

Universidad Autónoma de Nuevo León
Facultad de Ingeniería Mecánica y Eléctrica

Adquisición de datos

Manual de operación del Equipo desarrollado.

M.C. Maestro: Antonio Cayetano Lozano García

Semestre: Ene – Jun 2022

Hora: LMV M3

Grupo: 001

Matrícula	Nombre	Carrera
1902373	Samuel Ernesto Torres Cosío	IMTC
1901410	Julio de Jesús Moreno Sánchez	IMTC
1894698	Christopher Russ Acuña Rodríguez	IMTC
1895481	Alfonso Emiliano Sandoval Juárez	IMTC
1820718	Cesar Alonso Cantú Espinosa	IMTC
1827011	Emilio González Rojas	IMTC

Lugar: Pedro de Alba SN, Niños Héroes, Ciudad Universitaria, San Nicolás de los Garza, N.L.

Fecha: 27/05/2022

Tabla de figuras

Ilustración 1: Carpetas y archivos en repositorio GitHub	4
Ilustración 2: Archivo "myserial.ino"	4
Ilustración 3: Código archivo myserial.ino	4
Ilustración 4: Código sin formato	5
Ilustración 5: Guardar archivo individual desde repositorio	5
Ilustración 6: Archivo con link de carpeta en la nube (programa)	5
Ilustración 7: Link de carpeta	6
Ilustración 8: Archivos de la carpeta en la nube	6
Ilustración 9: Seleccionar archivos	6
Ilustración 10: Descargar archivos	6
Ilustración 11: IDE de Arduino, configuración de placa y puerto	7
Ilustración 12: Archivo myserial.ino	7
Ilustración 13: Sketch subido exitosamente	8
Ilustración 14: Representación inexacta de conexiones Arduino-DAC	8
Ilustración 15: Archivos comprimidos	9
Ilustración 16: Carpetas descomprimidas	9
Ilustración 17: Archivo ejecutable	9
Ilustración 18: Archivo ejecutable	9
Ilustración 19: Menú principal GUI	9
Ilustración 20: Área de trabajo GUI	10
Ilustración 21: Botones de señales y sus configuraciones	11
Ilustración 22: Controles de operación GUI	12
Ilustración 23: Cuadro de error por datos inválidos	12
Ilustración 24: Cuadro de error por datos inválidos	12
Ilustración 25: Cuadro de error por datos inválidos	13
Ilustración 26: Cuadro de error por conexión fallida	13
Ilustración 27: Aspecto de la interfaz tras lectura de datos	14
Ilustración 28: Controles de operación (botones)	14
Ilustración 29: Cuadro de información lectura exitosa	15
Ilustración 30: Guardado de datos en fichero	16
Ilustración 31: Mensaje de guardado exitoso	17
Ilustración 32: Fichero de texto con datos guardados (escalón)	17
Ilustración 33: Fichero de texto con datos guardados (senoide)	18
Ilustración 34: Datos del desarrollador	18

Ficha técnica

Nombre	IO-DAQ
Clasificación	Adquisición y guardado de datos
Datos del desarrollador	Julio Moreno juliomoreno7217@gmail.com
Aplicación/funciones	Usuario final, escuelas/Enviar y medir señales eléctricas
Descripción	IO-DAQ es un software de uso libre que cumple con las necesidades básicas para introducir estudiantes al control y adquisición de datos
Tamaño de la aplicación y versión	31 Mb, versión 2022
Datos de instalación	Ejecutar la instalación a partir de carpeta en la nube desde repositorio de Github.

Tabla 1: Ficha técnica del proyecto

Manual de operación del Equipo desarrollado.

Nota: Antes de comenzar a hacer uso del software es necesario contar con Windows.

Manual de instalación y uso de software IO-DAQS

~Instalación del software

Los archivos ejecutables al igual que el sketch de arduino se encuentran en la nube (Mega), para acceder a ellos basta con primero ingresar a github y dar con el repositorio adecuado.

Formas de acceder al repositorio

- A) Acceder a partir del hipervínculo: (<https://github.com/IODAQ-Experts/IODAQ-Repository.git>)
- B) Buscar el el nombre repositorio u organización en el buscador del sitio web github: (IODAQ-Experts/IO-DAQS-Repository)

De una otra forma se mostrará lo siguiente:

	.vscode	readingdata loop fixed	20 days ago
	__pycache__	prints removed, contact ifno added	21 hours ago
	myserial	All signal s working fine!	23 hours ago
	IO-DAQS.py	Intro tab1 added	18 hours ago
	Program Link.md	Rename Program Link.txt to Program Link.md	10 minutes ago
	README.md	Update README.md	21 hours ago
	WorkingArea.py	prints removed, contact ifno added	21 hours ago
	bibliography.txt	Slope and sine signals development -still on proves	11 days ago
	serial-pilot.py	readingdata loop fixed	20 days ago

Ilustración 1: Carpetas y archivos en repositorio GitHub

Una vez en el repositorio, aparecen los archivos que se pueden descargar. En esta ocasión el sketch de arduino se puede descargar directamente del repositorio o a través de la carpeta compartida en la nube (Mega). Mientras que el archivo ejecutable del programa debe descargarse forzosamente desde la carpeta compartida.

Descargar **directamente del repositorio**:

1.- Hacer click en la carpeta “myserial”:

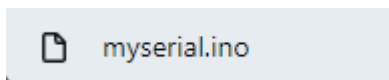






Ilustración 2: Archivo "myserial.ino"

2.- Abrir archivo myserial.ino

3.- Se va a mostrar el código y a la derecha una serie de botones, se selecciona “Raw”:

221 lines (186 sloc) | 8.66 KB

RawBlame





```

1  #define InputPinMeasurement A1
2  #define OutputPinMeasurement A2
3  const int FeedingVoltagePin[]={2,3,4,5,6,7,8,9};
4  void setup() {
5      Serial.begin(115200);
6      Serial.println("Puerto encendido");
7      pinMode(InputPinMeasurement,INPUT);
8      pinMode(OutputPinMeasurement,INPUT);
9      for(int j=0;j<8;j++){
10         pinMode(FeedingVoltagePin[j],OUTPUT);
11     }
12
13 }
14
15 void loop() {
16     DecodeDataChain();
17 }
--

```

Ilustración 3: Código archivo myserial.ino

4.- El código del archivo aparecerá sin formato pero con la extensión original, por lo que para guardarlo, se da click derecho y después en guardar como:

```

#define InputPinMeasurement A1
#define OutputPinMeasurement A2
const int FeedingVoltagePin[]={2,3,4,5,6,7,8,9};
void setup() {
  Serial.begin(115200);
  Serial.println("Puerto encendido");
  pinMode(InputPinMeasurement,INPUT);
  pinMode(OutputPinMeasurement,INPUT);
  for(int j=0;j<8;j++){
    pinMode(FeedingVoltagePin[j],OUTPUT);
  }
}
  
```

Atrás
 Reenviar
 Volver a cargar
 Guardar como...
 Imprimir...
 Enviar...
 Buscar imágenes con Google Lens

Ilustración 4: Código sin formato

5.- Aparecerá una pantalla de dialogo para guardar el archivo, se cambia la extensión a “.ino” si es necesario

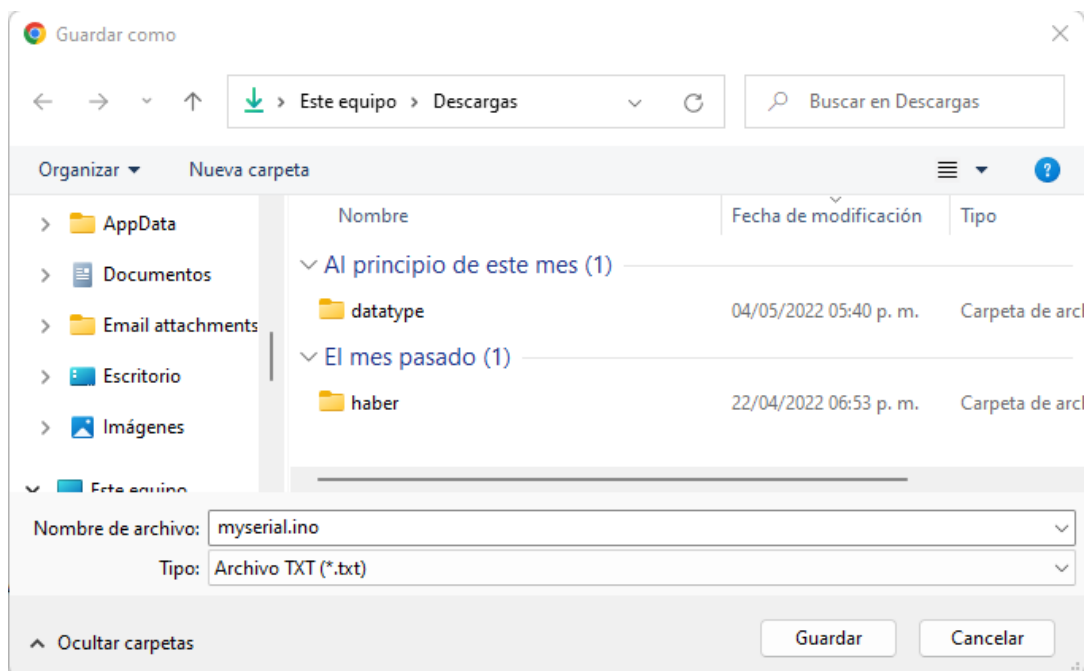


Ilustración 5: Guardar archivo individual desde repositorio

Descargar a través de la **carpeta compartida en la nube (Mega)**:

1.- Se abre el archivo de nombre “Program Link.md”:

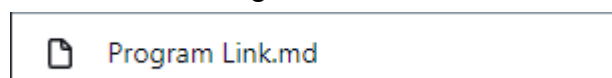


Ilustración 6: Archivo con link de carpeta en la nube (programa)

2.- Se da click el hipervínculo contenido en el mismo:

1 lines (1 sloc) | 85 Bytes

Program download link (mega): <https://mega.nz/folder/z9VAyQjZ#FqmQHNVgay82HhpdN7OvFQ>

Ilustración 7: Link de carpeta

3.- El link redirige a la carpeta en la nube, la cual contiene tres archivos, dos .zip y uno .ino.

DAQ Software


Name	Size
 IO-DAQS(F).zip	31.4 MB
 IO-DAQS(S).zip	30.0 MB
 myserial.ino	9 KB

Ilustración 8: Archivos de la carpeta en la nube

4.- Solo se requiere descargar uno de los archivos .zip, ya que ambos contienen el mismo programa ejecutable, con la diferencia que IO-DAQS(S).zip, tiene un solo archivo dentro, mientras que IO-DAQS(F).zip tiene múltiples archivos. Así que en uno se tiene acceso a los archivos internos del programa, y en el otro no. Los archivos se pueden descargar todos a la vez en un archivo zip que los contenga a todos (boton "Download as ZIP"):

DAQ Software Download as ZIP

Name	Size	Type
 IO-DAQS(F).zip	31.4 MB	ZIP Compressed
 IO-DAQS(S).zip	30.0 MB	ZIP Compressed
 myserial.ino	9 KB	INO File

Ilustración 9: Seleccionar archivos

o descargar cada archivo individualmente al seleccionarlo y dar click derecho:

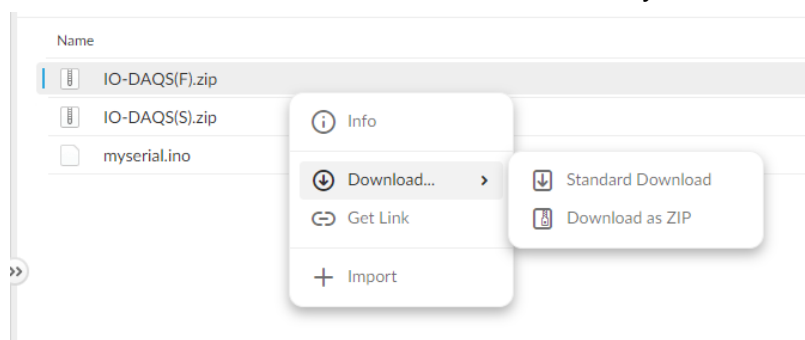


Ilustración 10: Descargar archivos

Una vez se descargan los archivos necesarios de una u otra forma; lo primero que se hace es cargar el código del archivo “myserial.ino” en la placa a través del IDE de Arduino; de lo contrario el microcontrolador no realizará las funciones requeridas. Para esto se aclara que la placa usada es Arduino UNO R3, por lo que es incierto el resultado de usar el mismo sketch en otras placas (debido al cambio en la distribución de pines); aun así se puede modificar de ser necesario. Para esto se debe seleccionar la placa a utilizar y seleccionar el puerto serial adecuado (menú “herramientas”):

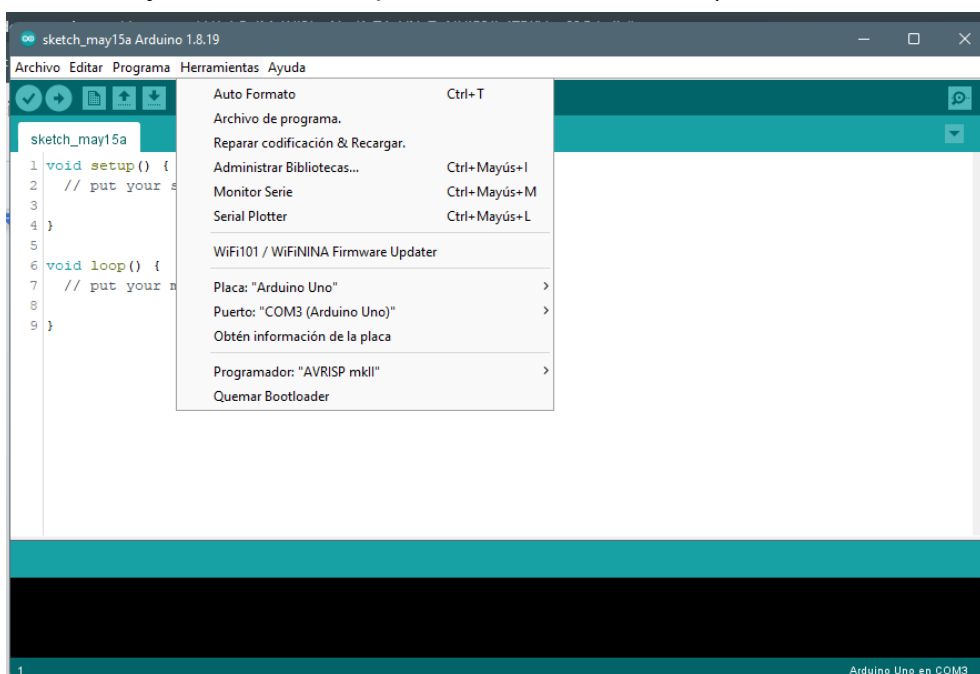


Ilustración 11: IDE de Arduino, configuración de placa y puerto

Con el IDE abierto y la placa con el puerto seleccionados sigue abrir el documento .ino:

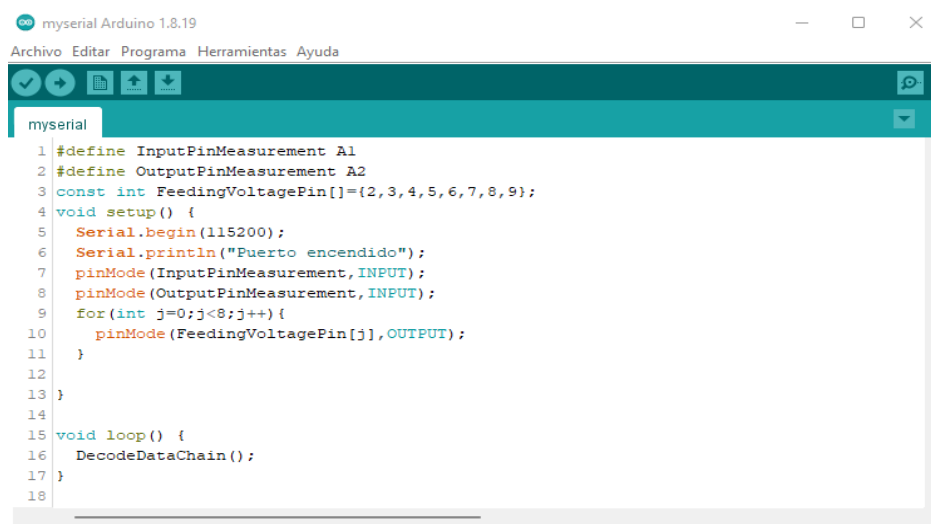


Ilustración 12: Archivo myserial.ino

Una vez abierto el archivo se selecciona el botón con el símbolo de flecha para subir el programa a la placa; aparecerá el siguiente mensaje si el código se cargo adecuadamente:

```
Subido
El Sketch usa 9688 bytes (30%) del espacio de almacenamiento de programa. El máximo es 32256 bytes.
Las variables Globales usan 284 bytes (13%) de la memoria dinámica, dejando 1764 bytes para las variables locales. El máximo es 2048 bytes.
```

Ilustración 13: Sketch subido exitosamente

Hasta este punto el microcontrolador ya está listo para operar. Cabe aclarar que para que el arduino verdaderamente mida efectivamente, las señales emitidas forzosamente son salidas digitales así que se requiere un convertidor digital analógico para obtener la señal deseada, y del mismo modo conectar los pines apropiados. Estos pines tanto de lectura como alimentación pueden ser modificados a conveniencia dentro del sketch sin invertirlos; la configuración predeterminada es tal que al ejecutarse no presenta problemas (diagrama de conexiones). Y no olvidar que al tratarse de tecnología TTL, solo se puede medir de 0 a 5 Volts.

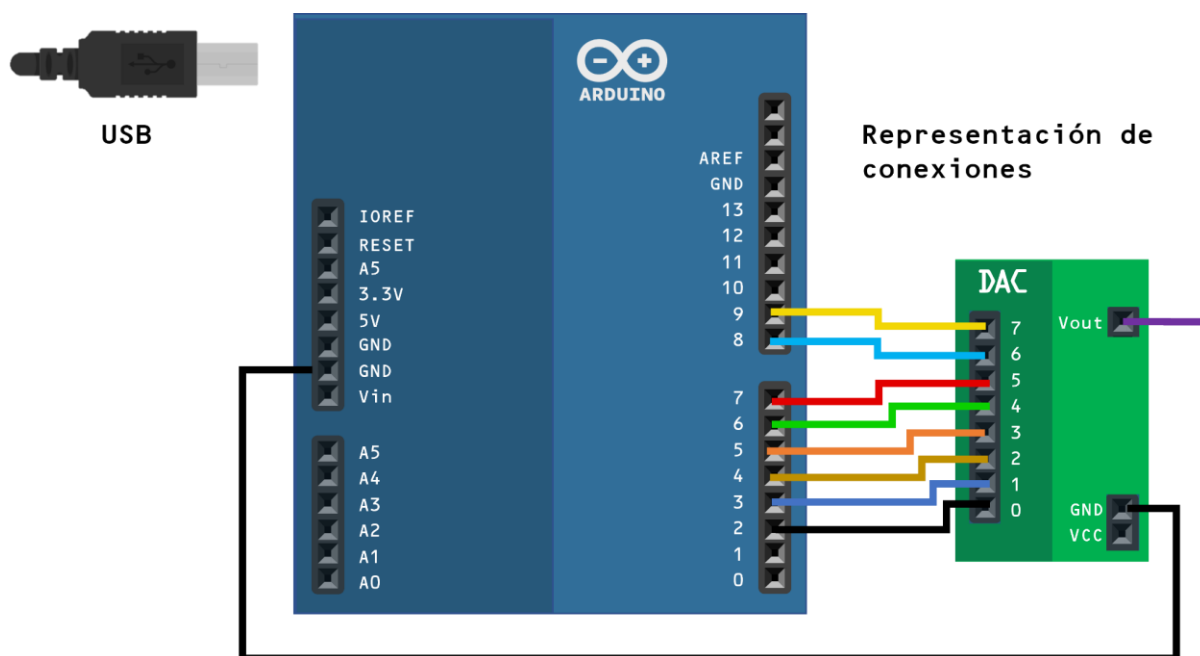


Ilustración 14: Representación inexacta de conexiones Arduino-DAC

Lo que sigue es abrir el programa IO-DAQS(F).exe o IO-DAQS(S).exe según sea el caso ("F" de "full", es decir una carpeta con todos los archivos y "S" de "single", es decir un solo archivo). Para esto basta con descomprimir el archivo IO-DAQS(F).zip:



IO-DAQS(F).zip

IO-DAQS(S).zip

Ilustración 15: Archivos comprimidos

Dentro se encontrará una carpeta con mismo nombre:

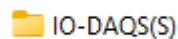
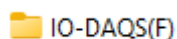


Ilustración 16: Carpetas descomprimidas

En el caso de IO-DAQS(S), dentro se encuentra un solo archivo que es el ejecutable:

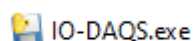


Ilustración 17: Archivo ejecutable

Mientras que en el caso de IO-DAQS(F), se encuentra una cantidad superior de archivos, entre ellos el ejecutable:

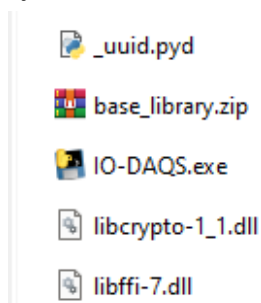


Ilustración 18: Archivo ejecutable

Una vez se ejecute el archivo ejecutable, se mostrará la interfaz como tal:

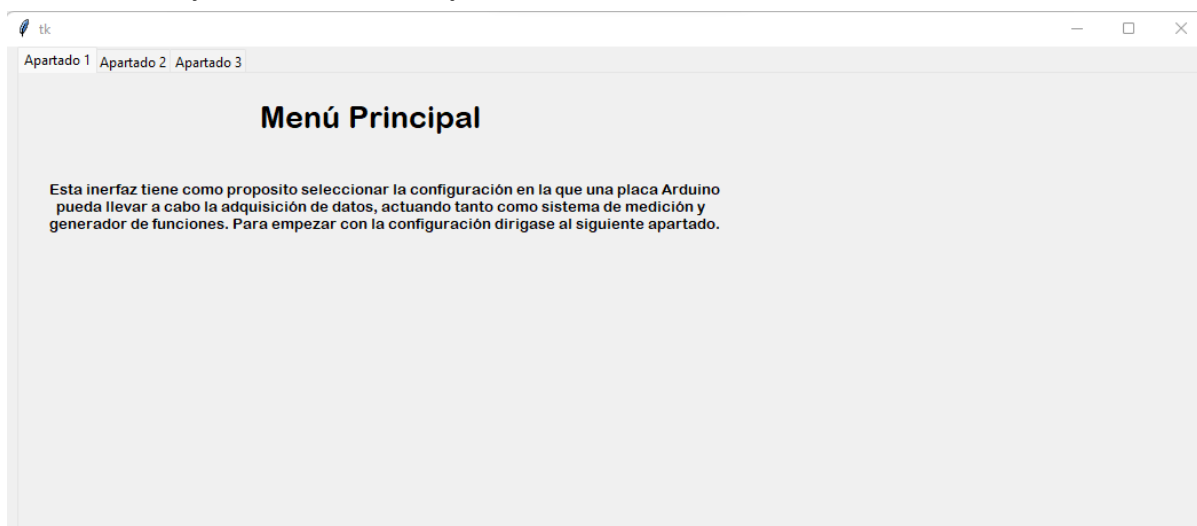


Ilustración 19: Menú principal GUI

Aparecen tres pestañas en la zona superior de la ventana, la primera solo contiene una breve descripción de la función de la interfaz, el segundo apartado es donde se establecen las configuraciones con las que trabajará el Arduino y el último apartado tiene información tanto del repositorio como del desarrollador software detrás del programa.

Ahora bien para poder empezar a manipular la interfaz se selecciona el “Apartado 2”:

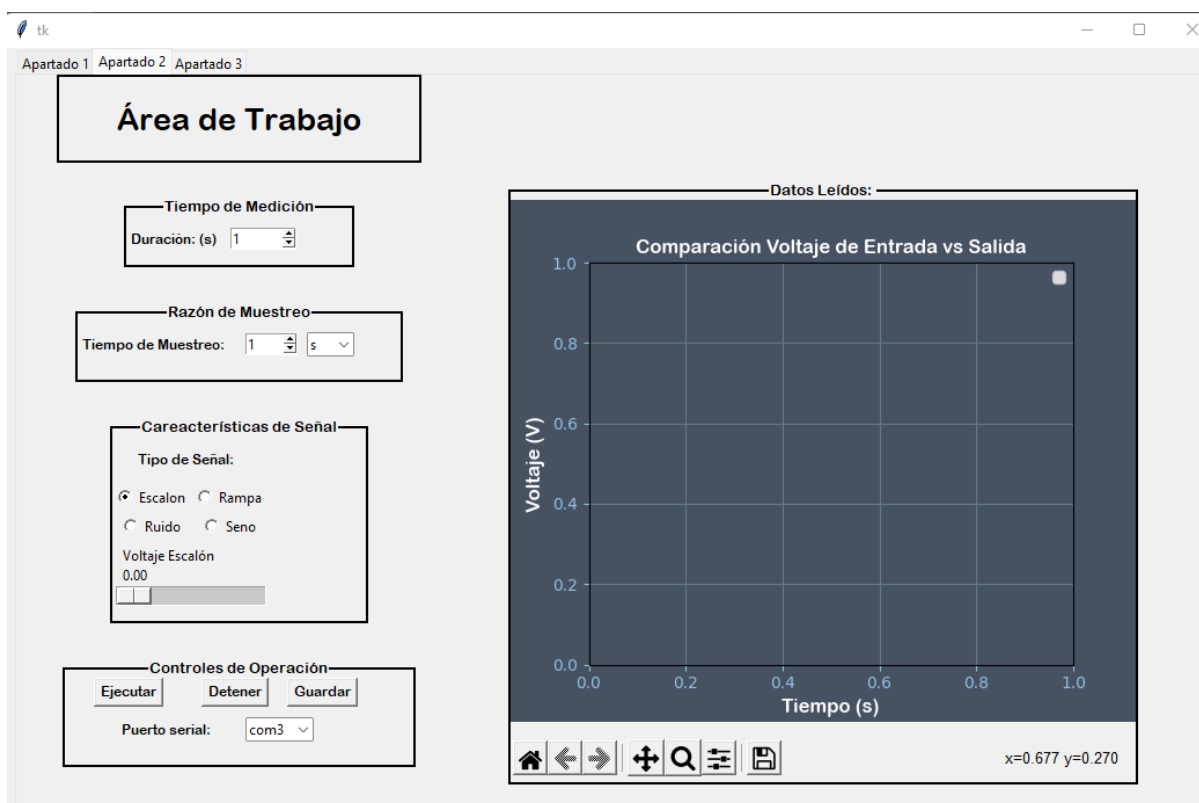


Ilustración 20: Área de trabajo GUI

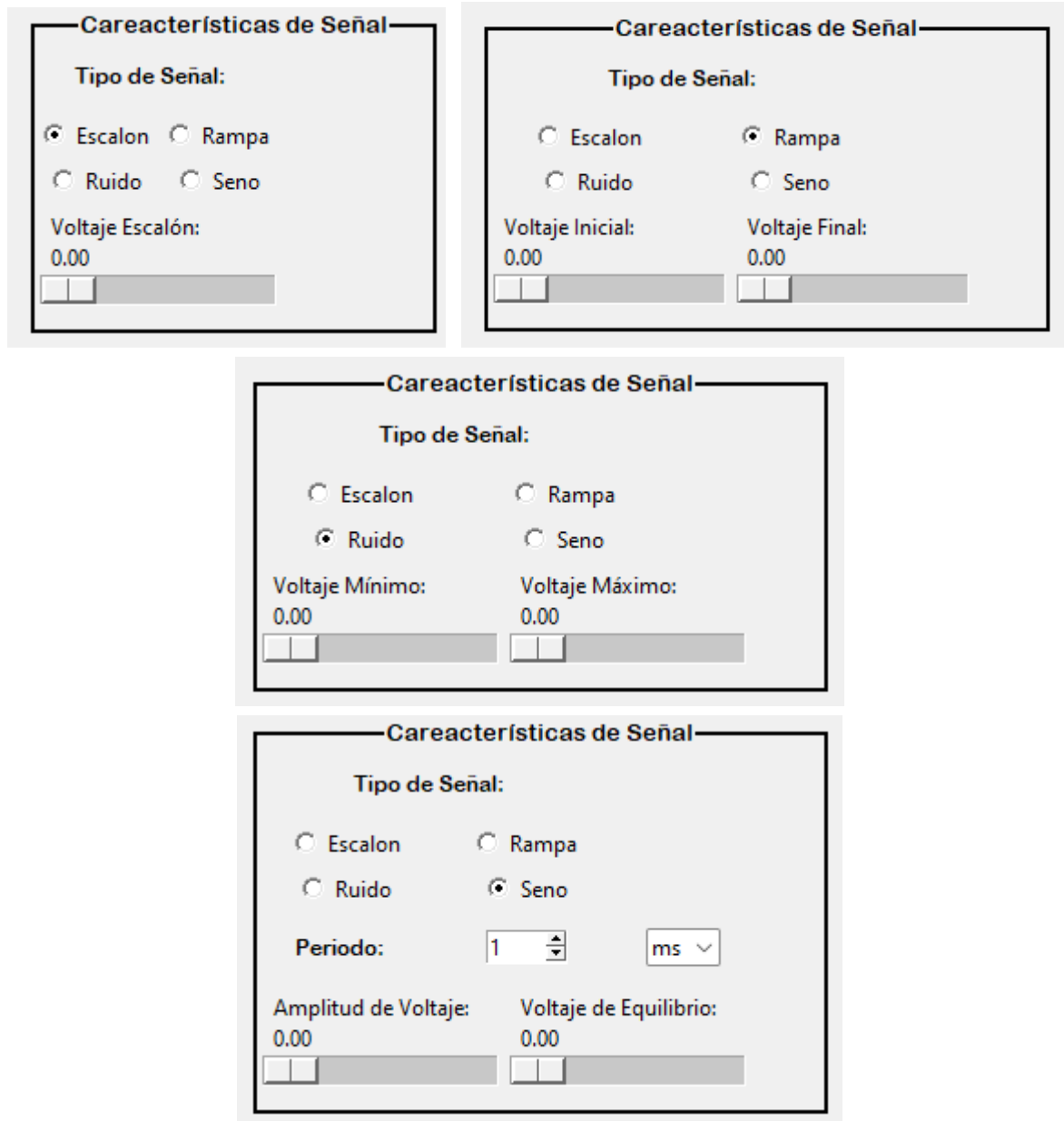
Antes de siquiera ejecutar las mediciones es necesario configurar adecuadamente los parámetros iniciales que son los siguientes:

Tiempo de Medición: Indica el tiempo en segundos que el arduino estará realizando mediciones.

Tiempo de Muestreo: Indica el tiempo que acontece entre una medición y la siguiente, las unidades pueden expresarse en segundos, milisegundos, microsegundos o nanosegundos. El tiempo de muestreo mínimo alcanzado actualmente es de 1.8 milisegundos.

Tipo de señal: Patrón en el que la señal variará a través del tiempo, hay cuatro posibles opciones (escalón, rampa, ruido y senoide). Dependiendo de la señal elegida se desplegarán los parámetros propios que esa señal necesita para poder emitirse.

Voltajes para cada señal:



The figure displays four instances of the 'Características de Señal' (Signal Characteristics) dialog box, each showing a different signal type selected:

- Top Left:** 'Tipo de Señal:' with 'Escalon' (Step) selected. It shows 'Voltaje Escalón:' (Step Voltage) set to 0.00.
- Top Right:** 'Tipo de Señal:' with 'Rampa' (Ramp) selected. It shows 'Voltaje Inicial:' (Initial Voltage) and 'Voltaje Final:' (Final Voltage) both set to 0.00.
- Middle:** 'Tipo de Señal:' with 'Ruido' (Noise) selected. It shows 'Voltaje Mínimo:' (Minimum Voltage) and 'Voltaje Máximo:' (Maximum Voltage) both set to 0.00.
- Bottom:** 'Tipo de Señal:' with 'Seno' (Sine) selected. It shows 'Periodo:' (Period) set to 1 ms, 'Amplitud de Voltaje:' (Voltage Amplitude) set to 0.00, and 'Voltaje de Equilibrio:' (Balance Voltage) set to 0.00.

Ilustración 21: Botones de señales y sus configuraciones

Finalmente el puerto a utilizar, el cual es configurable, hay algunas opciones y si no se encuentra, basta con modificar el número de puerto:

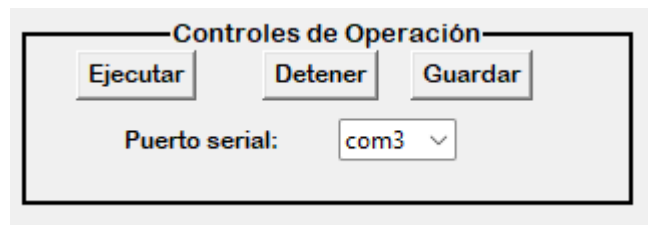


Ilustración 22: Controles de operación GUI

Además si se selecciona la señal senoidal, se agrega un parámetro más el cual es el periodo de la señal. Este periodo se puede expresar en unidades de forma similar al tiempo de muestreo. Cabe mencionar que tras una serie de pruebas el Periodo mínimo alcanzado fue de 5 milisegundos.

Algunas condiciones acerca de los parámetros incluyen:

- El tiempo de medición debe ser mayor que el tiempo de muestreo puesto, que de ser así, no se haría ninguna medición, y no tendría mucho sentido.

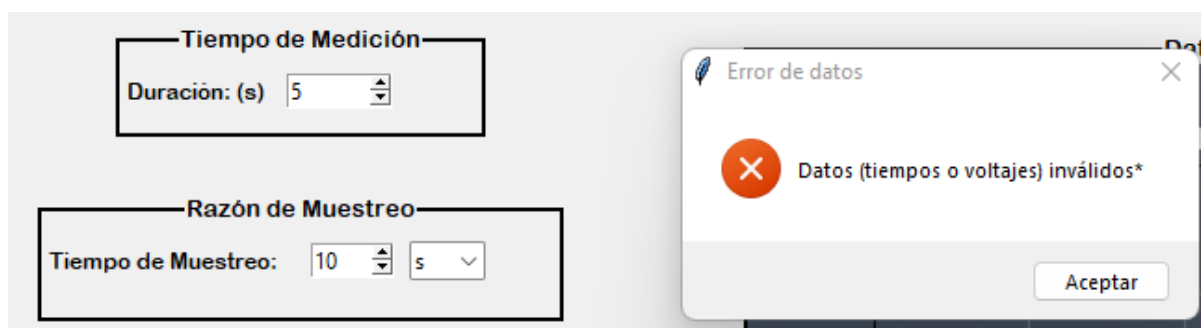


Ilustración 23: Cuadro de error por datos inválidos

- Los tiempos de medición, muestreo y periodo deben ser mayor a 0 V o que en su defecto los valores escrito en la casilla no sean caracteres alfanuméricos (texto).

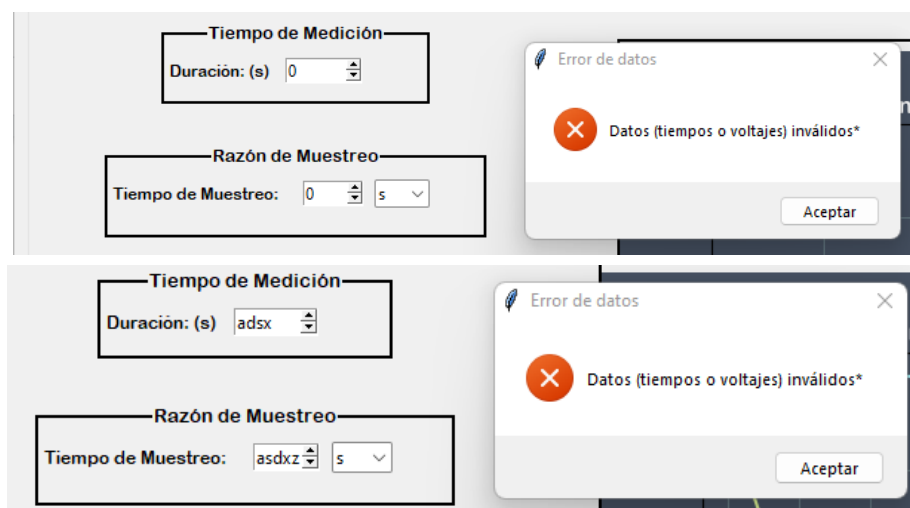


Ilustración 24: Cuadro de error por datos inválidos

- Si la señal es del tipo ruido y resulta que el voltaje máximo es en realidad menor que el mínimo ($\text{max} < \text{min}$), marca error, al igual que en los casos anteriores, pero al final se interpreta que los valores debieron estar invertidos, por lo que al arduino se envían los valores intercambiados a modo que se cumpla la condición ($\text{max} > \text{min}$).

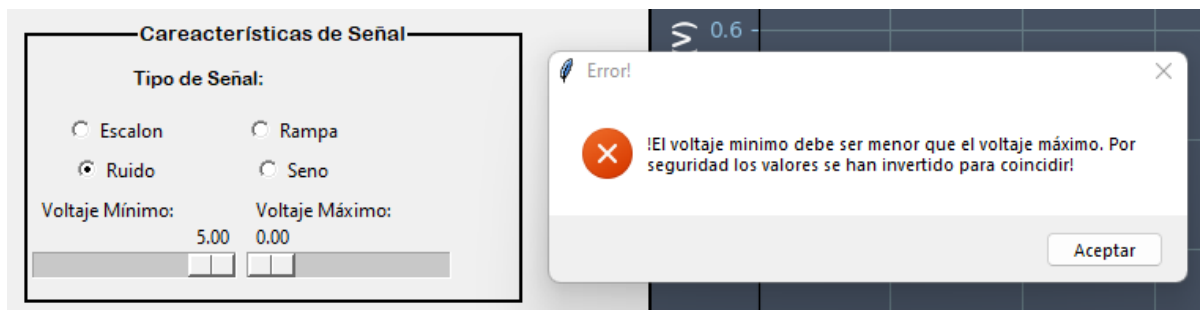


Ilustración 25: Cuadro de error por datos inválidos

- Además si se intenta ejecutar (enviar estos datos para iniciar las mediciones) y el arduino no está conectado, también indica un error de conexión. Que de hecho incluso si estuviera conectado pero los eventos ocurren, la conexión no se establece con el arduino.

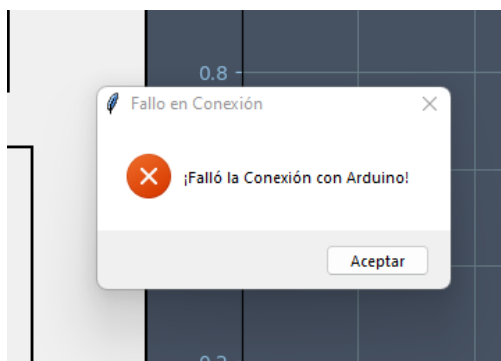


Ilustración 26: Cuadro de error por conexión fallida

- La señal seno puede que se configure a modo que verticalmente los valores sobrepasan los 5 Volts o que sean menores a los 0 V, en cuyos casos, ya que no se pueden enviar esos estados al DAC, ocurre un recorte, cuando $0 > \text{Voltaje} > 5$.

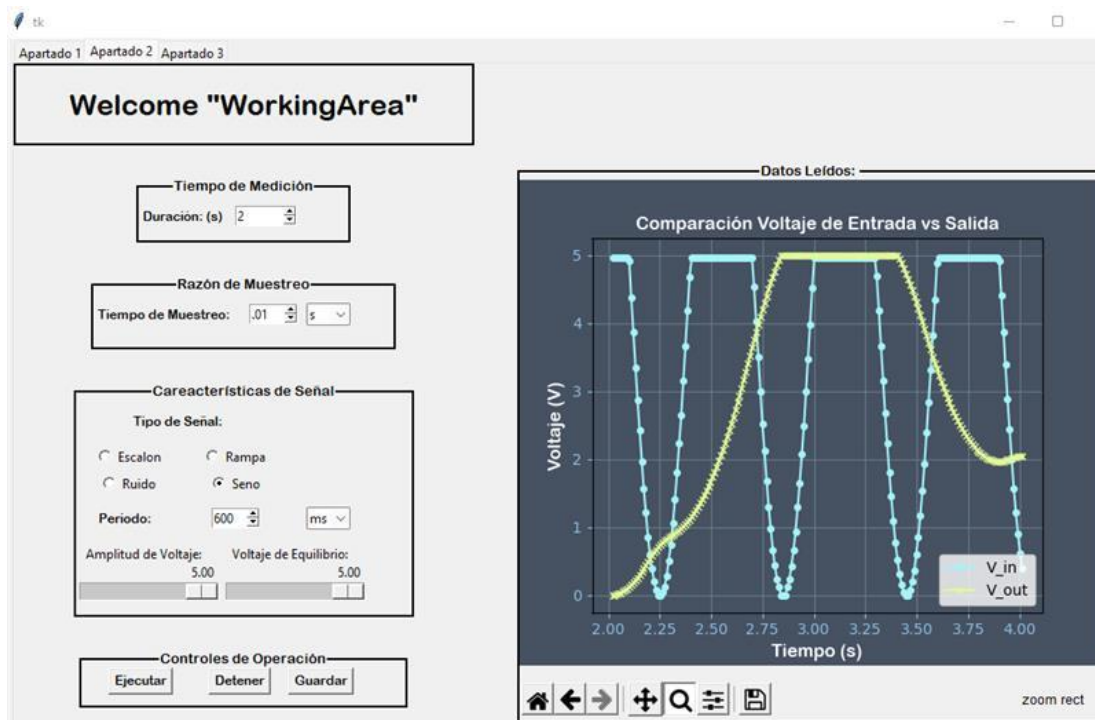


Ilustración 27: Aspecto de la interfaz tras lectura de datos

Evidentemente se tienen tres botones cuyas funciones son intuitivas, pues el primero es la puesta en marcha, y analiza los parámetros antes de hacer funcionar al arduino. Si los parámetros son válidos, el arduino procederá a enviar la señal elegida al igual que tomará muestras; una vez termina aparece un mensaje que indica que las mediciones y la gráfica de las mismas fue exitosa:

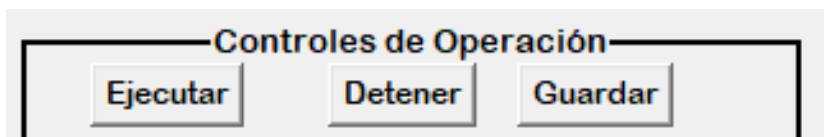


Ilustración 28: Controles de operación (botones)

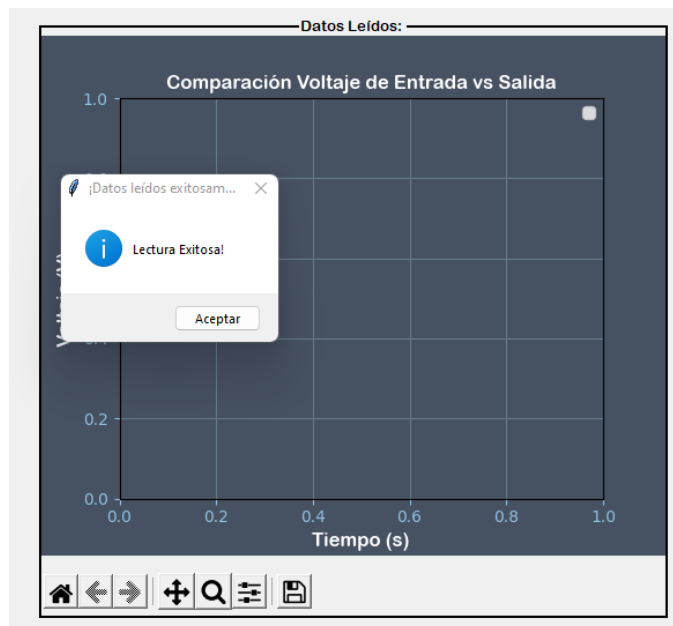


Ilustración 29: Cuadro de información lectura exitosa

Tras dar click en aceptar, los datos leídos se mostrarán en el plano cartesiano:

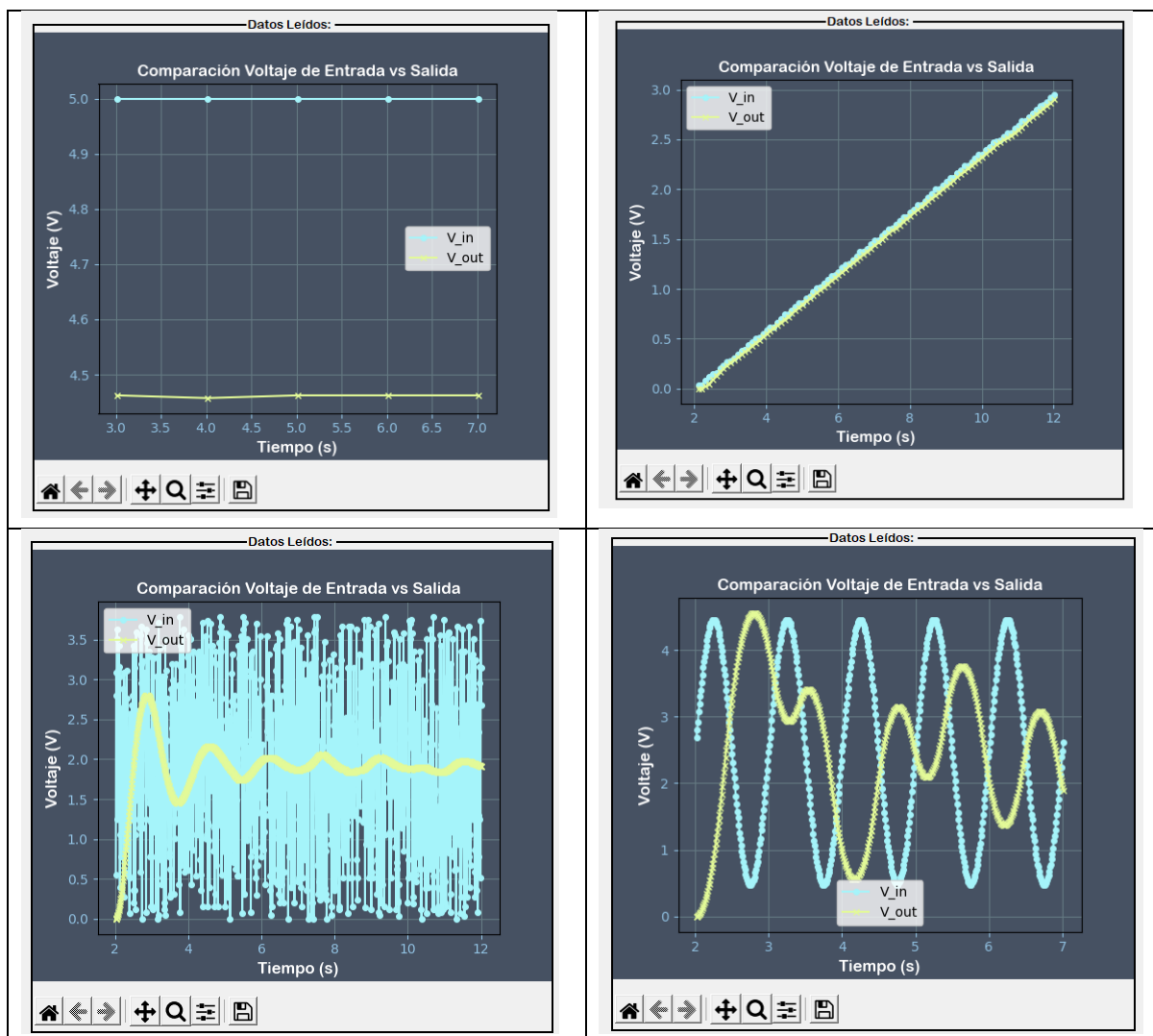


Tabla 2: Señales de entrada

Se muestra el título de la gráfica, las etiquetas de los ejes al igual que la leyenda de las gráficas con su color característico. Al igual que justo debajo aparece una barra de herramientas que permite modificar la gráfica a modo de resaltar características de importancia como: desplazamiento de la gráfica, cambio de tamaños, restaurar la vista predeterminada, hacer zoom en una zona o bien guardar la gráfica como imagen.

El botón de “Detener” en sí cierra el puerto sería a modo que no ocurra un flujo de datos, terminando las mediciones.

Finalmente el botón “Guardar” permite que las mediciones leídas a través del puerto serial, se guardan en archivos de distintas extensiones como: .txt, .cvs y .xls. Todos pertenecientes a la categoría de archivos tipo ASCII. Para que esta función ocurra, se debe tener por lo menos una medición obtenida; de lo contrario no se llevará a cabo el proceso. Además permite guardar el archivo en el lugar y con el nombre que se quiera:

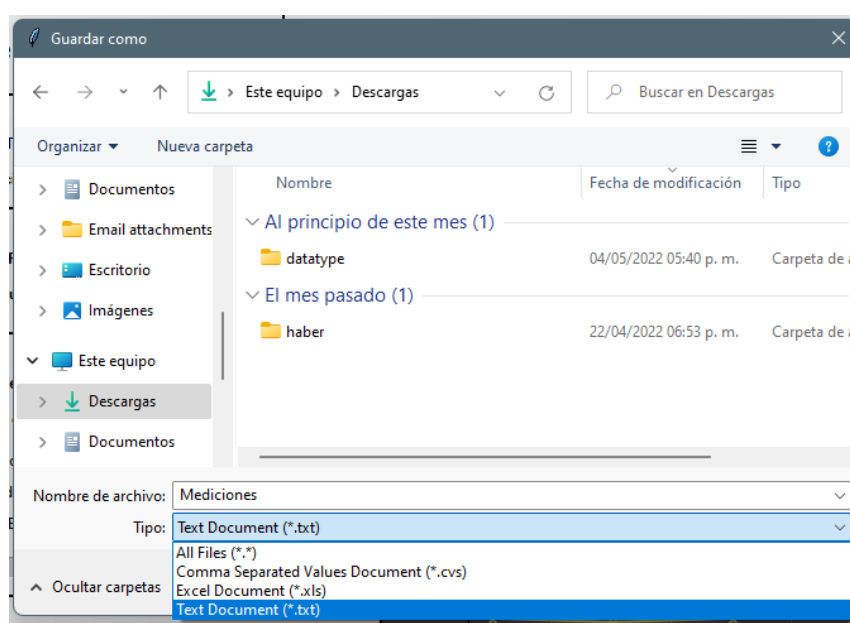


Ilustración 30: Guardado de datos en fichero

Justo después de seleccionar la extensión y seleccionar en guardar aparecerá una ventana emergente indicando el guardado correcto de los datos:

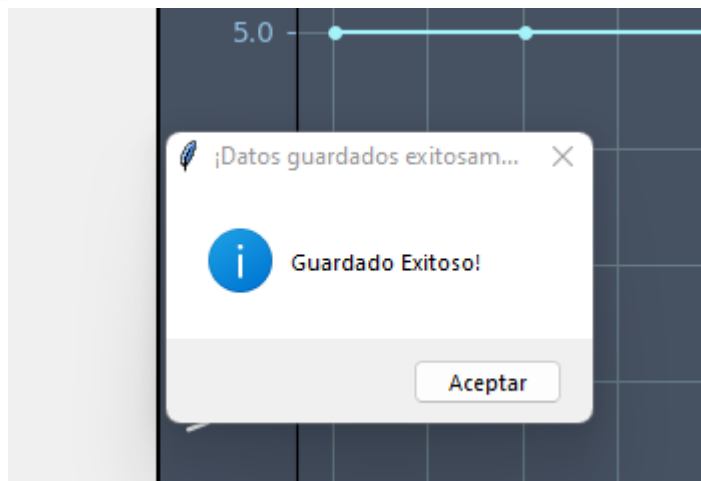


Ilustración 31: Mensaje de guardado exitoso

El resultado de este guardado se puede apreciar:

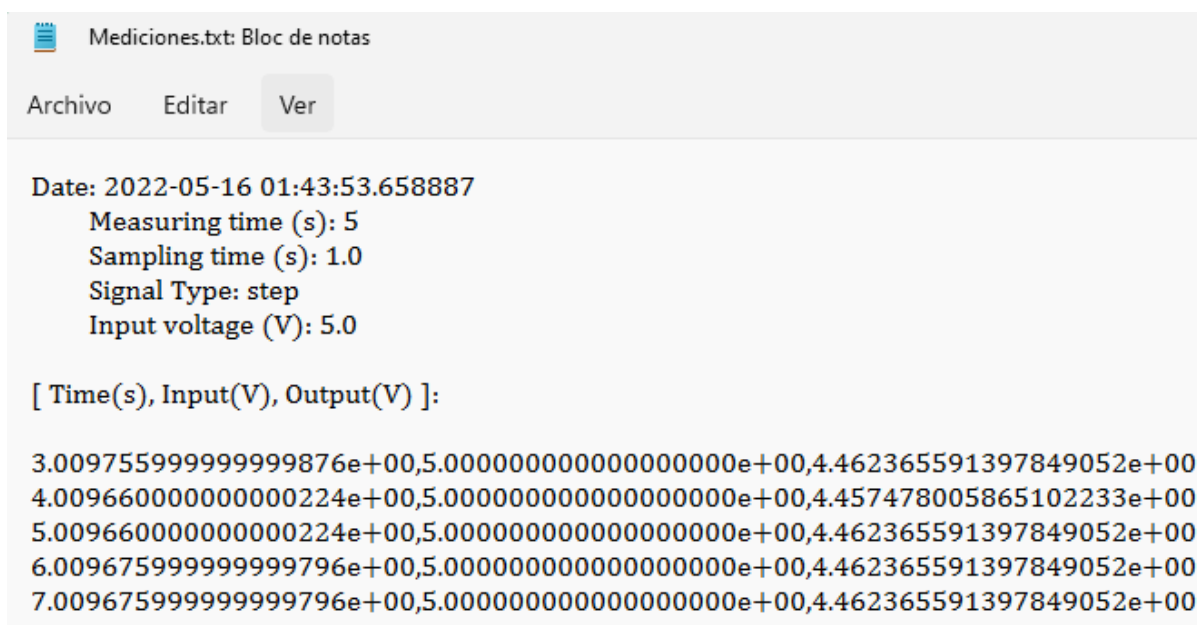


Ilustración 32: Fichero de texto con datos guardados (escalón)

A como se observa, al principio se muestran datos referentes a las condiciones iniciales de la prueba y después los datos leídos y separados por comas, haciendo alusión al tiempo, voltaje de entrada y el voltaje de salida. Las condiciones iniciales se mostrarán dependiendo también de la señal, pues en una los parámetros son más o menos, como el periodo en la señal seno. Por ejemplo:

Date: 2022-05-16 01:48:16.611837

Measuring time (s): 10

Sampling time (s): 0.5

Signal Type: sine

Amplitude voltage (V): 5.0

Equilibrium voltage (V): 0.0

Period (s): 1.0

[Time(s), Input(V), Output(V)]:

```
2.509879999999999889e+00,5.0000000000000000e+00,0.0000000000000000e+00
3.009959999999999969e+00,5.0000000000000000e+00,0.0000000000000000e+00
3.5101360000000000145e+00,5.0000000000000000e+00,0.0000000000000000e+00
4.0103200000000000107e+00,5.0000000000000000e+00,0.0000000000000000e+00
4.510407999999999973e+00,5.0000000000000000e+00,0.0000000000000000e+00
5.0105199999999999641e+00,5.0000000000000000e+00,0.0000000000000000e+00
5.5106359999999999868e+00,5.0000000000000000e+00,0.0000000000000000e+00
```

Ilustración 33: Fichero de texto con datos guardados (senoide)

El último apartado incluye información del desarrollador del programa:



Ilustración 34: Datos del desarrollador