplicaciones digitales que se quieran realizar con estos sistemas caóticos.

DISEÑO DE EXPERIMENTOS

Con el fin de verificar que el comportamiento de los modelos discretizados sea congruente con el comportamiento típico de los sistemas caóticos de Lorenz, Lü, Chen, Rössler, Liu-Chen, Malasoma, MACM, Rössler afín, Chua y Sprott, sus modelos discretizados son embebidos en Arduino y su respuesta se compara con la de los modelos adimensionales en simulaciones numéricas en el software Scilab.

Fig. 1. Dispositivo digital Arduino Mega 2560.

Un código de programación con el que se puede implementar el sistema de Lorenz (1) en Arduino Mega 2560 (dispositivo mostrado en la figura 1) es el siguiente:

```
/* Código para generar señales de atractores caóticos */
//Parámetros del sistema caótico:
      float sigma=10;
      float rho=28;
      float beta=8/3;
      float Dt=0.002;//paso de integración
//Condiciones iniciales:
      float x1=0.001;
      float x2=0.001;
      float x3=0.001;
      float X1,X2,X3;
void setup()    // Configura el puerto serial:
   {
   Serial.begin(57600);
   Serial.println("CLEARDATA"); //Borra datos anteriores
   Serial.println("LABEL,Hora,Tiempo(segundos),x1,x2,x3"); //Escribe encabezados de las señales que se desea adquirir
   Serial.println("RESETTIMER"); //Restablece el temporizador a 0
   }
void loop()//programa principal iterativo
   {
   //Imprime valor de señales en muestra k:
   Serial.print("DATA,TIME,TIMER,");
   Serial.print(x1);
   Serial.print(",");
   Serial.print(x2);
   Serial.print(",");
   Serial.println(x3);
   //Modelo del sistema caótico:
   X1=x1+Dt*(sigma*(x2-x1));
   X2=x2+Dt*(rho*x1-x2-x1*x3);
   X3=x3+Dt*(x1*x2-beta*x3);
   //Sobreescribe variables para iniciar siguiente iteración:
   x1=X1;
   x2=X2;
   x3=X3;}
```

Para implementar los demás sistemas caóticos, puede utilizarse el mismo código de programación, simplemente reemplazando las líneas de programación correspondientes a sus parámetros y ecuaciones de modelo:

Para el sistema Lü (2), reemplazar las líneas:

```
//Parámetros del sistema caótico
      int a=36;
      int b=3;
      int c=20;

//Modelo del sistema caótico
     X1=x1+Dt*(a*(x2-x1));
     X2=x2+Dt*(-x1*x3+c*x2);
     X3=x3+Dt*(x1*x2-b*x3);
```

Para el sistema Chen (3), reemplazar por las líneas:

```
//Parámetros del sistema caótico
      int a=35;
      int b=3;
      int c=28;

//Modelo del sistema caótico
     X1=x1+Dt*(a*(x2-x1));
     X2=x2+Dt*((c-a)*x1-x1*x3+c*x2);
     X3=x3+Dt*(x1*x2-b*x3);
```

Para el sistema Rössler (4), reemplazar por las siguientes líneas:

```
//Parámetros del sistema caótico:
      float a=0.2;
```

```
        float b=0.2;
        float c=5.7;
        float Dt=0.005;//paso de integración

//Condiciones iniciales:
        float x1=0.01;
        float x2=0.01;
        float x3=0.01;
        float X1,X2,X3;

//Modelo del sistema caótico:
    X1=x1+Dt*(-x2-x3);
    X2=x2+Dt*(x1+a*x2);
    X3=x3+Dt*(b+x3*(x1-c));
```

Para el sistema Liu-Chen (5), sustituir las siguientes líneas:

```
//Parámetros del sistema caótico:
        int a=5;
        int b=10;
        float c=3.4;
        float Dt=0.0008;//paso de integración
//Condiciones iniciales:
        float x1=0.1;
        float x2=0.1;
        float x3=0.1;

//Modelo del sistema caótico:
    X1=x1+Dt*(a*x1-x2*x3);
    X2=x2+Dt*(-b*x2+x1*x3);
    X3=x3+Dt*(-c*x3+x1*x2);
```

Para el sistema caótico de Malasoma (6), reemplazar las siguientes líneas:

```
//Parámetros del sistema caótico
        float alpha=2.04;
        float Dt=0.005;//Paso de integración
//Condiciones iniciales
        float x1=1;
        float x2=-1;
        float x3=1;

//Modelo del sistema caótico
      X1=x1+Dt*(x2);
      X2=x2+Dt*(x3);
      X3=x3+Dt*(-alpha*x3-x1+x1*x2);
```

Para el sistema caótico MACM (7), reemplazar las líneas:

```
//Parámetros del sistema caótico:
        float a=1.786;
        float b=2.074;
        float c=0.5;
        float d=1.2;
        float Dt=0.02;//paso de integración

//Modelo del sistema caótico:
    X1=x1+Dt*(-a*x1-b*x2*x3);
    X2=x2+Dt*(-x1+c*x2);
    X3=x3+Dt*(d-x2*x2-x3);
```

Para el sistema afín a Rössler (8) reemplazar por las líneas:

```
//Parámetros del sistema caótico:
        float alpha=0.05;
        float beta=0.5;
        float lambda=1;
        float gamma=0.23;
        float mu=15;
        float Dt=0.1;//paso de integración
        float eta;
```

```
//Modelo del sistema caótico:
        if(x1>3){
                eta=mu*(x1-3);
        }
        else{
                eta=0;
        }
        X1=x1+Dt*(-alpha*x1-beta*x2-lambda*x3);
        X2=x2+Dt*(x1+gamma*x2);
        X3=x3+Dt*(eta-x3);
```

Para el sistema caótico de Chua (9), reemplazar las líneas:

```
//Parámetros del sistema caótico:
    int alpha=10;
    int beta=18;
    float gamma=0.14;
    float a=-1.3636;
    float b=-0.7364;
    float Dt=0.02;//paso de integración

//Modelo del sistema caótico:
        if(x1<-1){
                eta=b*x1-a+b;
        }
        else{
                if(x1>1){
                        eta=b*x1+a-b;
                }
                else{
                        eta=a*x1;
                }
        }
        X1=x1+Dt*alpha*(x2-x1-eta);
        X2=x2+Dt*(x1-x2+x3);
        X3=x3+Dt*(-beta*x2-gamma*x3);
```

Para el sistema Sprott (10), reemplazar las siguientes líneas, agregando al final del código las líneas de la función signo:

```
//Parámetros del sistema caótico
    float a=0.6;
    float Dt=0.02;//Paso de integración

//Modelo del sistema caótico (elegir la función eta que corresponda)
   eta=abs(x1)-0.9;                    // -----función a)----
   //eta=-7.4*max(x1,0)+0.5;           // -----función b)----
   //eta=1.11*x1-4.5*sign(x1);         // -----función c)----
   //eta=-1.2*x1+2*sign(x1);           // -----función d)----
   X1=x1+Dt*x2;
   X2=x2+Dt*x3;
   X3=x3+Dt*(-a*x3-x2+eta);

//Función signo para las funciones c) y d)
  int sign(float val)
  {
  if(val>0) return 1;
  if(val==0) return 0;
  if(val<0) return -1;}
```

ADQUISICIÓN DE DATOS DE ARDUINO

Para la adquisición de datos desde Arduino a Excel en tiempo real, se utilizó la extensión PLX-DAQ. Primero, se descarga y descomprime dicha extensión, obtenida de https://www.parallax.com/downloads/plx-daq

Fig. 2. Interfaz PLX-DAQ.

Luego, se compila en Arduino el código de programación de la sección anterior correspondiente. Se ejecuta el archivo descomprimido de la extensión 'PLX-DAQ.xlsm', cuya interfaz se muestra en la figura 2. En ella se selecciona el puerto donde está conectado el Arduino y la velocidad en baudios. Se ejecuta el programa de Arduino y entonces se presiona el botón *Connect*, el cual inicia la adquisición de datos en Excel, como se ilustra en la figura 3.

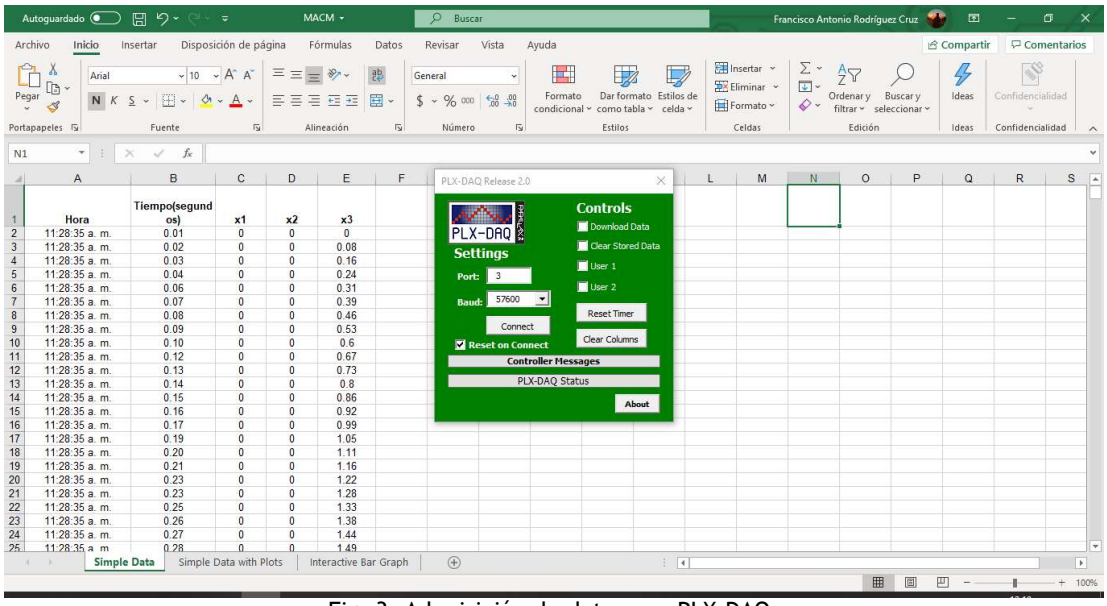


Fig. 3. Adquisición de datos con PLX-DAQ.

RESULTADOS DE SIMULACIONES Y DE IMPLEMENTACIÓN

A continuación, se muestran y comparan los resultados obtenidos de las implementaciones de modelos discretizados y simulaciones de los modelos continuos adimensionales. Los datos numéricos obtenidos se muestran gráficamente en forma de proyecciones de los atractores caóticos en los planos de fase.

Sistema de Lorenz

En la figura 4, se muestra el atractor del sistema continuo adimensional[6] en Scilab. En la figura 5 se muestra el atractor del modelo discretizado (1) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

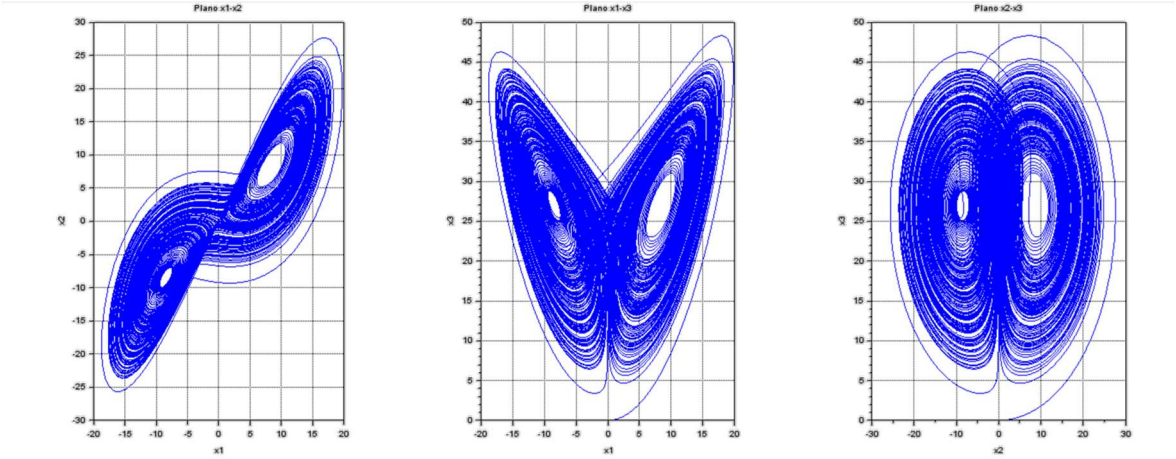
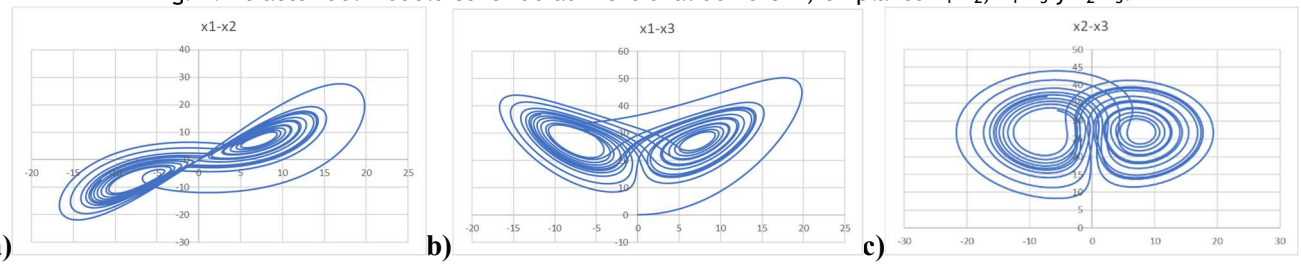


Fig. 4. Atractor del modelo continuo adimensional de Lorenz, en planos x_1-x_2, x_1-x_3 y x_2-x_3.



a) **b)** **c)**

Fig. 5. Atractor del modelo discretizado de Lorenz (1) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema de Lü

En la figura 6, se muestra el atractor del sistema continuo adimensional[7] en Scilab. En la figura 7 se muestra el atractor del modelo discretizado (2) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

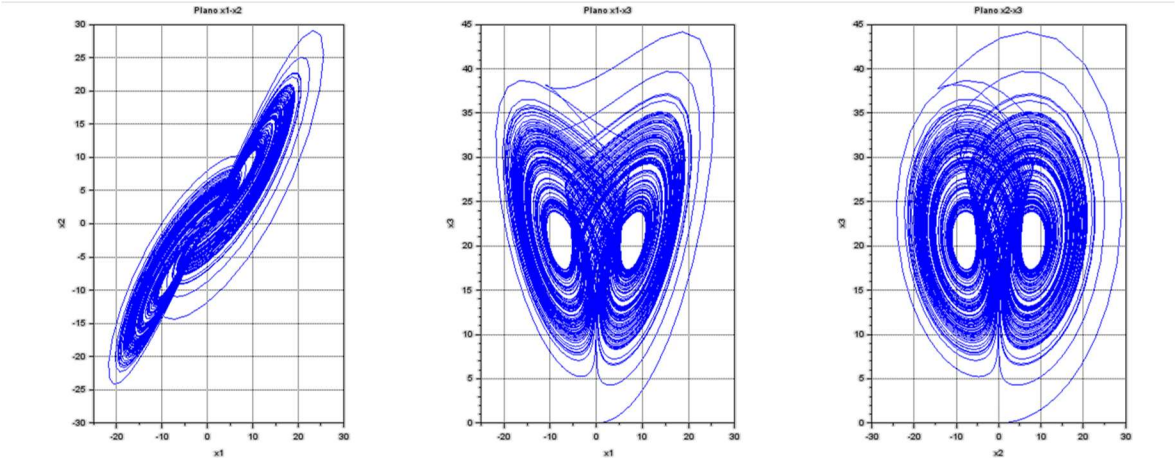
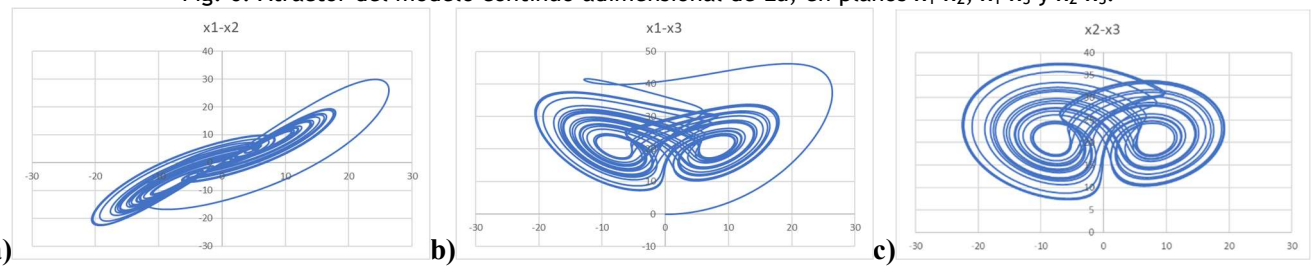


Fig. 6. Atractor del modelo continuo adimensional de Lü, en planos x_1-x_2, x_1-x_3 y x_2-x_3.



a) **b)** **c)**

Fig. 7. Atractor del modelo discretizado de Lü (2) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema de Chen

En la figura 8, se muestra el atractor del sistema continuo adimensional[8] en Scilab. En la figura 9 se muestra el atractor del modelo discretizado (3) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

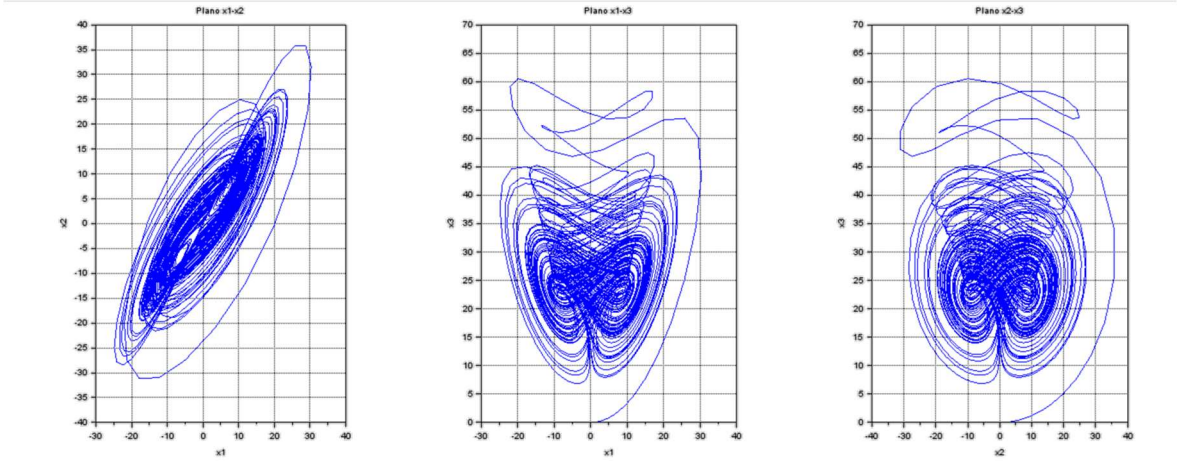
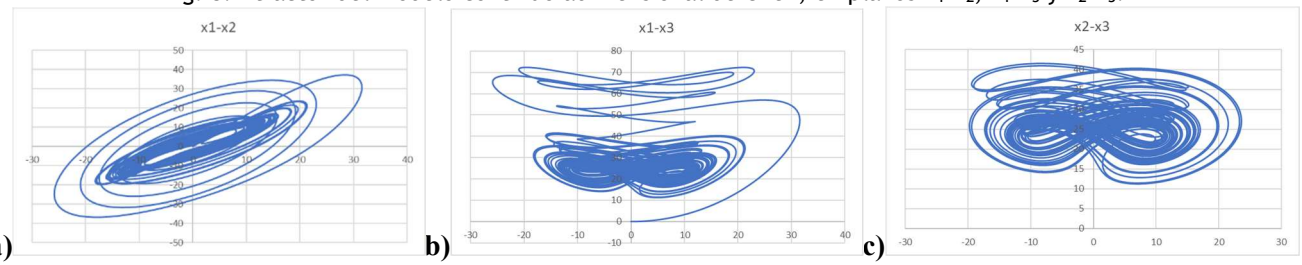


Fig. 8. Atractor del modelo continuo adimensional de Chen, en planos x_1-x_2, x_1-x_3 y x_2-x_3.



a) **b)** **c)**

Fig. 9. Atractor del modelo discretizado de Chen (3) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema de Rössler

En la figura 10, se muestra el atractor del sistema continuo adimensional[9] en Scilab. En la figura 11 se muestra el atractor del modelo discretizado (4) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

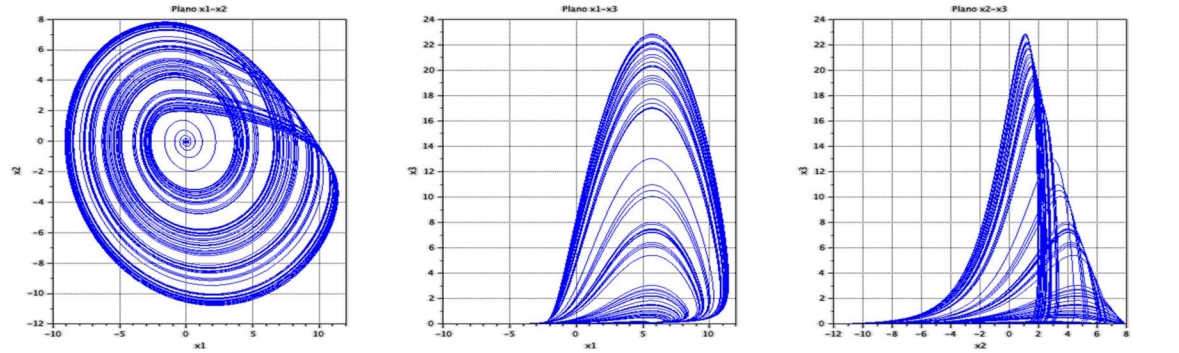
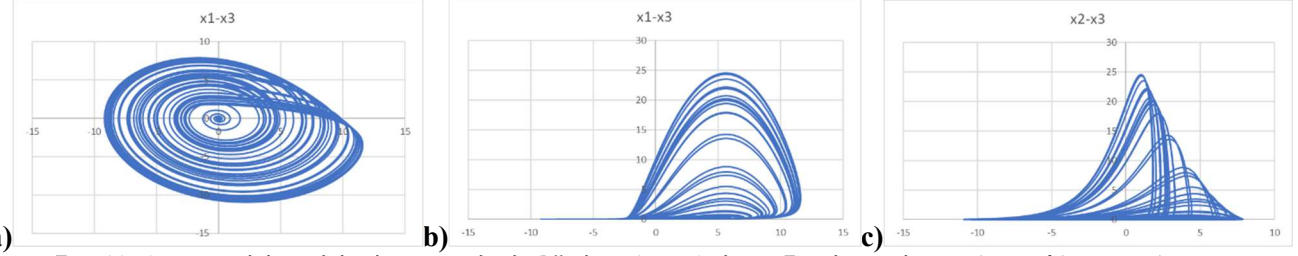


Fig. 10. Atractor del modelo continuo adimensional de Rössler, en planos x_1-x_2, x_1-x_3 y x_2-x_3.



a) **b)** **c)**

Fig. 11. Atractor del modelo discretizado de Rössler (4) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema Liu-Chen

En la figura 12, se muestra el atractor del sistema continuo adimensional[10] en Scilab. En la figura 13 se muestra el atractor del modelo discretizado (5) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

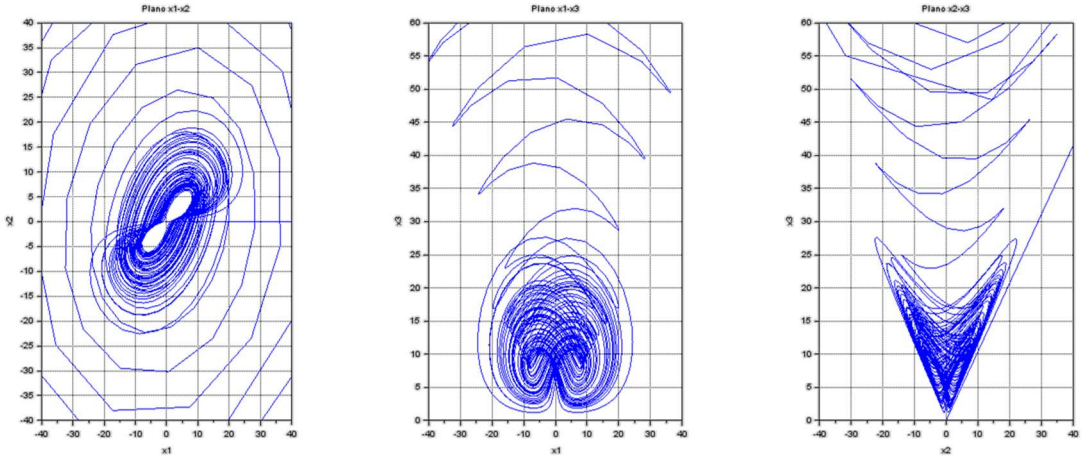
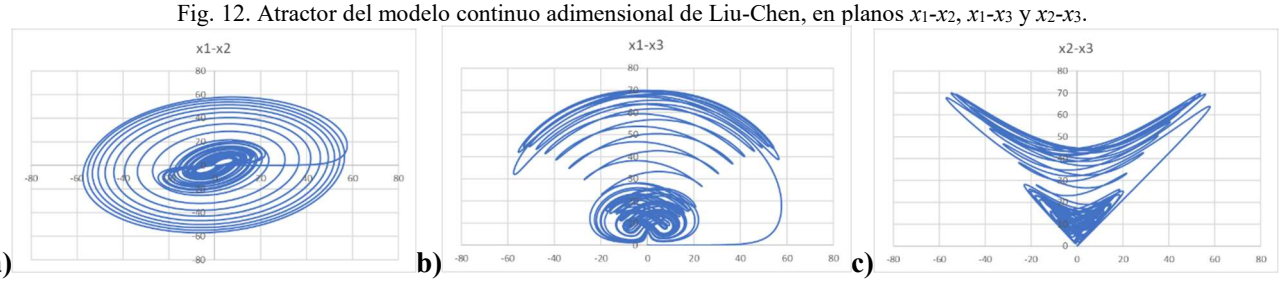


Fig. 12. Atractor del modelo continuo adimensional de Liu-Chen, en planos x_1-x_2, x_1-x_3 y x_2-x_3.



a) **b)** **c)**

Fig. 13. Atractor del modelo discretizado Liu-Chen (5) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema Malasoma

En la figura 14, se muestra el atractor del sistema continuo adimensional[11] en Scilab. En la figura 15 se muestra el atractor del modelo discretizado (6) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

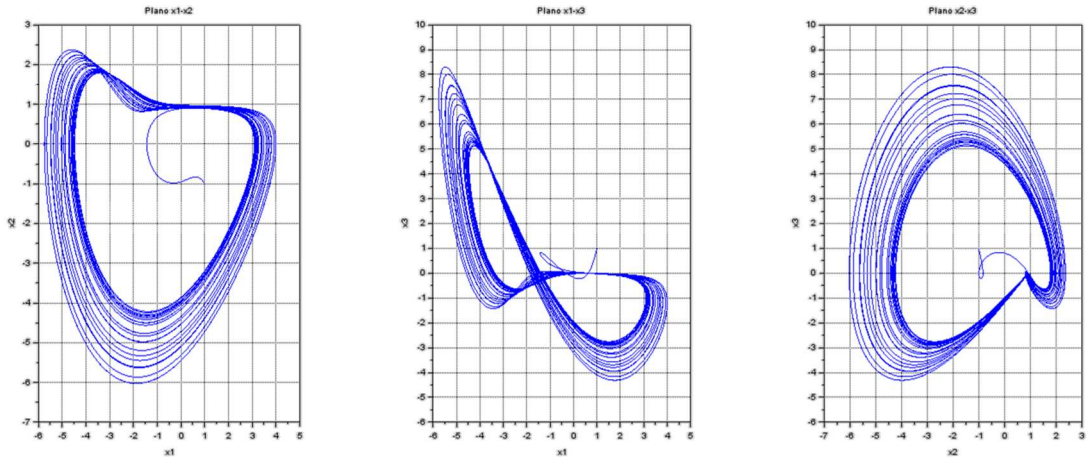


Fig. 14. Atractor del modelo continuo adimensional de Malasoma, en planos x_1-x_2, x_1-x_3 y x_2-x_3.

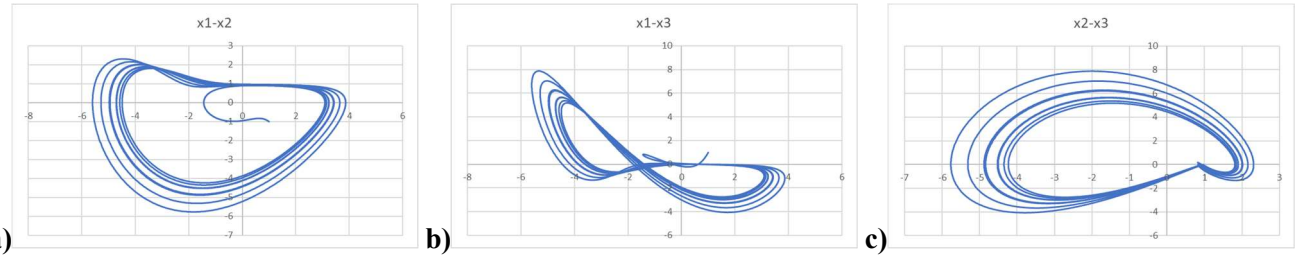


Fig. 15. Atractor del modelo discretizado Malasoma (6) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema MACM

En la figura 16, se muestra el atractor del sistema continuo adimensional[3] en Scilab. En la figura 17 se muestra el atractor del modelo discretizado (7) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

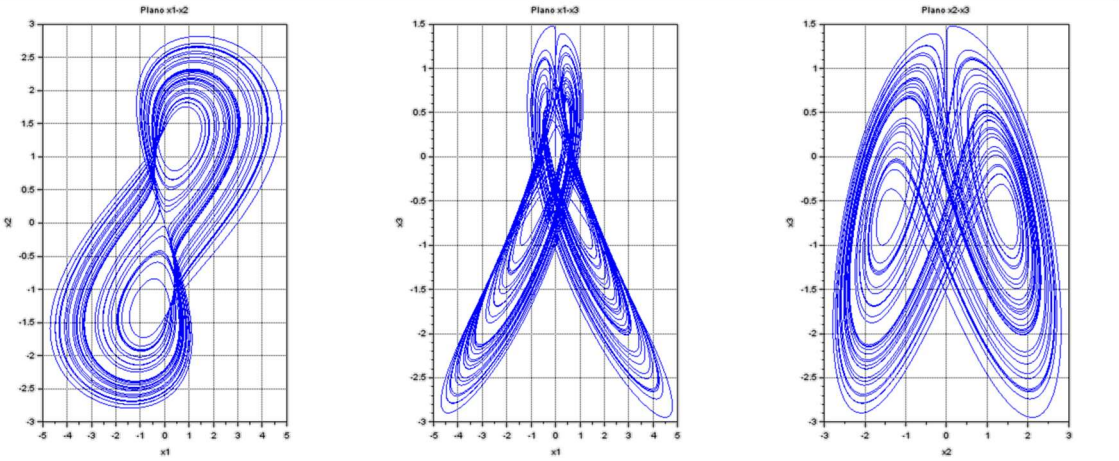


Fig. 16. Atractor del modelo continuo adimensional MACM, en planos x_1-x_2, x_1-x_3 y x_2-x_3.

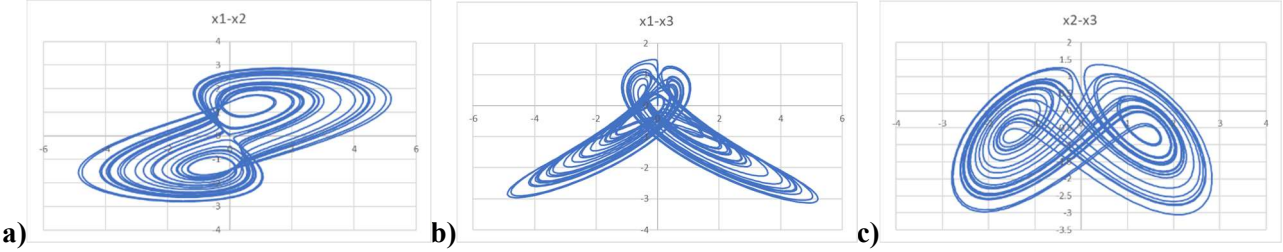


Fig. 17. Atractor del modelo discretizado MACM (7) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema afín a Rössler

En la figura 18, se muestra el atractor del sistema continuo adimensional[12] en Scilab. En la figura 19 se muestra el atractor del modelo discretizado (8) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

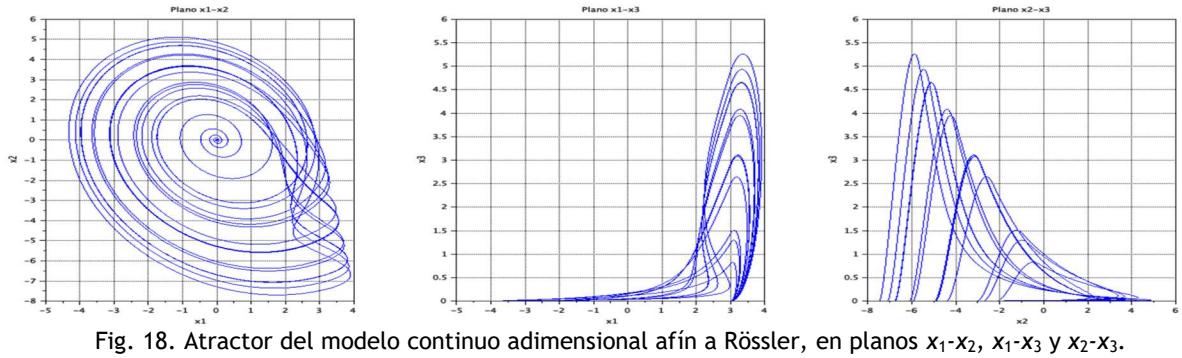


Fig. 18. Atractor del modelo continuo adimensional afín a Rössler, en planos x_1-x_2, x_1-x_3 y x_2-x_3.

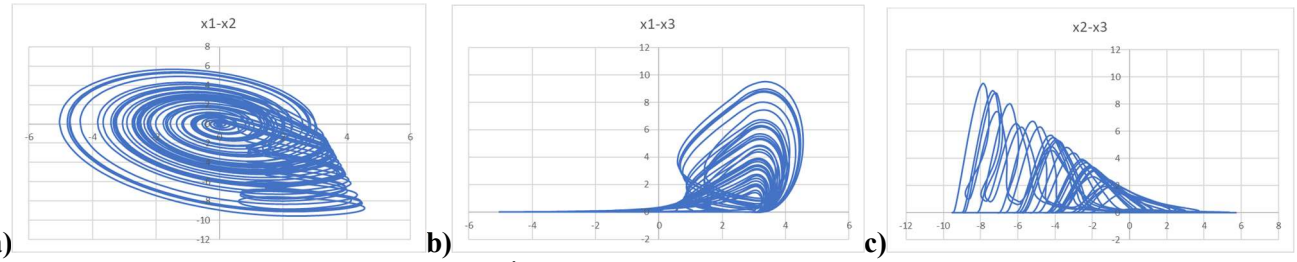


Fig. 19. Atractor del modelo discretizado afín a Rössler (8) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistema de Chua

En la figura 20, se muestra el atractor del sistema continuo adimensional[13] en Scilab. En la figura 21 se muestra el atractor del modelo discretizado (9) embebido en Arduino, usando la herramienta de adquisición de datos para MS Excel.

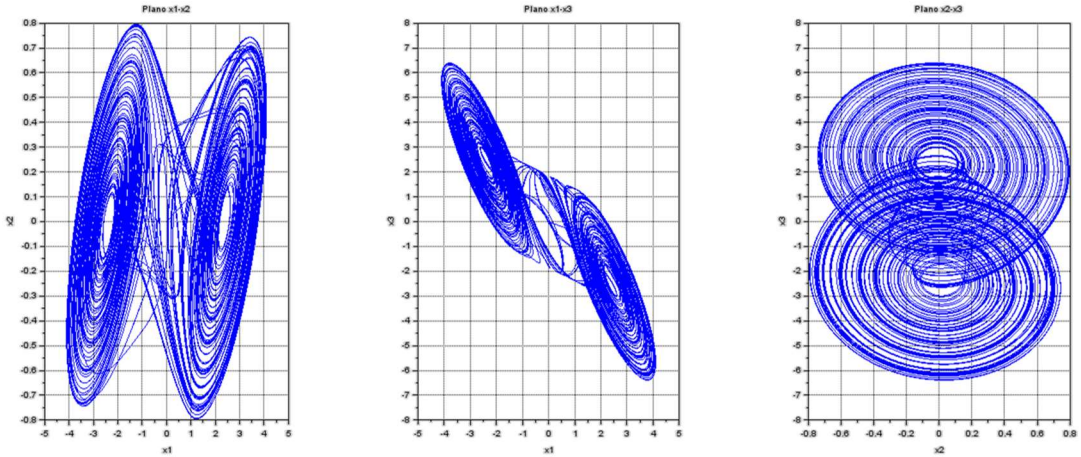


Fig. 20. Atractor del modelo continuo adimensional de Chua, en planos x_1-x_2, x_1-x_3 y x_2-x_3.

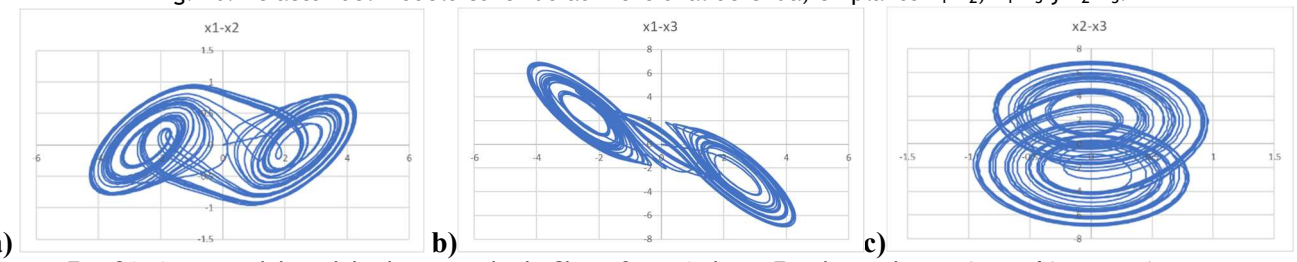


Fig. 21. Atractor del modelo discretizado de Chua (9) en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Sistemas Sprott

En la figura 22, se muestra el atractor del sistema continuo adimensional[14] en Scilab. En la figura 23 se muestra el atractor del modelo (10) embebido en Arduino para a), usando la herramienta de adquisición de datos para MS Excel.

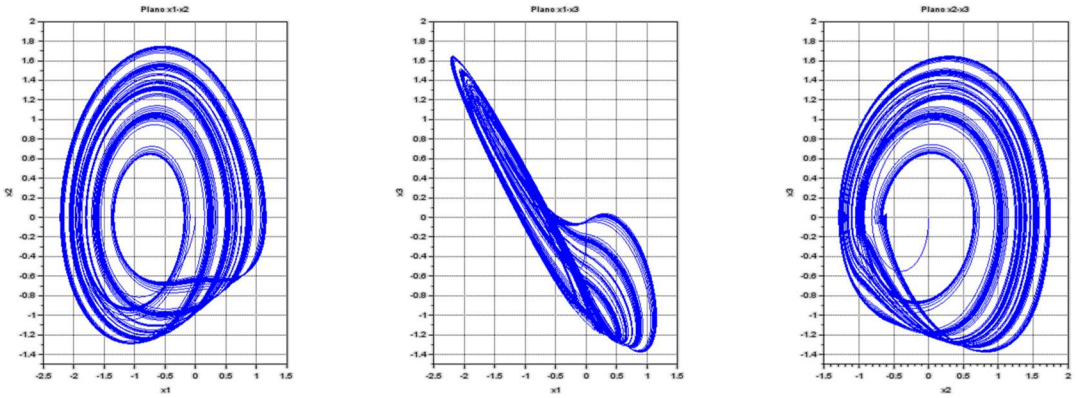


Fig. 22. Atractor del modelo continuo adimensional de Sprott a), en planos x_1-x_2, x_1-x_3 y x_2-x_3.

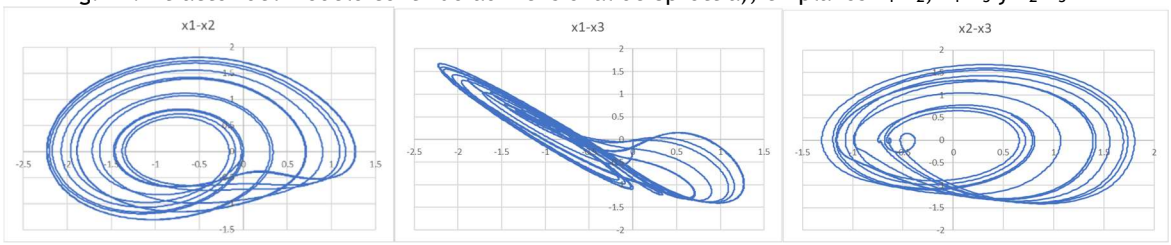


Fig. 23. Atractor del modelo discretizado de Sprott (10) para a), en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

En la figura 24, se muestra el atractor del sistema continuo adimensional[14] en Scilab. En la figura 25 se muestra el atractor del modelo (10) embebido en Arduino para b), usando la herramienta de adquisición de datos para MS Excel.

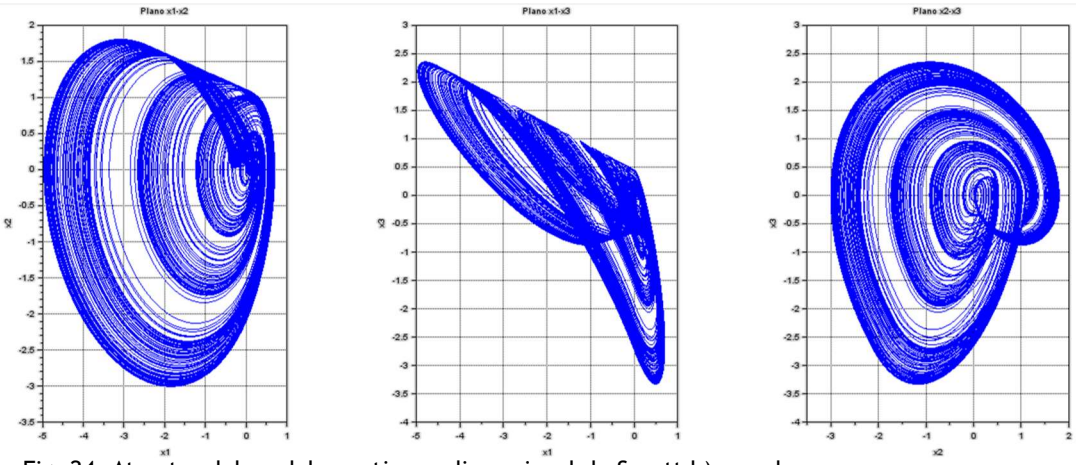


Fig. 24. Atractor del modelo continuo adimensional de Sprott b), en planos x_1-x_2, x_1-x_3 y x_2-x_3.

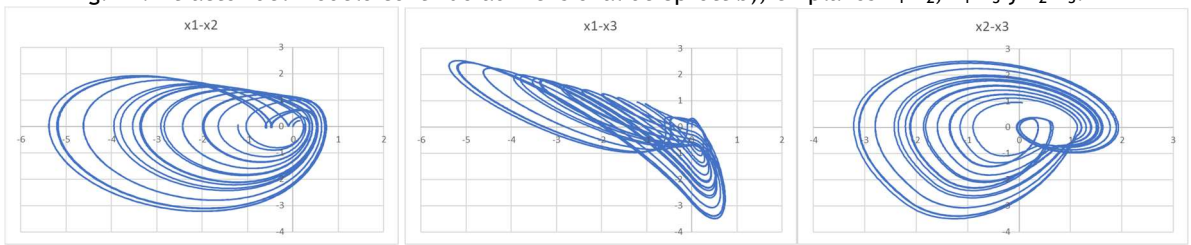


Fig. 25. Atractor del modelo discretizado de Sprott (10) para b), en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

En la figura 26, se muestra el atractor del sistema continuo adimensional[14] en Scilab. En la figura 27 se muestra el atractor del modelo (10) embebido en Arduino para c), usando la herramienta de adquisición de datos para MS Excel.

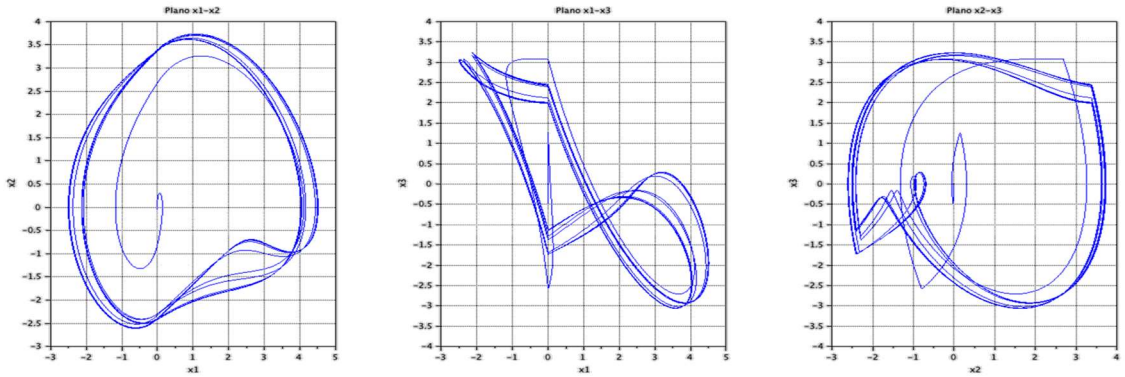


Fig. 26. Atractor del modelo continuo adimensional de Sprott c), en planos x_1-x_2, x_1-x_3 y x_2-x_3.

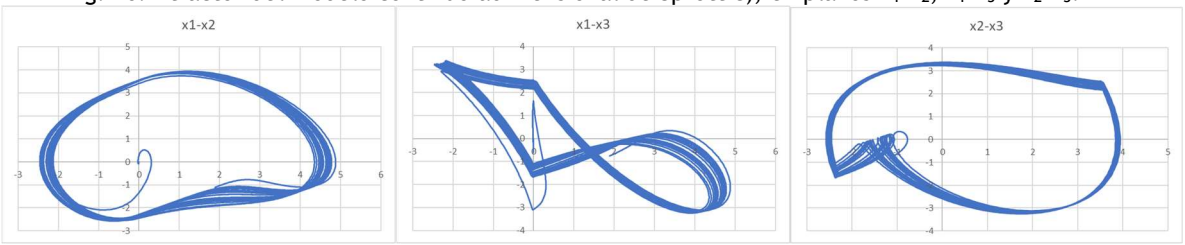


Fig. 27. Atractor del modelo discretizado de Sprott (10) para c), en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

En la figura 28, se muestra el atractor del sistema continuo adimensional[14] en Scilab. En la figura 29 se muestra el atractor del modelo (10) embebido en Arduino para d), usando la herramienta de adquisición de datos para MS Excel.

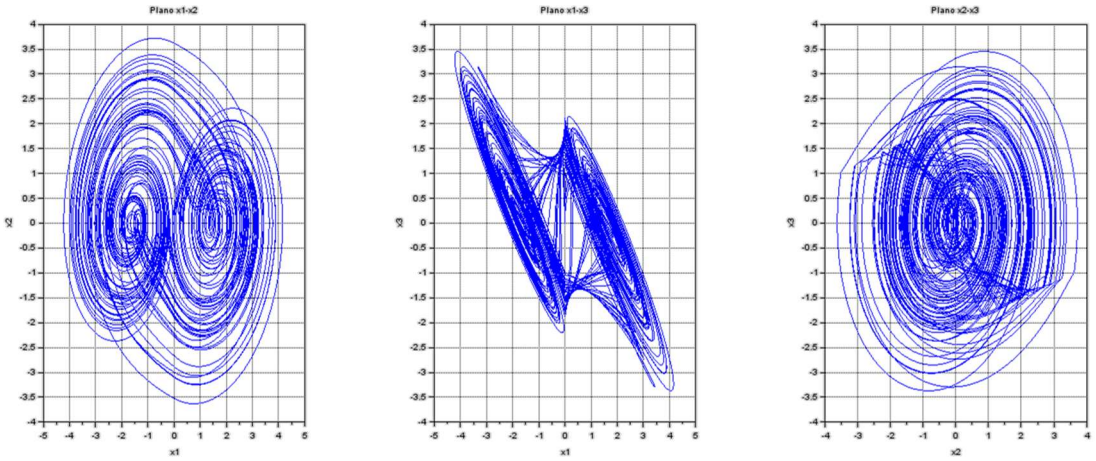


Fig. 28. Atractor del modelo continuo adimensional de Sprott d), en planos x_1-x_2, x_1-x_3 y x_2-x_3.

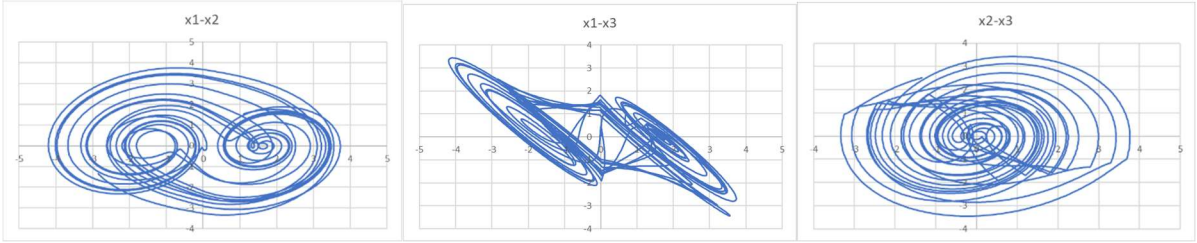


Fig. 29. Atractor del modelo discretizado de Sprott (10) para d), en Arduino-Excel, en planos **a)**x_1-x_2,**b)**x_1-x_3 y **c)**x_2-x_3.

Discusión de resultados

En las figuras 4 a 29, se puede apreciar que los atractores producidos con el microcontrolador Arduino presentan características de forma y amplitud congruentes con los atractores caóticos típicos de los modelos matemáticos continuos adimensionales respectivos. Por lo que puede concluirse que son válidos los modelos discretizados por el método de Euler presentados de (1) a (10). Como es bien sabido, mediante este método de discretización fue

complicado lograr que los atractores se mantuvieran estables, ya que, elegir un tiempo de muestreo muy grande o pequeño influye en la estabilidad del algoritmo. No obstante, se logró encontrar los tiempos de muestreo adecuados, basados en algunos trabajos de la literatura[1,4], así como selección arbitraria de sus valores empíricamente.

CONCLUSIONES

En esta tercera parte, se mostró la implementación digital de sistemas caóticos en la plataforma de código abierto Arduino. Particularmente, se utilizó el método de discretización de Euler en el tiempo para sistemas dinámicos, por ser uno de los métodos más simples mediante los cuales se obtiene su modelo en ecuaciones en diferencias. Aunque es complicado encontrar los tiempos de muestreo adecuados para la estabilidad de los atractores, se logró seleccionarlos para que los atractores obtenidos con Arduino, e ilustrados con Excel, fueran congruentes con los de sistemas caóticos continuos respectivos. Estas implementaciones digitales facilitan sus aplicaciones en diversos campos como el control y las comunicaciones, además que la misma metodología puede extenderse a otras plataformas digitales de programación y sistemas embebidos, así como utilizar mejores métodos de discretización.

REFERENCIAS

1. Chiu, R., Mora-González, M., and Lopez-Mancilla, D. Osciladores Caóticos implementados en Microcontrolador PIC18F. Nova scientia. (2013) 6(12): 60-77.
2. Valcárcel, S. Generación de Secuencias Caóticas para CDMA. Tesis, Universidad Politécnica de Madrid. (2003).
3. Méndez-Ramírez, R., Cruz-Hernández, C., Arellano-Delgado, A. and Martínez-Clark, R. A new simple chaotic Lorenz-type system and its digital realization using a TFT touch-screen display embedded system. Complexity, 2017(6820492): 1–13.
4. Méndez-Ramírez, R. Implementación de osciladores caóticos en sistemas embebidos y aplicaciones. Tesis de doctorado. Centro de Investigación Científica y de Educación Superior de Ensenada, 2018.
5. Platas-Garza, M.A., Zambrano-Serrano, E., Rodríguez-Cruz, J.R., Posadas-Castillo, C. Implementation of an encrypted-compressed image wireless transmission scheme based on chaotic fractional-order systems. Chinese Journal of Physics, 2020, In Press: https://doi.org/10.1016/j.cjph.2020.11.014
6. Lorenz, E.N. Deterministic Nonperiodic Flow. Journal of the Atmospheric Sciences, (1963) 20: 130-141.
7. Lü, J. and Chen, G. A New Chaotic Attractor Coined. International Journal of Bifurcation and Chaos, (2002) 12, 659-661.
8. Chen, G. and Ueta, T. Yet Another Chaotic Attractor. International Journal of Bifurcation and Chaos, (1999) 9, 1465-1466.
9. Rössler, O. E. An equation for continuous chaos. Physics Letters A. (1976) 57(5): 397-398.
10. Pehlivan İ, Kurt E, Lai Q, Basaran A, Kutlu M. A Multiscroll Chaotic Attractor and its Electronic Circuit Implementation, Chaos Theory and Applications. 2019; 1(1): 29-37.
11. Malasoma, J.M. A new class of minimal chaotic flows. Physics Letters A, 2002; 305: 52-58.
12. Zanin M, Sevilla-Escoboza JR, Jaimes-Reátegui R, García-López JH, Huerta-Cuellar G, Pisarchik AN. Synchronization attack to chaotic communication systems. Discontinuity, Nonlinearity, and Complexity. (2013) 1: 333-343.
13. Matsumoto, T. A Chaotic Attractor from Chua's Circuit. IEEE Trans. Circuits Syst. (1984) 31(12): 1055-1058.
14. Sprott, J.C. A new class of chaotic circuit. Physics Letters A, 2000; 266: 19-23.