ADQUISICIÓN DE DATOS DE BAJO COSTO VÍA; MATLAB SIMULINK-ARDUINO

Salvador Tavera Mosqueda1
staveram1500@alumno.ipn.mx Ramón Silva Ortigoza1
Eduardo Hernández Márquez1
Alfredo Roldán Caballero1
Cristofer Mateo Rodríguez López 1
José Rafael García Sánchez2

1 Instituto Politécnico Nacional CIDETEC Área de Mecatrónica Unidad Profesional Adolfo López Mateos
2Universidad Autónoma Metropolitana Unidad Lerma Departamento de Procesos Productivos

Resumen

Las aplicaciones en las que se puede utilizar una DAQ son, por ejemplo; instrumentación, como interfaz hombre máquina (HMI), automatización, entre otros. Las DAQ profesionales como las desarrolladas por compañías internacionales como National Instrument, DsPace, Siemens, entre otras, ofrecen soluciones integrales. Sin embargo, adquirir una tarjeta de las mencionadas, resulta ser muy costoso. Por lo anterior, en este trabajo se presenta una alternativa de bajo costo para aplicaciones estudiantiles haciendo uso de Arduino para reemplazar una DAQ profesional. Con la finalidad de mostrar la efectividad de Arduino como DAQ se lleva a cabo una comparación de resultados de simulación de un circuito diseñado para carga y descarga de un capacitor contra los resultados obtenidos en la adquisición de datos en tiempo real del mismo circuito, de esta forma se sientan las bases para realizar una adquisición de datos eficiente.

I. Introducción

Una DAQ actúa como la interfaz entre una PC y variables físicas [1]. Su propósito principal es digitalizar las señales proporcionadas por sensores que registran los cambios de variables como temperatura, humedad, sonido, entre otras. Lo anterior es con la finalidad de que puedan ser interpretadas por una PC; para su visualización, generar bases de datos o controlar procesos. Los tres subsistemas que, en general, componen a un dispositivo DAQ son; 1) circuito acondicionador de señales, 2) convertidor analógico-digital (ADC) y 3) bus de comunicación con una PC.

El recurso de hardware embebido Arduino, es una plataforma electrónica de hardware y software abierto [2], que incorpora un microcontrolador reprogramable con entradas y salidas definidas como analógicas o digitales. Por ser un recurso libre, se han diseñado una gran cantidad de aplicaciones en diferentes áreas, por mencionar algunas a saber: medicina, robótica, entretenimiento, automotriz. La popularidad de Arduino ha sido tal que, empresas como MathWorks, desarrolladora de Matlab, ha incluido herramientas para la comunicación y programación de Arduino en su software. En la Figura 1, se muestra, mediante un diagrama a bloques, la conexión que se realiza para lograr la comunicación entre Arduino y Matlab-Simulink.

2. Instalación de complemento en Matlab para Arduino

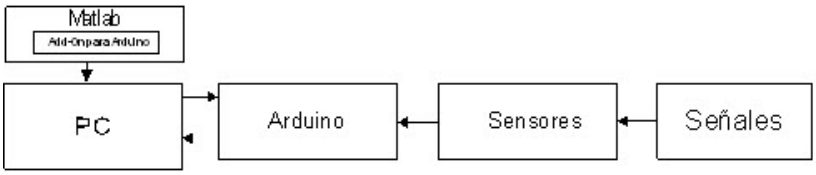

Figura 1: Comunicación entre Arduino y PC vía Matlab-Simulink

Para realizar la comunicación con el hardware embebido Arduino se debe instalar en Matlab el Add-on necesario. A continuación, se describen dos formas para realizar dicha instalación.

Es posible descargarlo desde la página de MathWorks [3] en https://goo.gl/BK5u5H. Una vez descargado, abrir Matlab, seleccionar la carpeta donde se almacenó el archivo y ejecutarlo, ver Figura 2.

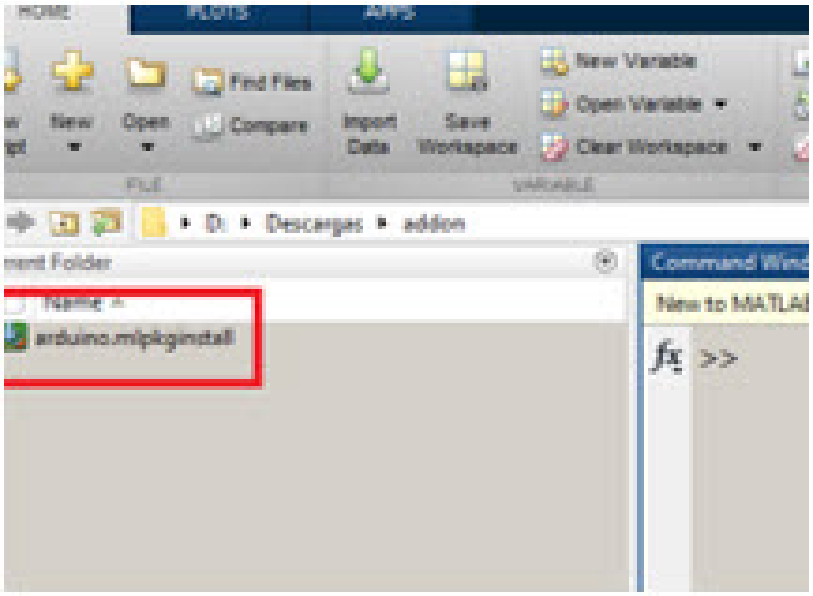

Figura 2: Archivo de comunicación con Arduino.

Es recomendable crear una carpeta de instalación para el Add-on de arduino. Posterior a seleccionar la carpeta de instalación y ejecutar el archivo, solo basta con aceptar los términos de licencia e instalar, ver Figura 3.

El siguiente método requiere crear una cuenta gratuita en MathWorks, posteriormente realizar los siguientes pasos:

1. Seleccionar el icono “Add-Ons”.
2. Elegir “Obtener Paquetes de Soporte para Hardware”
3. Seleccionar la opción “Instalar desde Internet”.
4. Buscar y seleccionar soporte para Arduino.
5. Elegir el paquete para Simulink que de soporte a Arduino.

Tras realizar los pasos anteriores y continuar, aceptar los términos de licencia para concluir la instalación. En la Figura 4 se muestran los pasos anteriormente descritos.

3. Configuración de Matlab-Simulink para Arduino

Una vez que se ha instalado con éxito el Add-on, se configuran los parámetros necesarios para lograr la comunicación entre el hardware y software. A continuación, se describen los pasos a seguir para lleva a cabo dicha configuración.

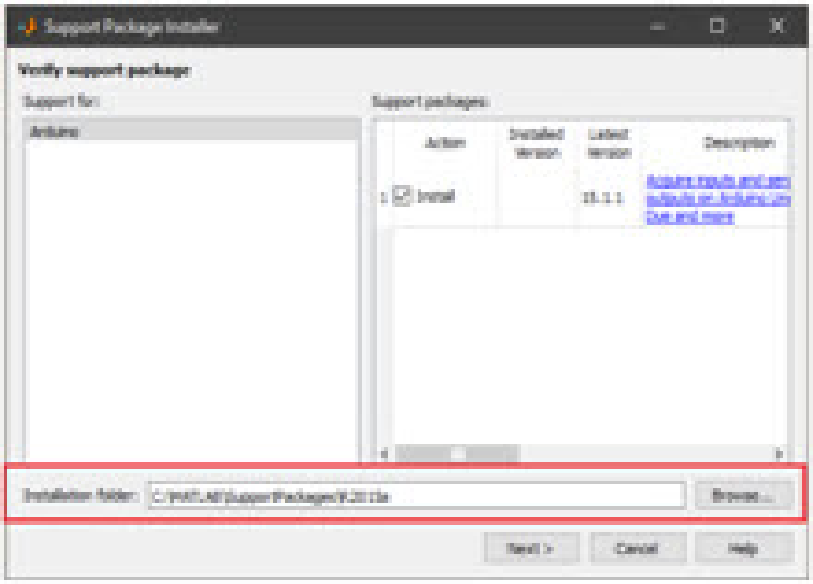

Figura 3: Instalación de Add-on.

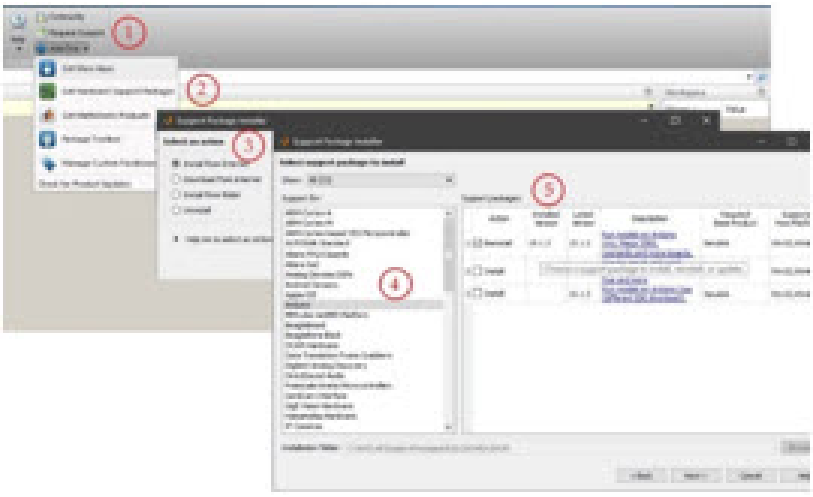

Figura 4: Instalación de Add-on desde Internet

1. Abrir un nuevo documento de Simulink.
2. En el menú Tools.
3. Seleccionar Run on Target Hardware.
4. Seleccionar Options.

En la Figura 5 se muestran los pasos anteriores.

Tras llevar acabo el paso número 4, se desplegará la ventana que se muestra en la Figura 6.

En Target Hardware seleccionar:

1. La tarjeta Arduino con la que se trabajará, para este caso, se esta utilizando un Arduino Mega 2560. Debido a que permite realizar las lecturas en tiempo real.
2. Se recomienda seleccionar el puerto de comunicación de forma manual.

4.Programación de interfaz en Matlab Simulink

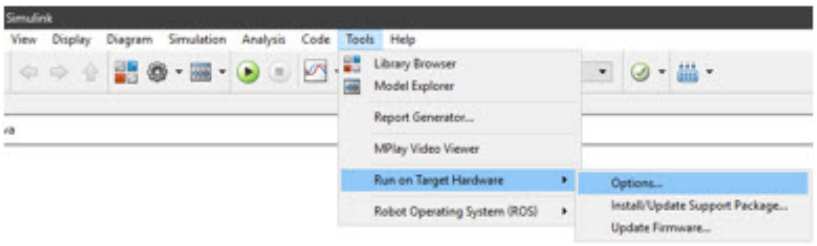

Figura 5: Configuración de Matlab-Simulink para comunicación

Figura 6: Selección del puerto de comunicación.

Hasta ahora se ha llevado a cabo la instalación del Add-on para Arduino, así como realizar la configuración para la comunicación. Aquí se explicará como realizar un programa para recibir y enviar datos mediante Matlab Simulink y Arduino.

En un archivo nuevo de Simulink, en el menú de librerías seleccionar Simulink Support Package for Arduino Hardware, posteriormente elegir Common así se podrán seleccionar los diferentes bloques disponibles para la comunicación con Arduino (Figura 7).

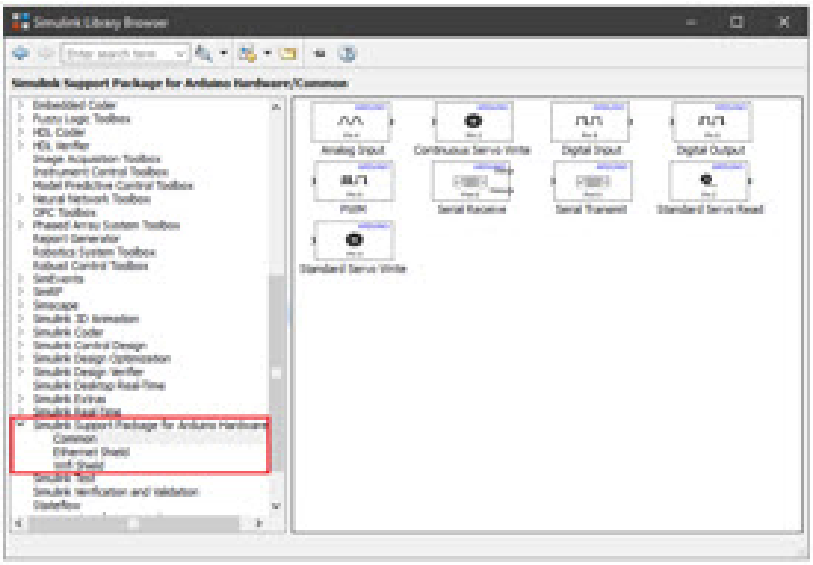

Figura 7: Bloques de Simulnk para Arduino.

4.1. Lectura de tiempo de carga y descarga de un capacitor

Un ejercicio interesante es observar y comprobar el tiempo de carga y descarga de un capacitor. Así, en esta subsección se realiza una comparación de los resultados obtenidos mediante simulación y los de la adquisición de datos.

Antes de presentar el circuito, se llevará a cabo una explicación de los bloques necesarios para lograr la una correcta adquisición de datos.

1. El primer bloque de debemos de utilizar es Anolog Input en este bloque, se selecciona la entrada por la que se recibirá la señal analógica. Entrando a las propiedades del bloque, es posible cambiar el número de entrada.
2. En los bloques propios de Simulink seleccionar Data Type Conversion, este bloque servirá para cambiar el tipo de dato de entrada a bouble (Figura 8a).
3. Seleccionar Discrete Filter, este bloque permitirá capturar una señal con mejor resolución (Figura 8b)
4. Mediante los bloques de multiplicar y dividir se realiza la conversión para obtener las lecturas correctas de voltaje, es decir de 0 V a 5 V.
5. Por último, se coloca el bloque scope para visualizar la señal recibida.

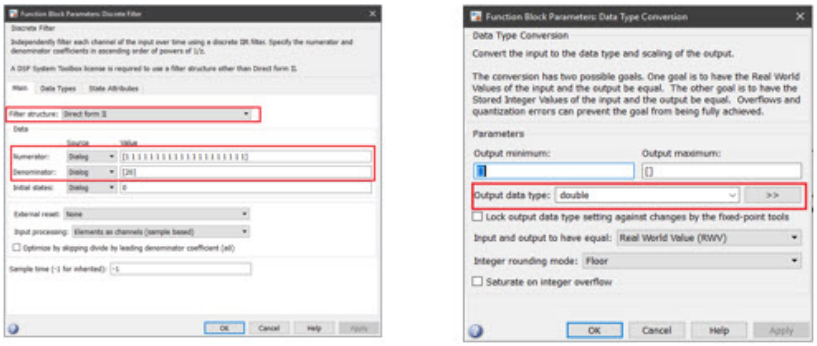

Figura 8: Configuración de bloques: filtro y conversión de datos.

En la Figura 9, se muestra el programa realizado en Matlab-Simulink con los bloques mencionados. Mientras que en la Figura 11, se muestra el circuito eléctrico.

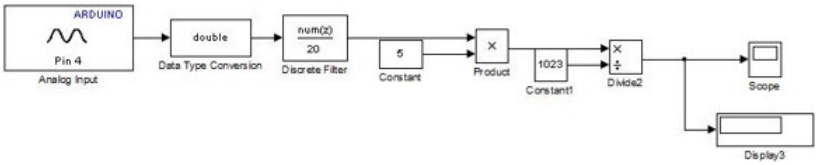

Figura 9: Programa para adquisición de datos analógicos.

Para visualizar en tiempo real los cambios de voltaje en el capacitor, se deben seguir los siguientes pasos.

1. Seleccionar el modo External.
2. Proporcionar el tiempo de ejecución.
3. Cargar el programa en arduino mediante Deploy to Hadware.
4. Ejecutar el programa.

En la Figura 10 se muestran los iconos de los pasos mencionados anteriormente.

4.2. Gráfica de carga y descarga de un capacitor

El circuito eléctrico de la Figura 11 fue realizado en Multisim [4] con los parámetros que se muestran en dicha figura.

Con los valores de Resistencia (R) y Capacitor (C) propuestos se obtiene la siguiente constante de tiempo:

para que un capacitor se cargue o descargue por completo se tiene que:

Tras llevar a cabo la simulación en Multisim se puede comprobar que el tiempo de carga y descarga se lleva a cabo en el tiempo calculado, cabe mencionar que el interruptor se cambió a la posición de descarga a los 135 s, ver Grafica 1.

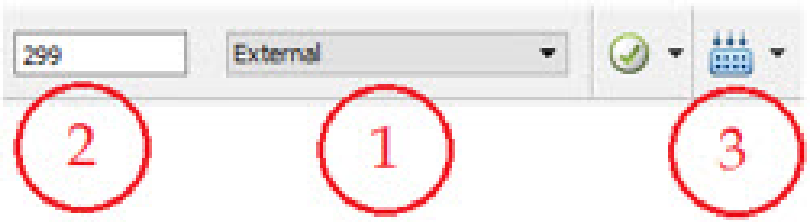

Figura 10: Pasos para cargar programa en Arduino.

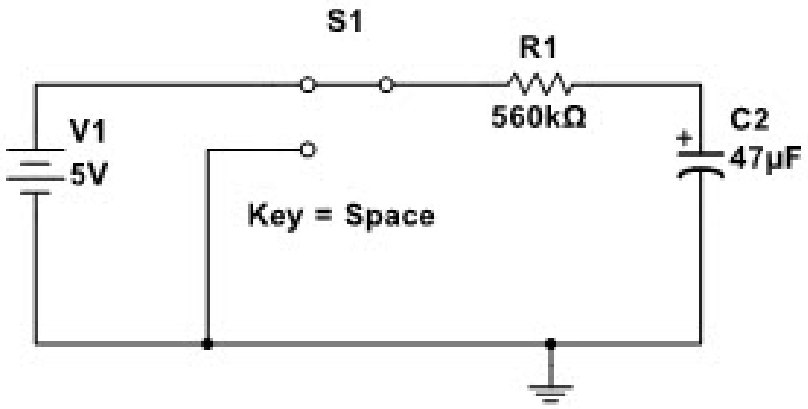

Figura 11: Circuito para cargar y descargar un capacitor.

$$5\tau = 131.6\ s. \tag{2}$$

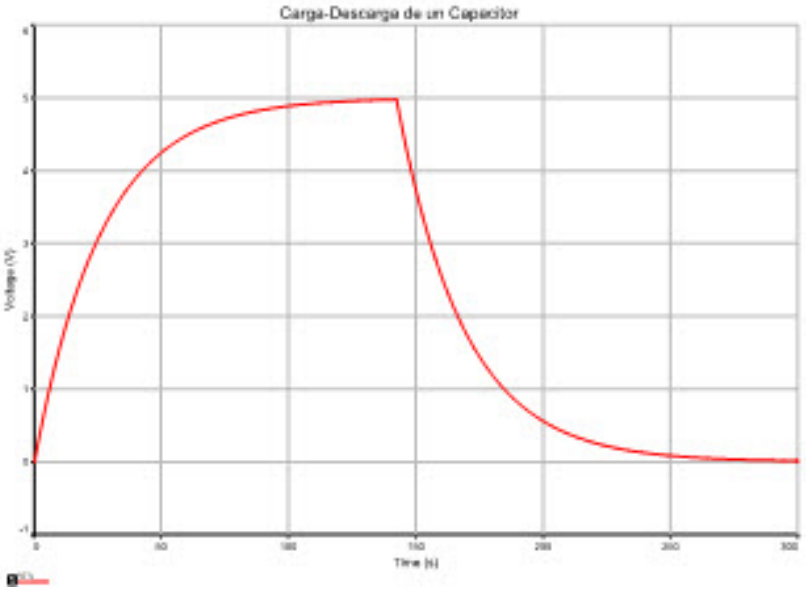

Gráfico 0.1: Resultados de simulación de carga y descarga de un capacitor.

Para llevar a cabo la adquisición de datos se llevó a la práctica el circuito mostrado en la Figura 11. El programa utilizado en Matlab-Simulink es el mostrado en la Figura 9, es necesario conectar en común las tierras de ambos circuitos y la lectura de voltaje se toma del nodo entre la resistencia y el capacitor. En la Grafica 2, se muestra el comportamiento en tiempo real de carga y descarga.

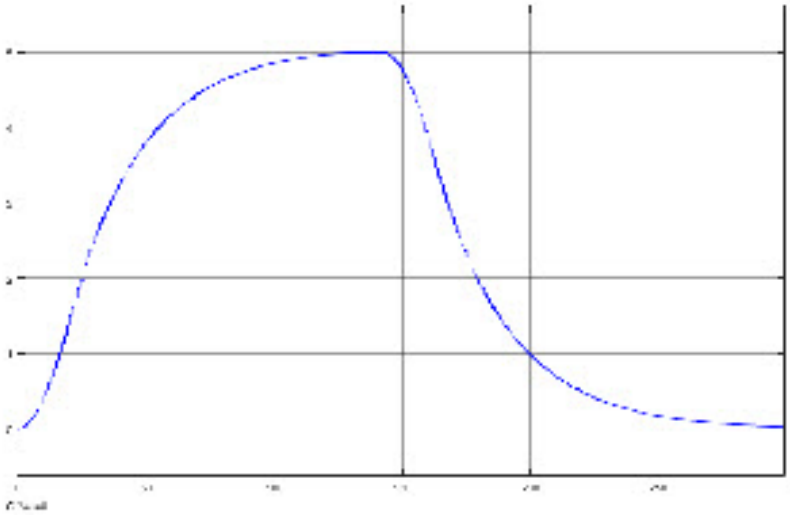

Gráfico 0.2: Resultados de carga y descarga de un capacitor obtenidos mediante Arduino.

Haciendo una comparativa de las gráficas 1 y 2, se puede comprobar que, aunque existen pequeñas diferencias entre ellas, se logra una buena aproximación con los resultados numéricos de la ecuación (2). En https://goo.gl/NFuEYd se han compartido los archivos de simulación y de la adquisición de datos.

5. Conclusiones

En el ejercicio desarrollado en el presente trabajo se demostró que una solución para la adquisición de datos de bajo costo, se puede realizar utilizando la tarjeta Arduino, mediante la comparativa que se ha llevado a cabo de las gráficas de tiempo de carga y descarga del capacitor, se observa que, en general, los resultados son satisfactorios y confiables, con lo cual se puede dar solución a múltiples aplicaciones estudiantiles donde sea necesario conocer y/o capturar datos en tiempo real.

Referencias

1. C. L. Clark (**2005**) *Título del artículo/LabVIEW digital signal processing. /revista/web* U.S: McGraw-Hil.

2. S. F. Barrett (**2013**) *Título del artículo/Arduino Microcontroller Processing for Everyone!, /revista/web* U.S: Morgan & Mlaypool Publishers.

3. Autor (**2019**) *Título del artículo//revista/MathWorks* Recuperado el 05 de marzo de 2019, de https://la.mathworks.com/

4. Autor (**2019**) *Título del artículo/libro/revista/National Instruments* Recuperado el 05 de marzo de 2019, de http://www.ni.com/es-mx.html