Sistema de adquisición de datos con Python y Arduino

E. Machado-Díaz[1], H. Coto-Fuentes[1]

***Resumen*— Los sistemas de adquisición de datos son indispensables en el modelado matemático de sistemas de control, pues permiten obtener la curva característica por métodos experimentales. Con el objetivo de desarrollar un sistema que permita las mediciones de voltaje y la posterior obtención de dichos modelos, se utilizó una aplicación visual en Python y se diseñó un protocolo basado en comandos a través del puerto serie con una placa Arduino DUE encargada de realizar el muestreo. La interfaz cuenta con una gráfica que se actualiza cada segundo y otra que grafica un número determinado de muestras a un periodo de muestreo establecido por el usuario. Las muestras tomadas son almacenadas en un fichero Excel con el fin de permitir analizarlas con otros softwares especializados. Del trabajo realizado se concluyó que el uso de herramientas libres permite implementar sistemas de adquisición de datos de manera sencilla y económica.**

***Palabras claves*—Adquisición de datos, Arduino, Python.**

***Abstract*— Data acquisition systems are indispensable in the mathematical modeling of control systems. They let to obtain the characteristic curve by experimental methods. For the development of voltage measurements, we used a visual application in Python and a design of commands for its serial port communication with an Arduino DUE board that let to measure variables to help in obtaining the model of a control system. The interface has a graph that update every second and another that plot a certain number of samples to a sampling period established by the user. The samples taken are stored in an Excel file in order to allow them be analyzed with other specialized software. From the work carried out, we concluded that the use of free tools allow realize data acquisition systems in a simple and economic way.**

***Keywords*—Data Acquisition, Arduino, Python.**

XIV. INTRODUCCIÓN

En años recientes se ha incrementado el uso de controladores digitales en sistemas de control [1]. Se utilizan para alcanzar el desempeño óptimo de los sistemas y que tengan la máxima productividad.

La tendencia actual de controlar los sistemas en forma digital en lugar de analógica se debe principalmente a la disponibilidad de computadoras digitales de bajo costo y a las ventajas de trabajar con señales digitales en lugar de señales analógicas.

A. Muestreo de señales

La señal de tiempo discreto es aquella que tiene valores o está definida sólo en los puntos de tiempo discreto $t = t_n$, donde *n* sólo toma valores enteros. La gráfica de una señal de tiempo discreto siempre estará en términos de los valores de t_n contra la variable de tiempo entera *n*.

Una de las formas más comunes en las que surgen las señales de tiempo discreto es muestreando señales analógicas. Suponga que una señal x(t) se aplica a un interruptor electrónico que se cierra brevemente cada T segundos. Si el lapso durante el cual el interruptor se cierra es mucho más pequeño que T, la salida del interruptor puede considerarse como una señal de tiempo discreto x[t] que es una función de los puntos de tiempo discreto. La señal de tiempo discreto resultante se conoce como versión muestreada de la señal continua y a T se le conoce como periodo de muestreo [2].

Según el teorema de muestreo de Nyquist – Shannon, para poder replicar con exactitud la forma de una onda es necesario que la frecuencia de muestreo sea como mínimo el doble de la máxima frecuencia a muestrear, sin embargo en términos prácticos el doble no es suficiente, entre mayor sea el número de muestras y más alta la frecuencia de muestreo se obtendrá una señal con más fidelidad [3].

Los dispositivos de muestro y retención se emplean de manera extensa en los sistemas de control digital y de datos muestreados. Son dispositivos que convierten una señal analógica en un tren de pulsos de amplitud modulada o en una señal digital, efectuando las tareas de cuantización y codificación. Así mismo mantiene o congela el valor del pulso o de la señal durante cierto tiempo [4].

B. Python

Python es un lenguaje de programación multiplataforma de código libre poderoso y fácil de aprender. Cuenta con estructuras de datos eficientes y de alto nivel, así como un enfoque simple pero efectivo en la programación orientada a objetos [5].

Es un lenguaje de programación que actualmente se utiliza en el ámbito ingenieril por sus librerías para ciencia e ingeniería, las que permiten resolver distintas ecuaciones

[1]*Instituto Tecnológico Superior de Lerdo,* Av. Tecnológico N 1555 Sur Periférico Gómez – Lerdo Km. 14.5 C.P. 35150 Cd. Lerdo. Durango. Eduardo.madied@gmail.com

y obtener gráficas muy parecidas a entornos de desarrollo más completos y complejos como MATLAB.

Se pueden desarrollar interfaces gráficas de usuario complejas utilizando el módulo Tkinter combinadas con los módulos Scipy (ciencia e ingeniería), Sympy (matemáticas simbólicas) y Matplotlib (gráficas) que permitan realizar aplicaciones de código libre potentes sin la necesidad de un software especializado con altos costos en licencias.

Las características de Python lo vuelven una herramienta ideal para aquellas personas que no tienen un conocimiento amplio en el desarrollo de software, pero poseen mínima experiencia en programación, siendo un lenguaje fácil de aprender y con basta documentación tanto en libros como en la Red. Al tener una consola interactiva, permite experimentar o manipular resultados sin la creación de un script [6].

Actualmente Python ha aumentado considerablemente su popularidad debido a ser el principal lenguaje de programación utilizado en mini ordenadores como lo es la Raspberry Pi. Se han hecho proyectos combinando el lenguaje con diversas tarjetas de código libre como lo es Arduino. Ejemplos de ellos son sistemas de riego automáticos donde se obtienen lecturas de sensores y se guardan en una nube digital con el fin de monitorear las variables por internet [7], así como sistemas de instrumentación virtual remota con el fin de elaborar prácticas de control de procesos en una plataforma web [8].

Debido a la facilidad de programación del lenguaje Python así como Arduino, cada vez es más frecuente el uso de estas herramientas en la docencia para brindar a los alumnos de nivel medio superior de competencias que puedan ser útiles al momento de estudiar una carrera. Ejemplos de ellos son sistemas robóticos basados en Python y Arduino a nivel secundaria que son de bajo costo y pueden ser construidos por los mismos docentes y alumnos, brindando de mejor equipamiento a la institución educativa [9].

Sin embargo, no solo se han utilizado las herramientas libres para el uso industrial, existen diferentes aplicaciones en medicina como la descrita en [10] que permite medir señales fisiológicas como lo son la temperatura corporal para que sean procesadas y analizadas a un bajo costo.

XV. Parte Técnica del Artículo

A continuación, se presenta la metodología implementada para la elaboración del proyecto.

A. Adquisición de datos y protocolo de comunicación en Arduino.

Para la adquisición de datos se optó por utilizar la plataforma Arduino debido a que es de código abierto y mantienen una gran popularidad, por lo que tiene una amplia documentación y tutoriales que permiten su manejo por casi cualquier persona involucrada en el desarrollo electrónico. Existen una gran variedad de placas Arduino, para este proyecto se utilizó una placa Arduino DUE pues presenta mejores características que una placa Arduino Uno para la adquisición y procesamiento de datos.

Se realizó la programación para la adquisición de datos por medio de un sensor ultrasónico HC-SR04, debido a que es uno de los sensores más populares, y la lectura de las entradas analógicas de la placa Arduino.

Debido a que la comunicación entre el Arduino y la aplicación en Python se realiza por comunicación serie, es necesario diseñar un protocolo de comandos que permitan el envío de instrucciones entre un medio y otro, con el fin de evitar la pérdida de información y tener instrucciones establecidas que puedan utilizarse en diferentes aplicaciones dependiendo de lo que se requiera.

Existen diferentes formas de realizar el protocolo, para este caso se utiliza una metodología basada en la librería brindada por los microcontroladores PSOC de Cypress mediante separadores en la trama [11].

El sistema de comandos es una cadena de texto que es enviada por el puerto serie al Arduino conformada por dos tramas, la primera es el comando y la segunda el parámetro a almacenar, separados entre ellas por un carácter establecido como separador (“@”) y posteriormente un símbolo de fin de trama (“/”). Un ejemplo de cadena de texto es el siguiente: Comando@parámetro/

Los comandos que pueden ser reconocidos se guardan en un *Array* y los parámetros en otro. De acuerdo a la posición del comando en el arreglo, es el índice de su parámetro. En la Figura 1 se muestra un ejemplo de los arreglos utilizados para esta aplicación.

```
int DATA[10];
const String CMD[]= {“SetSamples”,”SetTime”,
“StartSample”};
byte NUM_CMD = 2;
```

Figura 1. Ejemplos de comandos

El arreglo “DATA” es donde se guardan los parámetros y el arreglo “CMD” es donde se almacenan los comandos. “NUM_CMD” corresponde al número de comandos programados que serán n-1 debido a que el índice comienza en 0. Cada que se agrega un comando es necesario aumentar esta variable para poder iterar.

En la Figura 2 se muestra el diagrama de flujo utilizado para el reconocimiento de comandos y el almacenamiento de parámetros.

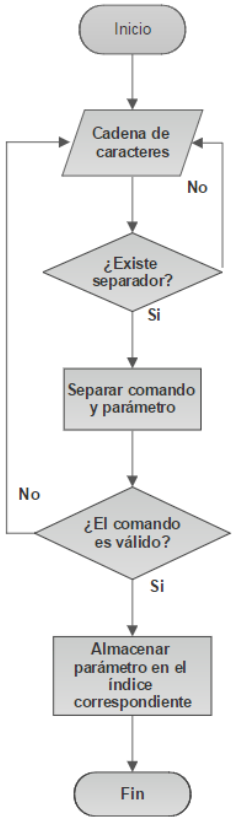

Figura 2. Diagrama de flujo para reconocimiento de comandos

Una vez almacenado el valor del parámetro en el arreglo. Éste no cambia hasta que sea ingresado de nuevo o hasta que la placa Arduino sea reiniciada, permitiendo acceder a él o cambiarlo las veces que sea necesario.

B. Diseño de la Interfaz Visual en Tkinter

La interfaz fue diseñada utilizando el paquete Anaconda que instala Python 3.5, así como las librerías más comunes para su uso científico y el IDE Spyder. La Figura 3 muestra el ejemplo básico de una ventana en Tkinter.

```
#Se importa la librería
import tkinter as tk
#Se crea la ventana
top = tk.Tk()
top.mainloop()
```

Figura 3. Ejemplo básico de aplicación

El primer paso es importar el módulo o la librería Tkinter, posteriormente se crea un objeto de ventana que corresponderá a la ventana principal. Para que el código de la aplicación funcione en forma cíclica es necesario colocar el método "*mainloop*" que permite tener en un ciclo infinito a la aplicación. La Figura 4 muestra el resultado obtenido con el código anterior.

Figura 4. Dibujo de la ventana principal

Los elementos como botones, cajas de texto, también llamados widgets, son dibujados mediante código. Existen dos tipos de formas de trabajar los widgets o elementos de ventana: mediante uso de rejilla, en donde la ventana se utiliza como tabla y se especifica el número de columna y de fila, o en forma de pila que permite dibujar un elemento y posteriormente elegir en qué posición será dibujado el siguiente. Para esta aplicación se utiliza el método de pila, ya que es más fácil trabajar con él si no se tiene un diseño previo en una cuadrícula de la aplicación.

```
import tkinter as tk
def funcion():
    texto.configure(text="Hola mundo")

top = tk.Tk()
top.geometry("100x100")
texto = tk.Label(top, text="Texto")
text.pack(padx=10,pady=15,side="top")
boton = tk.Button(top,text="boton",command=funcion)
boton.pack(padx=10,pady=10)

top.mainloop()
```

Figura 5. Ejemplo de widgets

En el código de la Figura 5 se muestra un ejemplo que permite utilizar un botón para realizar una acción. El método *geometry* permite establecer las medidas en pixeles de la ventana principal, posteriormente cuando se crea un elemento de ventana es necesario colocar los diferentes argumentos de cada uno. El primer argumento corresponde al nombre de la ventana en la cual se va a dibujar el elemento, seguido de diferentes tipos que pueden ser el texto por defecto, el color o el fondo.

Una vez declarado el elemento se utiliza el método *pack()* que lo dibuja en la ventana, si se declara el elemento pero no se coloca esta instrucción, éste no se mostrará. Este método puede trabajar con diferentes argumentos como lo son la tolerancia entre cada eje y el argumento *side*, permite decirle a la aplicación en qué posición será dibujado el siguiente elemento con referencia al elemento creado.

En los elementos que pueden ser manipulados por el usuario, como los botones, es necesario programar una función que será ejecutada cada que sea presionado el botón. El nombre de la función a ejecutar será el argumento *command* en la declaración del objeto. La Figura 6 muestra el resultado del código mostrado.

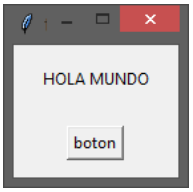

Figura 6. Ejemplo de widgets

C. Recolección de muestras con la Interfaz

Para el uso de la comunicación serial en Python es necesario utilizar el paquete Pyserial instalado por defecto al utilizar el paquete Anaconda. De igual forma se ha utilizado la entrada analógica A0 de la placa Arduino Due, en la cual se tienen que tener los cuidados necesarios al momento de colocar la señal a muestreas pues ésta no puede sobrepasar los 3.3V que son el voltaje de trabajo de la tarjeta Arduino Due. En la Figura 7 se muestran las instrucciones básicas para el manejo del puerto serial con Python.

```
import serial
#Abrir el puerto de comunicación
arduino = serial.Serial('COM3',9600)
texto = input("Ingresa una cadena: ")
arduino.write(texto.encode('ascii'))
arduino.readline() #leer hasta un salto de línea
```

Figura 7. Instrucciones de puerto serie

Se importa la librería y se crea un objeto de tipo serial donde se presentan dos argumentos principales, el puerto serie al cual se encuentra conectado el dispositivo, y la velocidad en baudios a la que se va a realizar la comunicación.

Para el envío de datos se debe codificar la cadena de texto en un formato válido para el envío de la información. Al tener una cadena de texto se le aplica el método *encode()* y como argumento la codificación ascii, si esto no se realiza la información enviada no corresponderá a la cadena de texto establecida.

En la recepción de datos existe el método *readline()* que permite leer el buffer del puerto serie hasta que exista un retorno de línea, en el caso de la cadena enviada por el Arduino es necesario colocar un salto de línea cada que se envía una muestra.

Debido a los tipos de datos en Python, la cadena recibida viene acompañada por caracteres especiales que no permiten la conversión del dato directo de cadena a tipo entero. La cadena recibida tiene el formato: b'Dato_numerico\r\n' por lo que es necesario solamente dejar el dato numérico, para ello se utilizó el algoritmo que se muestra en la Figura 8 que permite obtener la cadena de datos de interés y guardarla como muestra. Como se observa en el formato, siempre se tienen 3 caracteres antes de la muestra y 5 después de ella por lo que el dato es la cadena entre esos índices.

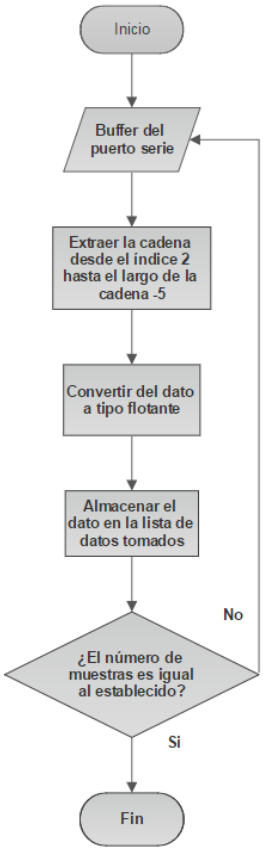

Figura 8. Diagrama de flujo para la conversión del dato

Los datos obtenidos son guardados en listas de Python que pueden iterarse para posteriormente graficarlos o almacenarlos en el fichero Excel. Para no saturar el buffer del puerto serie, el arreglo del tiempo es generado de forma manual utilizando los parámetros dados en la entrada de texto de la interfaz como se muestra en la Figura 9.

```
T_Muestreo = float(T_Muestreo/1000)
T_Max = float(samples)*T_Muestreo
for x in range(0,Num_Muestras):
    self.segundos.append(float(T_Muestreo*x))
```

Figura 9. Creación de la lista del tiempo

XVI. RESULTADOS

La interfaz visual consta de 3 ventanas principales: el menú principal mostrado en la Figura 10, el cual contiene los botones para la elección de las características del programa, como es la gráfica animada y el muestreo.

Figura 10. Menú principal de la interfaz

En la ventana de Live Plot se presenta la gráfica animada utilizando la librería Matplotlib que puede verse en la Figura 11, tiene una caja de opciones donde se muestran los puertos COM disponibles, así como los botones para realizar la conexión, el inicio de muestreo, pausa de la gráfica y reanudación.

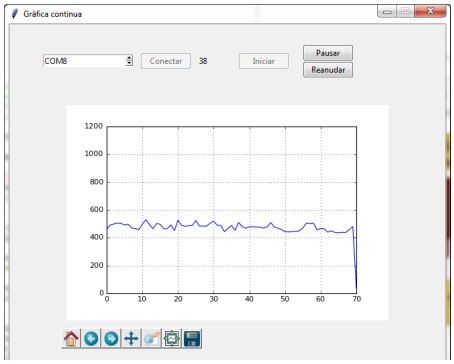

Figura 11. Ventana de gráfica animada

El tiempo de refrescamiento de la gráfica es de un segundo, por lo que se utiliza para muestras lentas, esto debido a las limitaciones de la librería. Así mismo tiene una ToolBox que permite manejar la gráfica, hacer acercamientos y guardar como imagen de formato PNG la vista actual con el fin de servir para documentar las muestras. La gráfica toma 50 muestras que son las que se observan, después de ellas se elimina la última muestra almacenada y se cambia por la muestra actual.

La opción de adquisición de datos se muestra en la Figura 12 donde se observa una gráfica de 50 muestras con un tiempo se muestreo de 100ms de datos aleatorios del conversor digital – análogo de la placa Arduino Due. La ventana cuenta con el botón de conectar, entradas numéricas donde se colocan el número de muestras y el tiempo de muestreo necesario, así como el botón de enviar que comienza el muestreo y exportar, que regresa el archivo Excel con las muestras tomadas.

Figura 12. Ventana de adquisición de datos

En la Figura 13 se presenta un ejemplo de una gráfica con muestras aleatorias almacenada en formato PNG que utilizarse para su análisis o en la documentación de los experimentos.

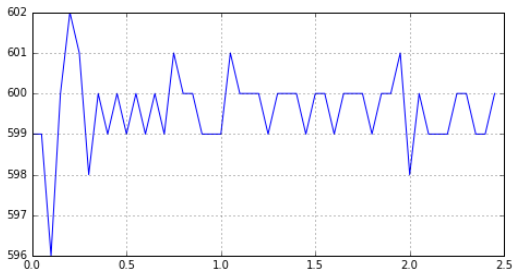

Figura 13. Ejemplo de gráfica en PNG

XVII. DISCUSIÓN, CONCLUSIÓN Y RECOMENDACIONES

La adquisición de datos es muy importante cuando se quiere conocer el modelo matemático de un sistema de control o para tener un control de muestras de sensores o actuadores. Actualmente existen sistemas que realizan éstas tareas o permiten programar interfaces visuales como lo son LabView o Matlab, sin embargo, ambas tienen un alto costo de licencia de uso que no puede ser pagada por la mayoría de las personas, por lo que el uso de herramientas libres como Arduino y Python presenta una buena alternativa.

La placa Arduino Due es una buena alternativa cuando se requiere medir señales a bajo nivel o se requiere una resolución mayor a la de un Arduino Uno u otra placa

común, así mismo trabaja a una velocidad mayor y tiene un mejor procesamiento. Sin embargo, ya que el sistema se comunica mediante protocolo serial, se pueden desarrollar los sistemas de comandos para diferentes plataformas que cuenten con esta característica como lo son los microcontroladores PIC de Microchip o PSOC por mencionar algunos.

El uso de un protocolo de comandos es una de las partes más complicadas en la programación de este tipo de interfaces, pero es necesaria para el buen funcionamiento de ésta ya que existe menos pérdida de información o confusión si la trama enviada de un dispositivo a otro no llega completa. Actualmente se trabaja en el apartado se sintonización de controladores mediante el método Ziegler-Nichols.

XVIII. Agradecimientos

Se agradece al Instituto Tecnológico de Lerdo por la formación profesional y en especial a la División de Posgrado por la ayuda y asesorías brindadas en este proyecto.

XIX. Apéndices

A. Código para la exportación de datos a Excel utilizando XLWT

```
wb = xlwt.Workbook()
ws = wb.add_sheet("Hoja 1")
for i, lista in enumerate(self.datos):
   ws.write(i,0,lista)
for j,lista2 in enumerate(self.segundos):
   ws.write(j,1,lista2)
wb.save("Muestras.xls")
```

B. Código para la obtención de muestras en Arduino.

```
for (int i=1; i<=DATA[i]; i++)
{
   Serial.println(analogRead(A0));
   delay(DATA[2]);
}

Serial.println("$");
DATA[3]=0;
```

XX. Referencias

[1] K. Ogata, Sistemas de control en tiempo discreto, Mexico: Prentice Hall, 1996.

[2] E. Kamen, Fundamentos de señales y sistemas usando la Web y MATLAB, México: Pearson Educación, 2008.

[3] K. Janschek, Mechatronic System Design: Methods, Models, Concepts, United States: Springer, 2012.

[4] B. Kuo, Sistemas de Control Digital, México: Grupo Editorial Patria, 2007.

[5] G. v. Rossum, El tutorial de Python, Argentina: Python Software Fundation, 2009.

[6] G. Borrell, «Python como entorno de desarrollo científico,» Universidad Politécnica de Madrid, Madrid, 2008.

[7] G. Escalas, «Diseño y desarrollo de un prototipo de Riego automático controlado con Raspberry Pi y Arduino,» Universidad Politécnica de Calatuña, España, 2014.

[8] E. Machado, «Prácticas de control de procesos utilizando instrumentación virutal remota,» Instituto Tecnológico Superior de Lerdo, México, 2016.

[9] J. Vega, «Entorno docente con Arduino y Python para Educación,» *Colegio Ntra. S.a Sagrado Corazón,* 2016.

[10] J. Arechalde, «Analizador de señales fisiológicas portable basado en plataformas low-cost,» *Trabajos Académicos-Escuela de Ingeniería de Bilbao,* 2016.

[11] Cypress, «Cypress,» Cypress Perform, 27 Septiembre 2017. [En línea]. Available: http://www.cypress.com/file/50321/download. [Último acceso: 15 Febrero 2017].

XXI. Biografía

Machado Díaz Eduardo. Nació en la ciudad de Gómez Palacio, Dgo. Obtuvo el título de Ingeniero Electrónico en Mecatrónica y Automatización del Instituto Tecnológico Superior de Lerdo en el año de 2017. Actualmente cursa la Maestría en Ingeniería Mecatrónica en el mismo instituto.

El ingeniero ha trabajado en proyectos de instrumentación virtual remota y diseño de interfaces. Sus áreas de interés son la instrumentación, la domótica y la automatización.

Hesner Coto Fuentes. Ingeniero en Automática egresado del Instituto Superior Politécnico José Antonio Echeverría, La Habana, Cuba. Maestro en Diseño de Sistemas Electrónicos por el Centro de Investigaciones en Microelectrónica, La Habana, Cuba, y Doctor en Ciencias en Ingeniería Eléctrica con especialización en Instrumentación Electrónica por el Instituto Tecnológico de la Laguna, Torreón, Coahuila, México.

En su trayectoria como docente - investigador se ha especializado en las áreas de Instrumentación Virtual y desarrollo de sistemas electrónicos aplicados a la medicina, la industria y el medio ambiente.