UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS

Instituto de Investigación en Ciencias Básicas y Aplicadas

Instituto de Investigación en Ciencias Básicas y Aplicadas

Centro de Investigación en Ingeniería y Ciencias Aplicadas

TÍTULO DE LA TESIS

DESARROLLO DE INTERFAZ GRÁFICA DE SOFTWARE LIBRE PARA REALIZAR ANÁLISIS DE RUIDO ELECTROQUÍMICO

T e s i s

Para obtener el título de

Licenciado en Tecnología con Área Terminal en Física Aplicada

Presenta

Jorge Misael Angeles Carlos

Director de Tesis

Dra. Arianna Parrales Bahena

Cuernavaca, Morelos, 7 de Noviembre de 2022

AGRADECIMIENTOS

A mis asesores de tesis, la Dra. Arianna Parrales Bahena y al Dr. Roy López Sesenes por la confianza, el apoyo y el tiempo dedicado durante todo el proyecto. Mi infinita gratitud por su paciencia, préstamo de equipo, orientación y guía para la culminación del presente trabajo.

De igual manera extiendo mi agradecimiento a la Mtra. Merle Lisbet García Estrada, por su apoyo en la gestión de los trámites administrativos a lo largo de la licenciatura. También quiero expresar mi reconocimiento a su paciencia y trabajo.

RESUMEN

El presente trabajo se centra en el desarrollo de una Interfaz Gráfica de Usuario de fácil manejo que permita la automatización para el cálculo y análisis de los datos de mediciones de Ruido Electroquímico relacionados con el estudio del proceso de corrosión.

Al evaluar los diferentes métodos de análisis de Ruido Electroquímico se optó por el método estadístico debido a que ofrece parámetros de fácil interpretación e implementación en un algoritmo computacional.

Para el diseño de la interfaz se empleó *App Designer Toolbox* del entorno de *Matlab*® debido a la versatilidad que ofrece para trabajar tanto con elementos visuales como con líneas de código. Además, tiene la posibilidad de compartir la aplicación como programa ejecutable.

El resultado del trabajo es una Interfaz Gráfica que permite con pocos pasos y requisitos la visualización, tratamiento, procesamiento y guardado de datos provenientes de Ruido Electroquímico.

SUMMARY

The present work focused on developing an easy-to-use Graphical User Interface that allowed automation for calculating and analyzing Electrochemical Noise measurement data related to the study of corrosion processes.

When evaluating the different methods of Electrochemical Noise analysis, the statistical method was chosen because it offers parameters that are easy to interpret and implement in a computational algorithm.

For the design of the interface, the App Designer Toolbox of the Matlab® environment was used due to the versatility it offers to work with both visual elements and lines of code. In addition, it offers the possibility to share the application as an executable program.

The result of the work was a Graphic Interface that allows the visualization, treatment, processing, and saving of data from Electrochemical Noise with few steps and requirements.

CONTENIDO

LISTA DE FIGURAS

LISTA DE TABLAS

NOMENCLATURA

Símbolo	Descripción	Unidad
b	Ordenada al origen	Adimensional (-)
IL	Índice de localización	Adimensional (-)
k	Curtosis	Adimensional (-)
m	Valor de la pendiente	Adimensional (-)
n	Número total de datos	Adimensional (-)
nt	Valor sin tendencia	Adimensional (-)
R_n	Resistencia de polarización al ruido	Ohm (Ω)
r	Coeficiente de correlación	Adimensional (-)
r^2	Coeficiente de determinación	Adimensional (-)
s_k	Sesgo	Adimensional (-)
y	Regresión lineal	Adimensional (-)
Variables		
X	Primer valor de entrada	-
Y	Segundo valor de entrada	-
Letras Griegas		
μ	Media estadística	Adimensional (-)
σ	Desviación estándar	Adimensional (-)
Subíndices		
i	Corriente	Ampere (A)
j	Dato contador o índice	Adimensional (-)
v	Voltaje	Volt (V)

CAPÍTULO I

1. INTRODUCCIÓN

Este capítulo tiene por objeto presentar en forma breve el fenómeno de la corrosión y la importancia de su estudio. De igual manera se expone al lector la motivación, desarrollo y aplicación del presente trabajo.

1.1. Antecedentes

Desde el origen de la humanidad, los metales han sido parte fundamental para su evolución y desarrollo tecnológico. Inicialmente, los metales fáciles de encontrar en un estado puro fueron los más utilizados; sin embargo, con el paso del tiempo y con la habilidad del ser humano para transformarlos, fue posible usarlos en combinaciones llamadas aleaciones, las cuales proporcionaban nuevas propiedades y características a las herramientas e instrumentos creados. La importancia de los metales en el desarrollo de la civilización ha sido tan notable que la antropología encuentra adecuado dividir la historia en etapas relacionadas con el material con el que fueron desarrolladas sus herramientas, así es posible distinguir a la Edad de Bronce, la Edad de Hierro y más recientemente a la Edad del Silicio (Galvele, 2011).

Los metales más destacados por sus diversas aplicaciones son el aluminio (Al), el cobre (Cu), el zinc (Zn) y el hierro (Fe), esto es debido en gran manera al tiempo que la humanidad ha logrado servirse de ellos, puesto que son elementos de gran resistencia, considerable abundancia (por lo tanto, de bajo costo) y de relativa facilidad de extracción. En su mayoría encuentran su uso principal en la construcción y en la fabricación de herramientas, pero, dada su gran versatilidad ha sido posible incorporarlos a cada ámbito de la vida diaria desde la agricultura hasta el sector salud.

A pesar de su gran utilidad en la vida diaria y sus múltiples usos, los metales no están exentos de sufrir desgaste, una consecuencia de su uso y principalmente de la exposición al medio ambiente. Este proceso de deterioro es denominado corrosión y es inherente al material,

además que puede haber casos en los cuales no existan cambios visibles en éste y sin embargo, sufrir fallas inesperadas consecuencia de su estructura interna (Fajardo González & Ruballo García, 2022). El daño causado por la corrosión en los metales provoca grandes pérdidas económicas, eleva los costos indirectos al requerir de mantenimiento y trabajo de supervisión para su detección temprana.

La corrosión de los metales comúnmente usados por la humanidad tiene su origen en la propia extracción, debido a que el estado natural de los elementos metálicos es en forma de óxido, sulfuro, cloruro, sulfato o carbonato, sin embargo, para poder ser empleados por las personas hace necesario sean tratados para obtenerlos en estado puro. Los métodos de purificación del metal requieren le sean suministradas altas concentraciones de energía, induciendo al elemento a niveles termodinámicamente inestables y con alta tendencia a volver a su estado base o forma natural (Javaherdashti, 2008). Lo anterior implica que al liberar al metal de impurezas se les lleva en contra de una reacción espontánea y normal en la naturaleza, que es la combinación del metal.

En su mayoría, los procesos de corrosión para la estabilidad del metal involucran reacciones de óxido-reducción (reacciones electroquímicas) en las cuales se hacen necesarios tres constituyentes: (1) electrodos (cátodo y un ánodo), (2) un medio conductor denominado electrolito, y, (3) una conexión eléctrica o puente entre los electrodos. Combinados los tres elementos se forma un símil de lo denominado como celda electroquímica, cuyo ejemplo más común es el de una pila o batería (Salazar Jiménez, 2015). El proceso de la reacción comienza con la interacción del metal con el electrolito, formando en su superficie zonas anódicas y catódicas. En la región anódica se lleva a cabo la disolución del metal (reacción de oxidación) es decir la cesión de electrones (aumentando de esta forma su número de oxidación), los cuales pasan a la solución en forma de iones. Estos electrones se desplazan sobre la superficie hasta alcanzar las zonas catódicas, en donde son captados o recibidos (disminuyendo su número de oxidación) y formando sobre la región compuestos insolubles (reacción de reducción). Esta sucesión de eventos está ligada a lo que se denomina Potencial Electroquímico, el cual se define como la susceptibilidad o la resistencia de un material metálico a la corrosión, cuyo valor cambia en dependencia de la composición del electrolito. Cuanto más positivo sea el valor de dicho potencial, más noble (resistente) es el material.

Mientras que, en el caso contrario, cuanto más negativo sea este, más reactivo es el material a la corrosión (Santander Morales, 2008).

La Serie Galvánica consiste en una tabla donde se ubican diferentes tipos de materiales con respecto a su Potencial Electroquímico, la Figura 1.1 muestra un ejemplo de una serie galvánica para algunos metales y aleaciones comunes (ECCA, 2022).

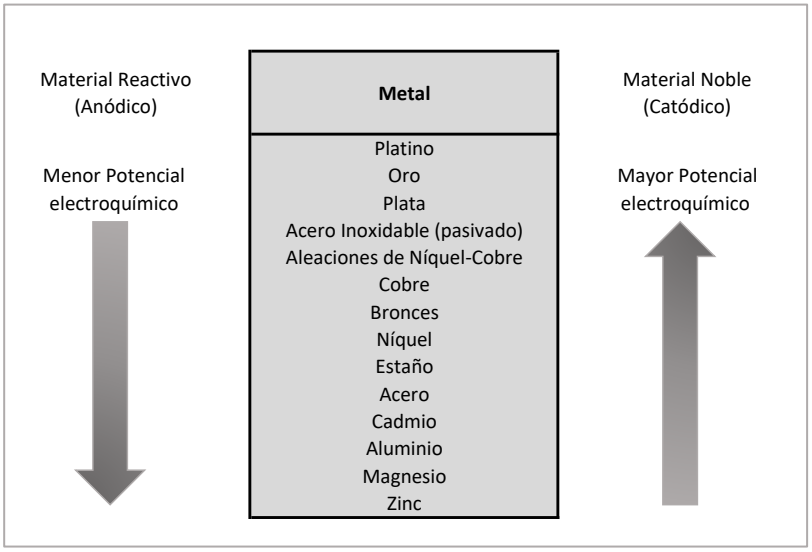


Figura 1.1 Serie Galvánica para metales y aleaciones comunes (ECCA, 2022).

Con base a la Serie Galvánica se puede determinar que metales o aleaciones son útiles bajo cierto tipo de condiciones, puesto que se conocerá su tendencia a corroerse. De lo anterior es posible hacer una separación en tres grupos (Medina, 2020):

- Metales Nobles: El metal es indiferente a la interacción con su ambiente, es decir, el nivel de reacción de corrosión será mínimo o inexistente. La energía libre en la reacción electroquímica es cero o negativa. Ejemplos de este tipo de metal son el oro y el platino.
- Metales Pasivos: El metal pareciese no ser atacado por la corrosión, debido a que posee una especie de película protectora sobre su superficie, por lo que se dice que el metal está pasivado. Es posible ver este tipo de comportamiento en aleaciones de acero inoxidable o en metales como el titanio.
- Metales Activos: Son aquellos de un Potencial Electroquímico más negativo, su superficie no presenta una cubierta protectora o es porosa, presentan una mayor tendencia

natural a la corrosión. Este es el caso típico de los metales más usados en la industria, como el hierro, aluminio y cromo.

Mencionada la causa del fenómeno, así como los materiales a los que afecta, a continuación, se presentan algunos ejemplos frecuentes de tipos de corrosión:

Corrosión generalizada

Denominada en ocasiones como corrosión uniforme, es aquella que desgasta en forma homogénea la superficie del material hasta deteriorarla completamente. Este tipo de desgaste provoca una gran pérdida del material, sin embargo, tiene la ventaja de ser relativamente fácil de predecir y controlar, por lo que su incidencia es baja (Revie, 2011).

Corrosión localizada

Es la manifestada en zonas específicas del material y debido a su difícil detectabilidad, esta representa un mayor riesgo potencial. Las regiones con daño suelen estar determinadas por la naturaleza propia del metal, la geometría o a las condiciones específicas a las cuales se somete.

Corrosión galvánica

Ocurre por la unión física o eléctrica de dos materiales de diferente naturaleza, los cuales al estar en contacto con un electrolito componen los elementos de una celda electroquímica, en la que el material con menor Potencial electroquímico tiende a corroer la superficie de contacto. Entre mayor sea la relación ánodo-cátodo, el proceso ocurrirá a una mayor velocidad (Javaherdashti, 2008).

Corrosión por fisura

Se produce en zonas estrechas en las que la concentración de oxígeno es reducida (en relación con el sistema), este efecto de enrarecimiento de oxígeno provoca que estas zonas reaccionen como un ánodo, propiciando el proceso de corrosión.

Corrosión por picadura (*pitting*)

Se presenta principalmente en materiales pasivados, en los que un aumento de agentes oxidantes o del pH del medio, provocan el desgaste de la capa pasivada, lo que permite a la corrosión desarrollarse en estas zonas.

Corrosión por cavitación

Similar a la corrosión por picadura, la cavitación ocurre en zonas de un material pasivado, en este caso de aquellos que son utilizados para el transporte de líquidos. En este sistema existe la posibilidad de formación de burbujas de aire, las cuales al implosionar contra la superficie del material desgastan la superficie pasivada, facilitando el proceso de corrosión.

Corrosión por erosión

Es observable en materiales pasivados utilizados para el transporte de fluidos, en los que existen materiales que cuentan con partículas de mayor dureza que la superficie pasivada, lo que promueve el desgaste y posteriormente la corrosión.

Corrosión por tensión o fatiga

Ocurre por la presencia de esfuerzos físicos sobre los materiales, lo que promueve el desgaste o debilitamiento del material, causando con ello la formación de fisuras causantes de la corrosión.

Corrosión por desaleación

Se presenta en aleaciones metálicas, en las que uno de los elementos, de mayor afinidad al oxígeno, se separa de la aleación, lo que provoca que el resto del material quede con una superficie porosa, la cual es propicia para la formación de corrosión.

Corrosión filiforme

Se presenta en ambientes con alta humedad sobre materiales con recubrimientos (como la pintura), los cuales, al ser dañados o rasgados, se induce el desarrollo de corrosión, la cual se propaga en forma de hebras o hilos en la superficie (Uhlig & Revie, 2008).

Como ha podido establecerse según el tipo de elemento metálico, las condiciones ambientales variaran la forma y velocidad en la que se presenta la corrosión. Una forma de controlar la corrosión es mediante el reconocimiento y entendimiento de los mecanismos que la causan. Esta información puede enfocarnos hacia el uso de materiales y propuesta de diseños resistentes, y/o la implementación de sistemas así como tratamientos de protección.

La búsqueda de medios para el control de la corrosión ha promovido que se desarrollen diversas técnicas a través de los cuales sea posible su detección. Una de las más valiosas ha sido la denominada como Ruido Electroquímico (EN por sus siglas en ingles de *Electrochemical Noise*). Normalmente por ruido se entiende a cualquier señal aleatoria o perturbatoria no deseada dentro de un sistema, sin embargo, como consecuencia de las investigaciones en corrosión realizadas en los años sesenta, fue posible observar que los sistemas electroquímicos de corrosión presentan en sí mismos, es decir, bajo ningún tipo de excitación externa, señales en forma de pequeñas variaciones (transientes) estocásticas (o aleatorias) en potencial y en corriente. Estas variaciones tienen como origen a la interfaz metal/electrolito, relacionadas con los procesos de transferencia de carga o energía libre por ionización que allí ocurren. Se entiende de este modo que el proceso de desgaste suceda bajo los efectos de la cinética de cargas dados por las condiciones locales suministrando una valiosa fuente de información de los procesos ocurridos en la superficie de los materiales (Sarmiento Klapper & Heyn , 2007).

Uno de los métodos más comunes para el análisis de Ruido Electroquímico es el análisis estadístico, el cual consiste en utilizar los parámetros estadísticos de media, desviación estándar, curtosis y sesgo. Para realizar este método de tratamiento de datos es necesario utilizar un software proporcionado por el equipo de medición o un software de pago. En algunas ocasiones, se puede utilizar o importar los datos desde una hoja de cálculo (normalmente Excel®), sin embargo, el proceso es laborioso e implica al usuario un alto coste de tiempo por la gran cantidad de datos y la adaptación de las fórmulas a cada medición en particular.

Es de destacar que a la fecha no se encuentra disponible algún software de distribución libre que permita ofrecer lo anteriormente propuesto. Por consiguiente, el desarrollo de un software que permita realizar de forma rápida y sencilla los cálculos necesarios para el

desarrollo del método estadístico ofrecería a estudiantes y profesores la posibilidad de reducir los tiempos de análisis de Ruido Electroquímico.

1.2. Hipótesis

Se espera que el desarrollo de un software con interfaz gráfica simplifique el tratamiento de los datos y facilite el cálculo de los parámetros necesarios para el análisis de Ruido Electroquímico.

1.3. Justificación

En la actualidad, el estudio del comportamiento de un material frente a la corrosión tiene un gran interés en toda rama del sector industrial, debido al impacto económico negativo que este fenómeno conlleva para su prevención o para la reparación de los daños causados.

Por la complejidad del fenómeno corrosivo, ha sido necesaria la propuesta de diversas técnicas y procedimientos para su análisis, las cuales centran su atención en conocer las causas, mecanismos de deterioro, su evaluación, y en el desarrollo de formas de control y protección.

Dentro de las técnicas para evaluar el comportamiento de la corrosión se encuentra el análisis de Ruido Electroquímico; el cual, mediante mediciones de voltaje y corriente, y el posterior cálculo de parámetros estadísticos, permite obtener información referente a la naturaleza y velocidad del proceso de corrosión. No obstante, son pocas las alternativas de automatización para el tratamiento de los datos obtenidos.

Actualmente, se cuenta con tres opciones para el tratamiento de los datos: la primera, el desarrollo de una herramienta propia para el cálculo de las operaciones (con el gasto de tiempo que implica), la segunda, emplear el software propio del equipo de medición, y la tercera, el usuario debe adquirir la licencia de softwares especializados, los cuales en su mayoría a pesar de poseer un entorno muy completo para el estudio de mediciones electroquímicas tienen en su contra su alto coste.

El desarrollo de una interfaz gráfica de distribución libre permitirá ofrecer una alternativa a estudiantes y profesores que requieran realizar de forma rápida y sencilla el análisis del Ruido Electroquímico por medio del método estadístico, de forma independiente al instrumento de medición o a la adquisición de licencias para el uso de software.

1.4. Objetivo General

Desarrollar una interfaz gráfica de software libre que permita la automatización del análisis de Ruido Electroquímico.

1.5. Objetivos Específicos

- Realizar una investigación de los softwares disponibles para el análisis de Ruido Electroquímico.
- Seleccionar con base en la bibliografía los parámetros estadísticos necesarios para el análisis de Ruido Electroquímico.
- Establecer el algoritmo de programación que desarrolle la secuencia del cálculo de los parámetros estadísticos.
- Diseñar una interfaz gráfica amigable al usuario con los elementos adecuados para la representación visual de los resultados.

CAPÍTULO II

2. MARCO TEÓRICO

En este capítulo se describe a la técnica de Ruido Electroquímico y sus métodos de análisis; el concepto y características de una Interfaz Gráfica de Usuario, así como su implementación en el estudio de EN.

2.1. Ruido Electroquímico

El Ruido Electroquímico es una técnica utilizada ampliamente para estudiar el proceso de corrosión en varios materiales metálicos con el que se consigue información valiosa a partir de las mediciones de las variaciones de corriente y potencial durante el proceso de corrosión. La técnica tiene como ventaja no producir señales distorsionantes y evitar alteraciones artificiales durante la medición (Zhao & Li, 2007).

La aplicación de la técnica de Ruido Electroquímico en el ámbito de la corrosión se puede clasificar en tres campos diferentes: investigación básica de fenómenos corrosivos, prueba y evaluación de la corrosión, y monitoreo a nivel industrial de la corrosión (Goellner, 2004). Cada ámbito tendrá variaciones en cuanto al registro y adquisición de la señal de ruido, por lo cual no existe un sistema universal para su estudio y la implementación de la técnica estará directamente ligada al tipo de aplicación buscada.

De forma general, un sistema de Ruido Electroquímico se compone de tres partes principales: En primer lugar, el elemento de medida, y más específicamente el Electrodo de trabajo, considerándose con él todos los factores que puedan actuar como variables de influencia. En segundo lugar, se considera el equipo de medición electroquímica (potencial y corriente) y los accesorios para realizar las mismas, incluyendo los accesorios para amplificación y filtros. Finalmente, se considera a los parámetros técnicos dependientes de la aplicación (Sarmiento Klapper & Heyn , 2007).

Como se mencionó anteriormente, no existe un sistema universal para llevarse a cabo la captación del Ruido Electroquímico, sin embargo, es posible presentar una configuración

general del arreglo experimental en la Figura 2.1. En la imagen se pueden observar los Electrodos de trabajo (W1 y W2), electrodo de referencia (RE), dispositivo para medida de corriente basado en el principio de baja resistencia (ZRA del inglés *Zero Resistance Ammeter*), dispositivo para medida de potencia con alta resistencia de entrada (V), filtros o amplificadores de señal, el sistema de adquisición de datos formado por un conversor de señal análogo-digital y una computadora (Sarmiento Klapper & Heyn , 2007).

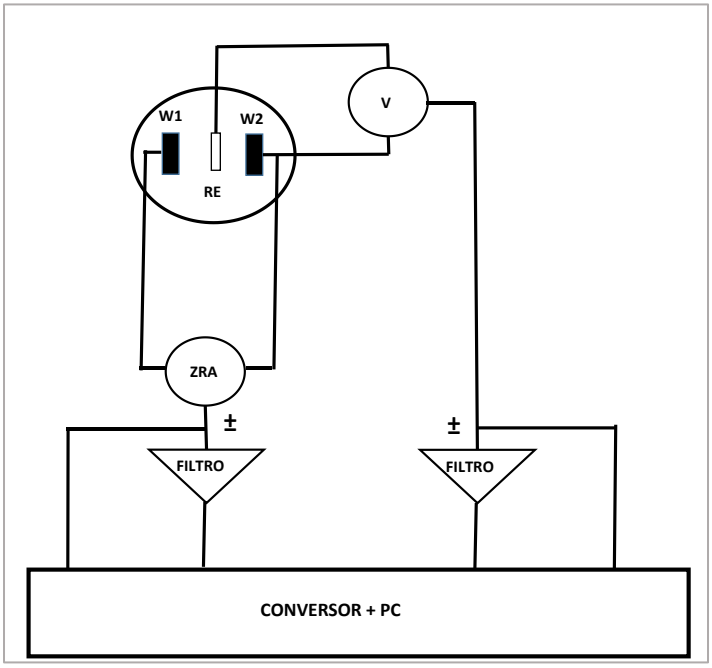


Figura 2.1 Representación esquemática de un sistema para la adquisición de la señal de Ruido Electroquímico.

En el análisis de los datos obtenidos durante la captación de la señal de Ruido Electroquímico es común asumir que los valores determinados de las variables involucradas no cambian durante el intervalo de medida. En la práctica esto no ocurre frecuentemente y un proceso de "extracción de tendencia" debe ser implementado por adelantado. Esto ocurre dado que existen influencias externas que producen efectos en la señal de ruido registrada, como por ejemplo el transporte de oxígeno a la superficie de un metal que hace difícil separar la señal del Ruido Electroquímico después de realizar la medida; lo que justifica la importancia que tiene el proceso de reducción de tendencia en la etapa de medición. En caso de no ser posible la aminoración de las influencias externas, se recomienda utilizar métodos de análisis de señal más complicadas, como podrían ser las curvas de polarización lineal y potencio-

dinámicas, así como por espectroscopia de impedancia electroquímica (Malo Tamayo & Uruchurtu Chavarín, 2000).

El proceso de extracción de tendencia comúnmente utilizado consiste en el ajuste de los datos a una línea recta (ajuste o regresión lineal) o, excepcionalmente, a un polinomio de orden más alto y se toma la desviación de cada dato real relativa al valor ajustado como ruido. Una vez corregida, la información puede ser estudiada directamente en el dominio del tiempo mediante el trazado de las oscilaciones del parámetro en cuestión, o como alternativa, los datos pueden transferirse al dominio de la frecuencia, para lo que haría necesario un tratamiento de los valores utilizando el método de Máxima Entropía (ME) o la Transformada Rápida de Fourier (FFT, por sus siglas en inglés de *Fast Fourier Transform*) (Mariaca & Bautista, 1997).

2.1.1. Métodos de análisis de Ruido Electroquímico

Los métodos más comunes que han sido desarrollados para analizar el Ruido Electroquímico son: análisis de las series de tiempo, análisis estadístico, análisis espectral y análisis por transformada de Wavelet. A continuación se describe cada uno de ellos en los siguientes incisos.

a) Análisis de las series de tiempo

Es el método más simple y directo que consiste en examinar visualmente las series de tiempo para la identificación de detalles característicos de los tipos de corrosión. Por ejemplo, la detección visual de transitorios de rompimiento y repasivación (formación de una película relativamente inerte sobre la superficie de un material) o de oscilaciones asociadas a resquicios de corrosión por picadura (Malo Tamayo & Uruchurtu Chavarín, 2000).

Por la estructura definida de la serie de tiempo y la relación establecida entre las características de los procesos físicos subyacentes, es fácil obtener información que pueda extraerse de este método.

b) Análisis estadístico

El análisis estadístico trata a la serie de tiempo como una colección de potenciales o corrientes individuales, ignorando la relación entre un valor y el siguiente (muestra de la población). Es decir, se considera a cada valor como un dato individual distribuido en un punto específico del tiempo de medición (serie de tiempo) y a las variaciones del comportamiento grupal como indicadores del proceso de corrosión (Casquete García , 2016).

Para determinar el comportamiento grupal de los datos individuales se lleva a cabo el cálculo de medidas estadísticas como: media, desviación estándar, curtosis y sesgo; así como de parámetros especiales dependientes de éstas, como la Resistencia de polarización al ruido y el Índice de localización. Los resultados de las medidas anteriormente mencionadas proporcionan información valiosa de la naturaleza del fenómeno corrosivo. A continuación, se presenta una breve descripción de cada una de ellas:

La media o promedio es el más común de los parámetros de interés, debido a que sus fluctuaciones en largos periodos de tiempo están directamente relacionados a cambios en el proceso de la corrosión (especialmente en el caso del tipo localizada).

La desviación estándar de potencial (σ_v) o de corriente (σ_i) pueden ser calculadas para monitorear la intensidad del proceso de corrosión. Altos valores de σ_i indican mayor actividad corrosiva. Esta medida estadística resulta particularmente útil para la técnica de Ruido Electroquímico; ejemplo de ello es en la detección de corrosión localizada, debido a que las variaciones de la señal serán de mayor intensidad a lo largo del tiempo en comparación a las obtenidas en el caso de la corrosión generalizada.

La curtosis es una medida de distribución de los datos y, en términos generales, los datos de Ruido Electroquímico no tienden a tener una distribución normal o gaussiana (valor de cero). Es más frecuente encontrar una distribución más acentuada (valores mayores a cero) o aplanada (valores negativos).

El sesgo es utilizado para conocer la simetría de la distribución de datos. Un valor de cero indica una distribución simétrica alrededor de la media. En caso de tener valor mayor o menor a la media se considerará como positivo o negativo, respectivamente. En el caso del Ruido

Electroquímico la medida del sesgo en corriente implica la activación de los electrodos de trabajo.

La denominada Resistencia de polarización al ruido (R_n) es posible de calcular por medio de la desviación estándar, es definida (por analogía a la ley de Ohm) como la relación entre la desviación estándar de potencial (σ_v) y la desviación estándar de ruido en corriente (σ_i) (Ecuación (1)). Por consiguiente, esta ecuación representa la resistencia del proceso electroquímico (Bautista & Vergara, 1997).

$$R_n = \frac{\sigma_v}{\sigma_i} \qquad (1)$$

El Índice de localización es un parámetro que implica la relación que existe entre la desviación estándar y la raíz de la media cuadrática de los datos. Esta medida ha sido considerada por diversos autores como una forma de determinar el tipo de corrosión prevaleciente, por ejemplo, la corrosión localizada es considerada en valores entre 1.0 y 0.1, para corrosión mixta de 0.1 y 0.01, mientras que para generalizada entre 0.01 y 0.001 (Botana, 2002).

El cálculo de las operaciones anteriormente descritas proporciona una forma fácil y rápida para la interpretación del proceso de corrosión en un material, además, tienen como ventaja el poder ser obtenidas mediante la iteración de operaciones matemáticas básicas, lo que las hace ideales para su implementación en un algoritmo computacional.

c) Análisis espectral

En el análisis espectral, las señales de corriente y voltaje son transformadas del dominio de tiempo al dominio de frecuencia usando la Transformada Rápida de Fourier (FFT sus siglas en inglés de *Fast Fourier Transform*) o el Método de la Máxima Entropía (MEM, siglas en inglés de *Maximum Entropy Method*). El resultado de la transformación de dominio es la denominada Densidad Espectral de Potencia (PSD por sus siglas en inglés de *Power Spectral Density*), la cual al ser graficada permite establecer de una manera más fácil (con respecto a lo observable en el dominio del tiempo) el intervalo de datos donde se concentra el mayor

número de variaciones; esto permite la comparación entre señales y detectar cambios en el fenómeno corrosivo (Luengas & Toloza, 2020).

Normalmente valores altos de PSD indican procesos de abrasión en la superficie de un material y mayor aceleración del proceso de corrosión debido al desgaste; en cambio, valores bajos de PSD es indicio de materiales que han mantenido intacta o casi intacta su capa pasiva (Shi & Zhang, 2008).

Para aplicar el análisis espectral se requiere que la señal sea estacionaria, es decir, que no varíe con respecto al tiempo y la caracteriza un valor de media que tiende a cero. Sin embargo, no siempre es posible encontrar este tipo de señal en sistemas de corrosión ya que comúnmente las señales son no estacionarias, especialmente al inicio del proceso, lo que origina problemas con la estimación espectral.

Para resolver esta problemática se utiliza un artificio matemático de remoción de la tendencia. Posteriormente, es común (mas no obligatorio) utilizar una función ventana (función que limita a un rango especifico los valores y que fuera de éstos asume valores nulos) para escoger secciones de la serie de tiempo que deseen analizarse. La ventana reduce la extensión de las frecuencias a lo largo del espectro y una vez realizado se aplica el algoritmo de Transformada Rápida de Fourier o de Máxima Entropía para obtener el espectro de potencia (equivalente a la varianza) (Rubio Foces, 2019). El espectro obtenido es de un comportamiento $1/f$ en el ancho de banda de bajas frecuencias, considerado presente en muchos procesos de la naturaleza, incluyendo el de la corrosión.

Al observarse gráficamente, el espectro ofrece la posibilidad de caracterizar los tipos de corrosión, lo cual se consigue a través del estudio de los cambios de la pendiente visibles en el gráfico. De esta manera, a mayor pendiente la superficie desgastada permanecerá en estado pasivo, en cambio, al obtener valores de pendiente negativos el proceso de corrosión se asemeja a un proceso corrosivo generalizado. Por su parte, los valores intermedios se corresponden con el proceso de corrosión localizada (Obando Ramírez, 2013).

d) Análisis por Transformada de Wavelet

A pesar de que los métodos del análisis espectral son útiles para analizar fenómenos de señales estacionarias (de frecuencia estable o constante), difícilmente pueden ser utilizados

para señales no estacionarias. La transformada de Wavelet (WT por sus siglas en inglés de *Wavelet Transform*) ha sido propuesta como una herramienta alternativa para sobreponer las limitaciones de la Transformada de Fourier en el análisis de información del Ruido Electroquímico. La WT puede proveer información de los transitorios en el tiempo y la posibilidad de trabajar con señales no estacionarias, también ha sido utilizada para diferenciar los tipos de corrosión y estudiar en sí el mecanismo (Zhao & Li, 2007).

La Transformada de Wavelet es un camino relativamente nuevo de procesamiento de señales de ruido. La forma continua de la transformada es una función oscilatoria real o compleja de promedio cero y largo infinito. En su versión discreta se utiliza para analizar la información del Ruido Electroquímico, obtenido en la práctica por el algoritmo de la Transformada Rápida de Wavelet (FWT por sus siglas en inglés: *Fast Wavelet Transform*) y el uso de filtros de diferentes cortes de frecuencias (filtro de paso bajo y filtro de paso alto) para el análisis de la señal en diferentes escalas. Después de filtrar, la información es sometida a una reducción de muestreo, que consiste en suprimir uno de cada dos coeficientes consecutivos de la información filtrada. Al final, la señal se descompone en coeficientes de alta y baja frecuencia que contienen la información sobre las fluctuaciones locales y la tendencia general de la señal respectivamente. La distribución de la señal completa puede ser estimada por el Gráfico de distribución de energía, el cual representa a la energía relativa acumulada (Qiao & Ou, 2007).

Como se puede observar, la técnica de Ruido Electroquímico ofrece la posibilidad de elegir entre diferentes métodos, cada uno de ellos tiene ventajas particulares, así como de limitantes. Sin embargo, el método estadístico ha prevalecido en el área experimental y de campo como el de mayor aceptación, ya que requiere de un número mínimo de variables para ofrecer resultados de alta confiabilidad en la determinación de los procesos de corrosión y realizar una fácil comparación con estándares establecidos.

Es importante mencionar que, para la implementación del método estadístico, el tratamiento de datos es mínimo y no requiere de una alta demanda de recursos en equipo de cómputo. Lo anterior hace atractivo al método para ser implementado mediante un algoritmo computacional. Sin embargo, a la fecha no es posible encontrar un software no dependiente del equipo de medición o de un código creado que haya sido liberado posterior a una

investigación; siendo tarea del usuario interesado llevar a cabo la creación de una herramienta propia para el uso de este método.

2.2 Interfaz Gráfica de Usuario

Desde la invención de la computadora, sin importar el tamaño y la forma, sus componentes básicos siguen siendo tres: un dispositivo de entrada de datos, una unidad de procesamiento para convertir a los datos en información, y un dispositivo de salida que haga posible al usuario la obtención de la información. No obstante, y a pesar de parecer no verse afectados por el paso del tiempo, ha sido la Interacción Humano-Computadora (IHC) la que ha "transformado" a estos componentes, debido a que esta interacción tiene como objetivo principal: transmitir de forma sencilla las tareas a la máquina y permitir al ser humano acceder a la información en modo más natural y libre. De esta forma, IHC recae principalmente en los equipos de entrada, salida y en el software que los gestiona.

La interacción humano-computadora puede considerarse como el lenguaje de comunicación entre la computadora y el ser humano, el cual se realiza mediante el uso de una determinada interacción para completar alguna tarea diseñada. Esta comunicación ha sufrido cambios progresivos principalmente en tres etapas: la primera consistía en comandos de entrada efectuados por un dispositivo conformado por teclas denominado teclado. La segunda etapa, que transcurre actualmente, se basa en la Interfaz Gráfica de Usuario y el dispositivo principal de entrada es el ratón. La tercera etapa se apoya en dispositivos táctiles, de reconocimiento de voz o captura de imagen, en esta etapa la interacción intenta aproximarse cada vez más a las formas naturales de comunicación humana (táctil, voz y visual respectivamente).

Para el soporte de la IHC ha sido necesario el desarrollo de programas informáticos que actúen como vínculo entre la entrada de las instrucciones que recibe la computadora y la información que recibe el usuario. Sin embargo, con el paso del tiempo, se ha demandado a las computadoras tareas de mayor complejidad (lo que involucra de igual manera instrucciones más precisas), volviendo indispensable la implementación de programas cada vez más completos que puedan hacer de la "comunicación" humano-computadora algo fácil e intuitivo.

Por lo anterior, dentro del desarrollo de software se da un peso fundamental a la creación de este tipo de programas, denominados como Interfaz de Usuario (UI siglas en inglés de *User Interface*), sobre todo porque al considerar una interfaz pobre o deficiente limitará la capacidad de la computadora para cumplir con eficiencia las tareas asignadas, afectando negativamente al resultado esperado por parte del usuario. Tradicionalmente, este tipo de interfaz utiliza una gestión de entrada-salida de datos textual, esto implica que el usuario deba introducir comandos precisos de texto para formar la secuencia o cadena de instrucciones, la cual pueda ejecutar la computadora y devolver en forma de caracteres de texto.

Las UI son consideradas como el tipo de IHC más simples, fáciles de programar y con menor demanda de recursos, sin embargo, a medida que la complejidad de la tarea aumenta se incrementan de igual manera sus desventajas, entre ellas es posible distinguir especialmente tres: la primera es la simplicidad, debido a que solo pueden presentar un mínimo de detalle en las opciones y características del ambiente de trabajo. La segunda desventaja es el ser poco intuitivas, es decir, que requiere forzosamente de un conocimiento previo de sus características para ser abordado, dificultando su acceso a un usuario novel. Como tercera desventaja se encuentra el tipo de información devuelta ya que al usuario sólo es posible mostrarle cadenas de caracteres, pudiéndose (en algunos casos) elegir la posición donde se quiere que éstos aparezcan.

Un paso más adelante en la evolución de las interfaces es el caso de la Interfaz Gráfica de Usuario (GUI por sus siglas en inglés de *Graphical User Interface*), en la cual la IHC es apoyada no solamente con texto, sino también con imágenes, y demás objetos pictóricos (iconos, cuadros, ventanas, botones, temas, etc.). Estas características permiten al usuario acostumbrarse con mayor facilidad al entorno de trabajo, lo que permite la mejora en la transmisión de instrucciones y obtención de resultados. La gestión de datos de entrada para las GUI tiene la ventaja de encontrase a la espera de interacción por parte del usuario, ya que cada uno de los elementos corresponden a una acción y consecuentemente a una instrucción; lo que en el lenguaje de programación de interfaces gráficas se denomina evento. Para el manejo de cada una de las acciones, cada uno de los elementos de la interfaz están ligados a un código particular, lo que permite que se ejecuten instrucciones concretas; a este tipo de programación se le llama código de gestión de eventos (Gallego Carrillo, 2005).

La naturaleza de las GUI, al contrario de las UI, tiende a hacerlas más cercanas al ideal de la IHC, debido a que su finalidad es crear una comunicación de instrucciones más sencilla y mucho más efectiva. De igual manera, la calidad de información recibida es mayor ya que puede ser representada por estímulos visuales que facilitan su interpretación, permitiendo a cualquier usuario un uso más amigable de ella. Por lo anterior, hoy en día es común que las interfaces graficas sean el estándar de una aplicación.

Descritas las ventajas de la implementación de una Interfaz Gráfica de Usuario conviene ahora hacer una breve descripción de los elementos estándar que la componen:

- Ventanas

  Son elementos capaces de contener a otros y por lo regular tienden a poder ser desplazados libremente por la pantalla. Se pueden identificar cuatro tipos básicos de ventanas:

  a) De aplicación. Estas contienen a los elementos generales del programa e incluyen a los botones de maximizar, minimizar y cerrar la aplicación.

  b) Cuadros de diálogo. Son comúnmente de carácter emergente y mostrados por un breve periodo de tiempo. También, son abiertos al ejecutarse una situación que requiere especial atención por parte del usuario y es común que al emerger bloqueen a los demás elementos hasta realizar la acción notificada.

  c) Internas. Se pueden considerar básicamente como contenedores ya que almacenan a otros elementos de la aplicación como gráficos, controles, herramientas, etc. Este tipo de ventana puede ser desplazada libremente dentro de los límites de la ventana de aplicación.

  d) Sin marco. Este tipo de ventana no presentan botones de interacción con el usuario y su función es indicar la ejecución del programa u ofrecer una estimación del tiempo de un proceso.

- Componentes

  Son elementos de la interfaz a los que se asocia una funcionalidad y por lo tanto tienen una entidad propia. Ejemplos de ellos son: botones, barras de desplazamiento, indicadores, desplegables, paneles, contenedores.

Es posible hacer una distinción de los componentes entre los de tipo control y los de tipo contenedor. Los de tipo control, tienen por función obtener los datos y posteriormente entregar la información al usuario. Por su parte, los de tipo contenedor son los encargados de mostrar a otros elementos dependiendo de la ventana activa y del proceso en ejecución.

- Menús

Son el listado de opciones posibles a ejecutar, las cuales iniciaran un evento o desplegaran un nuevo menú específico de la acción deseada. Los menús ofrecen al usuario un rápido acceso a las funciones básicas y una adecuada jerarquización de las opciones para la transmisión de instrucciones complejas.

Un buen desempeño de una interfaz dependerá de la correcta selección inicial y distribución de los elementos anteriormente descritos, así como, de una adecuada asociación de código de instrucción a cada uno de ellos.

2.2.1 Interfaces graficas de usuario para el análisis de Ruido Electroquímico

A continuación, se hará un breve resumen de las herramientas de tratamientos de datos más utilizadas:

- **Software ECN – Electrochemical Noise Measurement**

El software *Electrochemical Noise Measurement* ECN (por su nombre en inglés) es un software desarrollado por Instruments Metrohm y proporcionado únicamente al adquirir equipo de experimentación de su marca. Este programa permite analizar los datos obtenidos de Ruido Electroquímico utilizando el método de la transformada rápida de Fourier (FFT). Además, el software ofrece un entorno de trabajo para el filtrado de las señales de ruido, así como un conjunto de métodos matemáticos para el tratamiento de los datos de medición.

Las áreas de aplicación a las que está enfocado son el estudio de corrosión por picadura y el estudio de recubrimientos por alta impedancia (Autolab, 2022). La Figura 2.2 muestra un ejemplo del grafico resultante del análisis por series de tiempo obtenido con el software ECN.

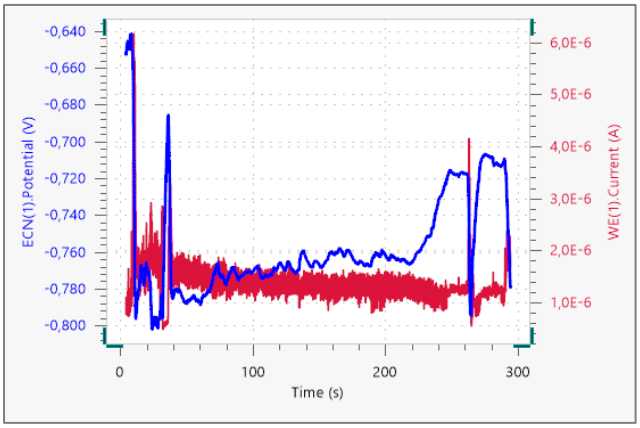


Figura 2.2. Series de tiempo obtenido con el software ECN de Metrohom Autolab.

- **Software ECN Analysis V1 - Electrochemical Noise ECN Analysis Version 1.0**

El software *Electrochemical Noise ECN Analysis* fue diseñado por el investigador Jose Dingle miembro de la Escuela de Ingeniería Eléctrica y Electrónica perteneciente a la Universidad de Manchester de Inglaterra. El software tiene como particularidad ofrecer una interfaz gráfica de distribución condicionada a un registro para la obtención de clave de activación por medio de una página web. El ECN *Analysis* V1 es compatible con gran variedad de potenciostatos ya que cuenta con un convertidor propio de formato de archivos de datos.

El software proporciona los resultados de impedancia y resistencia al ruido, así como la densidad espectral, la carga y la frecuencia de los eventos de la corrosión mediante la implementación del método estadístico y del dominio de la frecuencia. Dado el diseño de su interfaz, estos resultados son representados de forma gráfica (Dingle, 2012) .

La principal desventaja e inconveniente de este software es que, a pesar de poder descargarlo de forma gratuita, el software después de su liberación no recibió soporte o actualizaciones desde el 2012 por lo que solo es posible su ejecución en sistemas operativos anteriores a ese año. La Figura 2.3 muestra un ejemplo de la interfaz gráfica del software ECN *Analysis* V1.

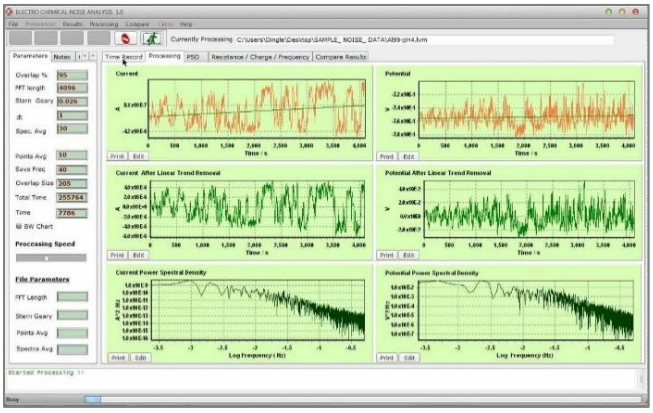


Figura 2.3. Interfaz gráfica de ECN Analysis V1 con grafica de resultados.

- **Software DigiElch Electrochemical Noise**

El software *DigiElch Electrochemical Noise* fue diseñado por *Gamry Instruments* cuya sede se encuentra en Pensilvania. Esta empresa diseña y construye accesorios e instrumentación electroquímica de alta precisión. Con la adquisición de equipos se pone a disposición del cliente alguna de las versiones que componen la paquetería.

La interfaz está basada en el sistema operativo Windows y puede importar y exportar datos al programa Microsoft Excel. Además del tratado de los datos propios del equipo de medición, el software permite la simulación de una amplia variedad de sistemas electroquímicos (Gamry Instruments, 2022). La versión completa de *DigiElch Electrochemical Noise* puede ser adquirida de forma independiente mediante una prueba con vigencia de 14 días con registro previo. La Figura 2.4 muestra un ejemplo de la interfaz gráfica del software *DigiElch Electrochemical Noise*.

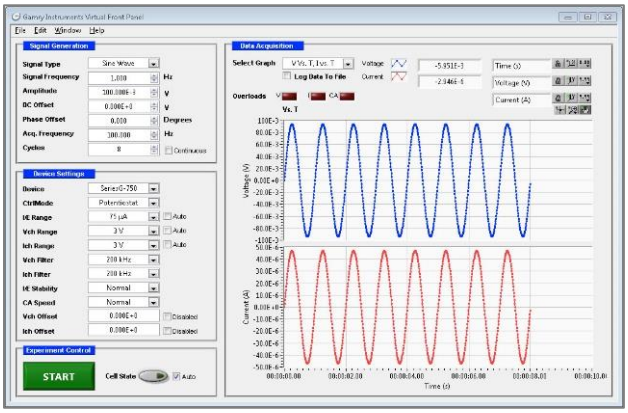


Figura 2.4. Interfaz gráfica del software para potenciostatos PCI4/Series G.

- **Software MARE. V1**

El software MARE. en su versión 1 fue desarrollado por el Dr. Jorge A. Ruiz Enciso, investigador del Instituto Nacional de Investigaciones Nucleares de México. MARE. V1 es un programa diseñado para la supervisión en tiempo real de la corrosión mediante el análisis de Ruido Electroquímico. El análisis es realizado por bloques de datos. Seleccionando un rango de la señal de Ruido Electroquímico mediante cursores, los parámetros como la Media, Valor RMS, Desviación Estándar, Varianza, entre otros son calculados. Además, los indicadores visuales del instrumento permiten determinar si la corrosión es de tipo uniforme, mixta o localizada (Ruiz Enciso & Rojas Salinas, 2007). La Figura 2.5 muestra la pantalla en ejecución del software MARE. V1.

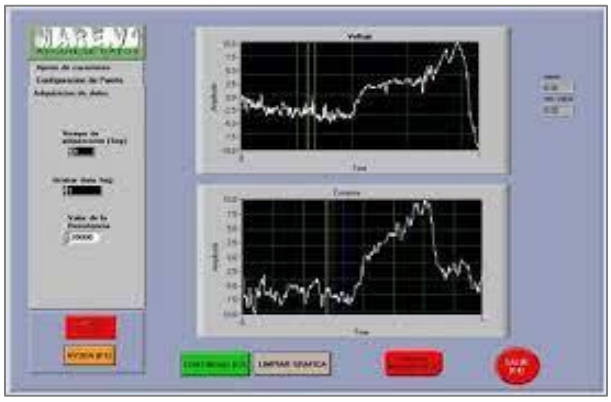


Figura 2.5. Pantalla de ejecución del software MARE.V1.

- **Software IVIUMSOFT**

El software IVIUMSOFT es un programa de pago desarrollado por la empresa *Allumcorp*, dedicada a proveer consumibles y equipos de laboratorio para electroquímica. IVIUMSOFT es una herramienta o entorno completo para análisis electroquímico que permite el control de diversos instrumentos (sean o no propios de *Allumcorp*) dada su alta compatibilidad varias plataformas como Labview, UB, Delphi, C, etc. La interfaz está basada en el sistema operativo Windows y presenta una gran flexibilidad para guardar datos. El software está pensado para ser integrado por módulos en requerimiento de sus clientes, por lo cual sus características y funcionalidades pueden ser extendidas (Ivium Technologies, 2022).

Entre las funcionalidades que más se destacan son la aplicación de técnicas como análisis de dominio de tiempo (corriente de corrosión, Índice de localización/picadura, resistencia de la

corrosión), análisis de frecuencia por transformada rápida de Fourier (espectros de impedancia y resistencia a la corrosión), y análisis de frecuencia por máxima entropía (número definible de coeficientes de modelo). En la Figura 2.6 se observa una de las pantallas del análisis realizado por el software.

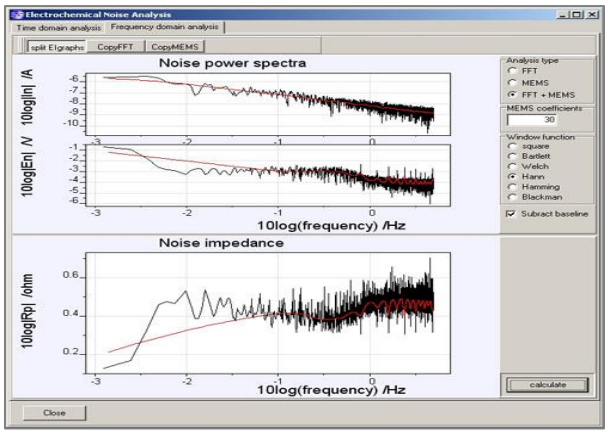


Figura 2.6. Interfaz gráfica del software IVIUMSOFT para análisis de Ruido Electroquímico.

Como ha podido observarse, a pesar de ser una valiosa herramienta para el estudio del proceso corrosivo, la principal fuente de software disponible para el análisis de Ruido Electroquímico proviene del instrumento de medición, o de la adquisición de una la licencia de uso de software especializado. Por otra parte, las iniciativas para la creación de un software independiente se han centrado exclusivamente en el diseño de una herramienta propia (enfocada a la investigación en turno) y de código restringido a discreción de la autorización expresa del creador. Ante tal panorama se ha optado por el uso de aplicaciones alternativas que puedan solucionar la necesidad de un instrumento de análisis, como puede ser el uso de hojas de cálculo de Microsoft Excel®. Sin embargo, la gran mayoría tiene dos limitantes principales: la primera, tener un escaso ambiente gráfico, lo que dificulta su operación a usuarios no familiarizados. La segunda limitante es el tiempo que demanda la preparación de las plantillas de trabajo, dado que en ocasiones los datos no se ajustan a las restricciones propuestas, lo que involucra un mayor esfuerzo para realizar ajustes.

Con motivo de lo anteriormente mencionado, se hace la propuesta del uso del software Matlab®, para el diseño de un programa que permita realizar los cálculos requeridos para este tipo de análisis, ya que es un entorno de programación que permite entre sus

características la creación de interfaces de usuario amigables y que demandan al usuario una menor complejidad para la transmisión de las instrucciones.

2.2.2 Interfaces graficas en Matlab®

Matlab® (abreviatura en ingles de *Matrix Laboratory*) es un software de cómputo numérico interactivo basado en matrices para cálculos científicos y de ingeniería. Matlab ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M).

Entre sus características básicas se encuentra la manipulación de arreglos numéricos (matrices), representar datos y funciones, la creación de Interfaces de Usuario (UI), así como su flexibilidad al tener compatibilidad con otros lenguajes y dispositivos de hardware. Además, puede ampliar sus capacidades al integrar cajas de herramientas para tareas específicas (*Toolbox*) tales como *Deep Learning* e inteligencia artificial, Sistemas de Control, Procesamiento de imágenes y señales, Análisis estadístico de datos financieros, etc. (Moore, 2007).

Para la creación y diseño de interfaces, Matlab® dispone de tres alternativas: Funciones de Matlab®, Entorno GUIDE y, recientemente, *App Designer*. Cada una de las opciones ofrece un conjunto de funcionalidades cuya elección por parte del usuario dependerá de las necesidades del tipo de proyecto, así como de los objetivos buscados en el trabajo.

Las interfaces con Funciones de Matlab® requieren codificar tanto el diseño como el comportamiento de la aplicación. Es decir, no solo se requiere programar las utilidades para el usuario sino también todos los aspectos de la interfaz (posición, propiedades de componentes, configuración de aspecto, etc.) lo que consume mucho tiempo de programación. La Figura 2.7 muestra la ventana de trabajo de Matlab® para la creación de interfaz mediante el uso de funciones.

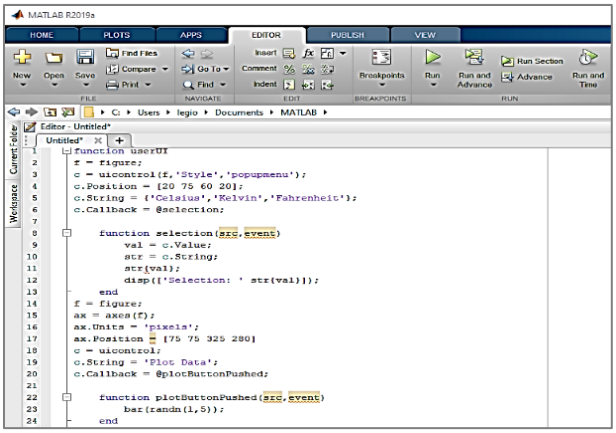


Figura 2.7. Ventana de trabajo de Matlab para la creación de interfaz mediante uso de funciones.

Por otro lado, el entorno GUIDE hace uso de un entorno grafico para la creación de interfaces. Este entorno fue introducido a la paquetería de Matlab® en 2016 y cuenta con la particularidad de permitir codificar el diseño y funciones de comportamiento de la aplicación por separado.

GUIDE permite economizar tiempo en programación al generar partes del código en forma automática. Mediante funciones predeterminadas denominadas *Callbacks*, los elementos de diseño se asocian a instrucciones establecidas y no se requiere generar líneas de código adicionales. Otra ventaja importante es tener agrupadas todos los componentes en paneles para ser seleccionados y arrastrados al área de trabajo (Cid Espinosa, 2018). La Figura 2.8 muestra el entorno grafico GUIDE en Matlab® con algunos elementos básicos de trabajo.

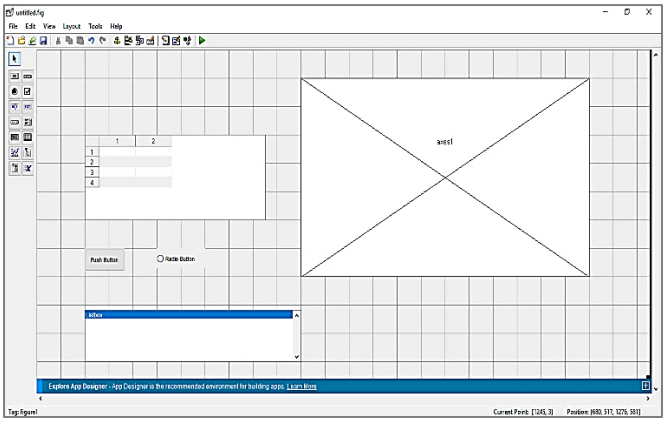


Figura 2.8. Entorno grafico GUIDE en MATLAB con algunos elementos básicos de trabajo.

El más reciente entorno proporcionado por Matlab® es *App Designer*. *App Designer* se distingue por permitir crear interfaces de usuario con carácter profesional, pero con la ventaja

de que el programador no requiere un alto nivel en conocimiento de programación para poder utilizarlo.

Al igual que GUIDE, *App Designer* permite trabajar la interfaz en dos partes, la primera enfocada al código de las funciones a ejecutar; y la segunda que se encarga puramente del aspecto visual de la aplicación. La facilidad para el manejo del entorno grafico es lo que destaca a *App Designer* de otras opciones debido a la variedad de componentes disponibles y cuyo grado de configuración por programación llega ser mínimo.

Una posibilidad importante que ofrece *App Designer* es la de compartir las interfaces creadas a terceros sean o no usuarios de Matlab®, esto se debe a la herramienta *Matlab Compiler* que permite el empaquetado de las aplicaciones para su uso independiente, sean en formato de aplicación web o de instalación tradicional en el equipo (Toledano Vivar, 2019).

El presente trabajo utiliza el entorno de *App Designer* (mostrado en la Figura 2.9) para el diseño y programación de la interfaz orientada al análisis de Ruido Electroquímico. Los elementos que lo componen así como su estructura se detallarán de forma más específica en el Capítulo 3.

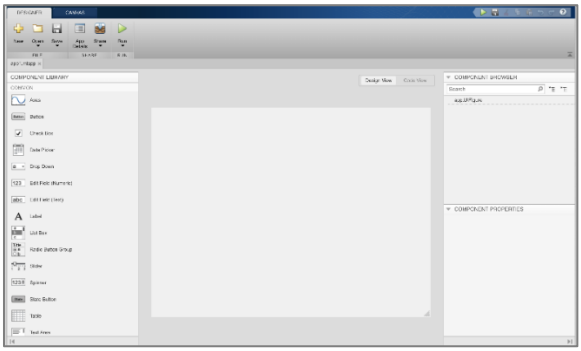


Figura 2.9. Entorno básico de App Designer para el diseño de interfaces gráficas.

CAPÍTULO III

3. METODOLOGÍA

En este capítulo, se presentará la formulación matemática del cálculo de los parámetros estadísticos, así como una breve descripción de los elementos que fueron necesarios para el desarrollo de la Interfaz Gráfica de Usuario. Para realizar una mejor descripción, la metodología se divide en dos partes: el Análisis de Ruido Electroquímico y el Diseño de la Interfaz Gráfica.

3.1. Análisis de Ruido Electroquímico

El análisis de Ruido Electroquímico propuesto en la interfaz se llevó a cabo mediante el cálculo de los siguientes parámetros estadísticos:

- Media
- Desviación estándar
- Curtosis
- Sesgo
- Índice de localización

Para realizar una comparativa entre la influencia de aplicar o no una técnica de remoción de tendencia, las operaciones estadísticas en el presente trabajo se realizaron tanto para los valores originales de entrada, como para la base de datos con el tratamiento, la fórmula empleada para la extracción de tendencia es mostrada en la Ecuación (2).

$$nt = X_j - X_{j+1} \tag{2}$$

El cálculo de la Media es útil para expresar la tendencia central de un conjunto de datos, en el presente trabajo se empleó como fórmula a la Ecuación (3) (Surendra, 2005).

$$\mu = \lim_{n\to\infty}\left[\frac{\sum_{j=1}^{n} X_j}{n}\right] \tag{3}$$

Donde X_j hace referencia a los datos de voltaje y corriente y n al número total de datos.

Mencionada en el capítulo anterior como un parámetro elemental para el análisis estadístico de Ruido Electroquímico, la Desviación estándar proporciona la medida de dispersión de los datos, la cual sirve para interpretar la intensidad del proceso de la corrosión. La operación que fue implementada es mostrada en la Ecuación (4).

$$\sigma = \lim_{n\to\infty}\left[\frac{\sqrt{\sum_{j=1}^{n}\left(X_j-\mu\right)^2}}{n}\right] \tag{4}$$

Donde μ hace referencia a la media calculada.

La Curtosis es una medida estadística que proporciona el grado de concentración de los valores de una variable alrededor de su media. En el presente trabajo la fórmula utilizada fue definida por la Ecuación (5).

$$k = \frac{n\left\{\sum_{j=1}^{n}\left(X_j-\mu\right)^4\right\}}{\left\{\sum_{j=1}^{n}\left(X_j-\mu\right)^2\right\}^2} \tag{5}$$

El cálculo del sesgo se realizó mediante la Ecuación (6), operación que proporciona la asimetría de los datos con respecto al valor promedio.

$$s_k = \frac{n^{1/2}\left\{\sum_{j=1}^{n}\left(X_j-\mu\right)^3\right\}}{\left\{\sum_{j=1}^{n}\left(X_j-\mu\right)^2\right\}^{\frac{3}{2}}} \tag{6}$$

Como fue descrito en el capítulo anterior, las medidas estadísticas propuestas permitieron el cálculo de otro parámetro indicador del fenómeno de la corrosión. De esta manera fue posible obtener el Índice de localización representado en la Ecuación (7).

$$IL = \frac{\sigma}{\sqrt{\frac{\left(\sum_{j=1}^{n} {X_j}^2\right)}{n}}} \quad (7)$$

Para una mejor comprensión de las gráficas y observar el comportamiento de los valores en las series de tiempo, se implementó de forma complementaria el método de regresión lineal, junto con el cálculo de los coeficientes de determinación y correlación, para ofrecer una cuantificación de la relación entre las variables de corriente y voltaje.

La ecuación de regresión lineal permite predecir el comportamiento de una variable con respecto a otra y establecer de esta forma una relación causal entre ellas. En este caso se realizó el ajuste de voltaje y corriente con respecto al tiempo para observar su tendencia a lo largo de la medición. La Ecuación (8) presenta el valor de la pendiente (m) y la Ecuación (9) la de ordenada al origen (b).

$$m = \frac{(n\sum X_j Y_j) - \left(\sum X_j \sum Y_j\right)}{(n\sum {X_j}^2) - \left(\sum X_j\right)^2} \quad (8)$$

$$b = \frac{\sum Y_j - \mu \sum X_j}{n} \quad (9)$$

Donde el valor de X_j hace referencia al tiempo, Y_j al valor de los datos de corriente o voltaje según corresponda y μ a su media respectiva.

Los valores hallados de m y b fueron sustituidos en la Ecuación

(10) para obtener el modelo de regresión lineal.

$$y = mx + b \quad (10)$$

El Coeficiente de correlación (r) indica la medida de intensidad de relación entre dos variables, el cual toma en consideración el valor de la media y la separación individual de cada dato puntual. Con base en lo anterior, fue posible determinar el ajuste de las variables

con el modelo lineal planteado de tiempo-voltaje y tiempo-corriente. La Ecuación (11) presenta la fórmula para determinar el coeficiente de correlación.

$$r = \frac{\sum(X_j - \bar{X}_j)(Y_j - \bar{Y}_j)}{\sqrt{\sum(X_j - \bar{X}_j)^2 \sum(Y_j - \bar{Y}_j)^2}} \quad (11)$$

En la formula se establece el valor de los datos de tiempo X_j, la media del tiempo $\overline{X_j}$, los datos de voltaje o corriente representados por Y_j y su media correspondiente como $\overline{Y_j}$.

El coeficiente de determinación ($\boldsymbol{r^2}$) es el cuadrado del coeficiente de correlación e indica la proporción en que una variable puede ser predicha con respecto a otra, es decir, el cambio en una de ellas hará posible pronosticar los posibles cambios en la otra. El coeficiente de determinación cuantifica la discrepancia entre los valores esperados y los que se adaptan al modelo. La fórmula de trabajo utilizada en el código se encuentra descrita en la Ecuación (12).

$$r^2 = \left(\frac{\sum(X_j - \bar{X}_j)(Y_j - \bar{Y}_j)}{\sqrt{\sum(X_j - \bar{X}_j)^2 \sum(Y_j - \bar{Y}_j)^2}} \right)^2 \quad (12)$$

3.2. Diseño de la Interfaz Gráfica de Usuario con *App Designer*

Como se mencionó en el capítulo anterior, *App Designer* del entorno Matlab® fue la plataforma para realizar el diseño y programación de la GUI. A continuación, se describirá brevemente algunas características del entorno.

El inicio de la aplicación puede realizarse de dos formas: tecleando appdesigner desde el *Command Window* de Matlab® o seleccionar con el cursor la pestaña *Home* y dar clic en *new* seguido por *app*, tal como se muestra en la Figura 3.1.

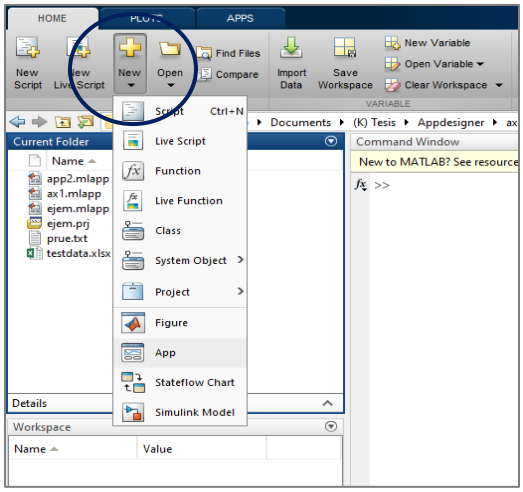


Figura 3.1. Ruta de acceso al entorno App Designer en Matlab®.

App Designer integra las dos tareas principales de la creación de aplicaciones que son: diseñar los componentes visuales de una Interfaz Gráfica de Usuario y programar el comportamiento de la aplicación. Por consiguiente, el entorno de trabajo se divide en dos ventanas: diseño y código.

3.2.1. Ventana de Diseño

La ventana de Diseño (La Figura 3.2 muestra su distribución por defecto) está conformada por los siguientes paneles de trabajo: Librería de componentes, Barra de herramientas, Navegador de componentes y Editor de diseño. Los cuáles serán descritos en los siguientes incisos:

a) Editor de diseño. Es el lienzo sobre el cual se distribuirán los diversos elementos o componentes de control que podrá observar el usuario en la interfaz.

b) Librería de componentes. En este espacio se agrupan los componentes disponibles para el diseño de la aplicación. Para ser utilizados se requiere seleccionarlos y arrastrarlos al Editor de diseño. Los diferentes componentes están clasificados en cuatro tipos:

- Comunes

- Contenedores
- Herramientas de figura
- Instrumentación

c) Navegador de componentes. Al seleccionar cada componente, se despliega una ventana con las características y opciones de cada uno, las cuales pueden ser modificadas de acuerdo con los requerimientos del usuario.

d) Barra de herramientas. Se compone de dos pestañas en las que se activan dos tipos de herramientas, las primeras denominadas de Diseño y las segundas de Lienzo. En la Tabla 3.1 y Tabla 3.2 se agrupan y describen brevemente.

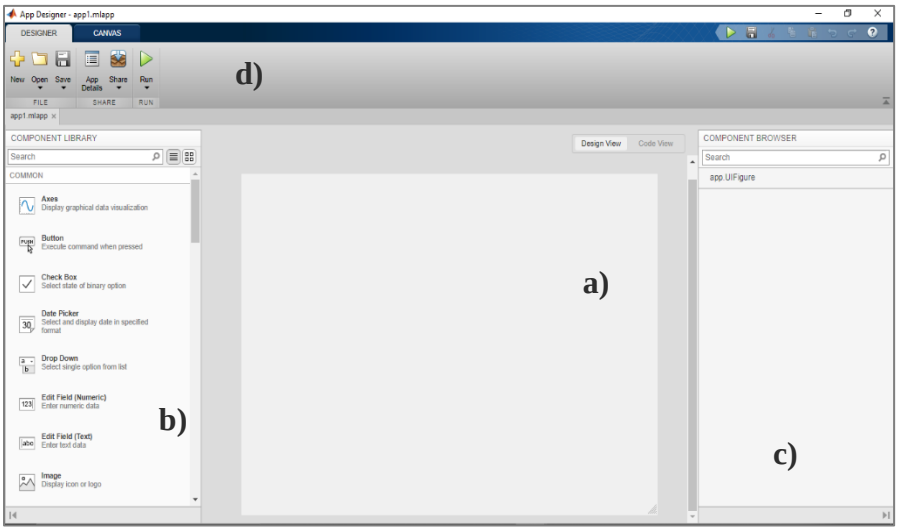


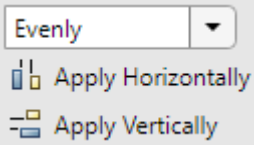
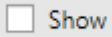
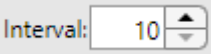
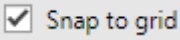
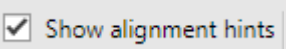
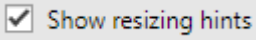
Figura 3.2. Ventana de Diseño App Designer y paneles de trabajo a) Editor de diseño, b) Librería de componentes, c) Navegador de componentes, d) Barra de herramientas.

Tabla 3.1 Barra de herramientas de la Ventana de Diseño.

Elemento	Nombre	Descripción
	***New***	Crea un nuevo archivo de *App Designer*. Despliega una ventana para seleccionar si se desea un archivo en blanco o utilizar alguno de los modelos base.
	***Open***	Despliega una ventana para buscar y abrir un archivo de *App Designer* existente.
	***Save***	Guarda los cambios efectuados en la aplicación.
	***App Details***	Permite escribir detalles de la aplicación.
	***Compare***	Compara las diferencias en código y elementos entre dos archivos existentes.
	***Share***	Compila la aplicación para que pueda ser compartida.
	***Run***	Ejecuta la aplicación y guarda los cambios realizados.

Tabla 3.2. Barra de herramientas del panel de Lienzo de la Ventana de Diseño.

Elemento	Nombre	Descripción
	***Save***	Guarda la aplicación y cambios en ella.
	***Convert***	Crea un ajuste automático de visualización de la aplicación en forma de paneles, los cuales ajustan su tamaño en función del espacio disponible en pantalla. Función introducida a partir la versión R2020b de *App Designer*.
	***Align***	Permite alinear los componentes seleccionados.

	***Same Size***	Iguala las proporciones entre los tamaños de los objetos seleccionados.
	***Grouping***	Agrupa los elementos seleccionados.
	***Reorder***	Da la opción de fijar o hacer móviles a los objetos del lienzo. Función introducida a partir la versión R2020b de *App Designer*.
	***Space***	Permite introducir espacio o distancia entre los componentes.
	***Show grid***	Activa una cuadricula en el espacio de trabajo para tenerla como referencia al alinear componentes.
	***Interval***	Cambia el tamaño de la cuadrícula.
	***Snap to grip***	Ajusta la posición de los componentes en la cuadrícula.
	***Show alignment hints***	Muestra líneas guía para alinear componentes.
	***Show resizing hints***	Permite visualizar líneas de apoyo para el ajuste de cambio en tamaño de los elementos.
	***Zoom***	Realiza un aumento o disminución del ángulo de visión de los elementos.
	***Run***	Ejecuta la aplicación. Al activar el botón los cambios realizados se guardarán de forma automática.

3.2.2. Ventana de Código

En la Ventana de Código se escriben las funciones de comandos y el llamado de los *Callbacks* asociados a los componentes agregados en la Ventana de Diseño. La Figura 3.3Figura 3.3 muestra la Ventana de código y sus paneles de trabajo respectivos, los cuales son descritos en los siguientes incisos:

a) Editor de código. Es el espacio que contiene el código de trabajo. Es de mencionar que parte de éste se genera automáticamente al colocar objetos en el Editor de diseño o al asignarles un *Callback*. En el Editor de código es posible usar las funciones permitidas dentro de la sintaxis de Matlab®.

b) Navegador de código. Es el espacio donde se enlistan los *Callbacks* utilizados, las funciones y propiedades de los elementos de la aplicación.

c) Diseño de la aplicación. Presenta una vista previa de la distribución de los componentes en la interfaz o aplicación.

d) Propiedades del botón. En esta sección se presentan las características a modificar de los elementos de la aplicación. Las propiedades pueden ser editadas para un uso inicial predefinido y posteriormente ser manipuladas mediante código desde el editor.

e) Barra de herramientas. Presenta aspectos comunes a la Ventana de Diseño. Sin embargo, cuenta con herramientas propias como la creación rápida de *Callbacks* o de funciones de utilidad, así como la creación o eliminación de comentarios. En la Tabla 3.3 se presentan de forma resumida sus elementos.

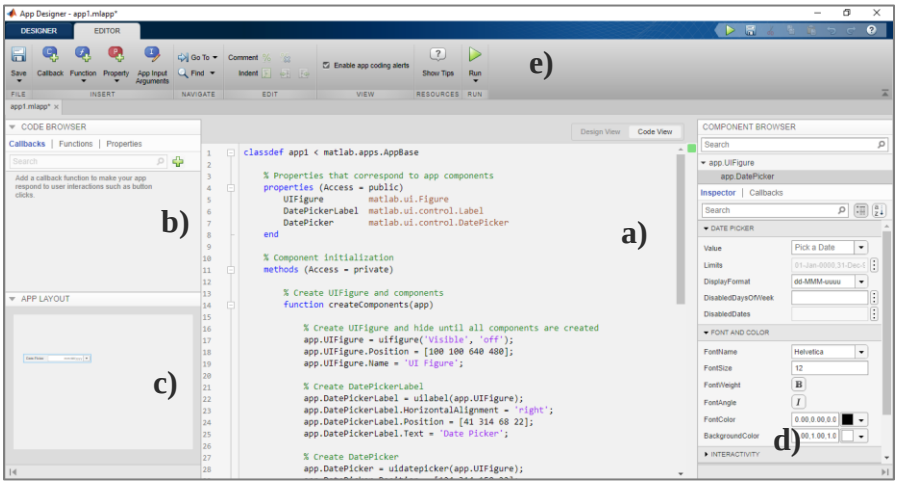
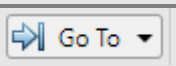
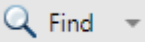
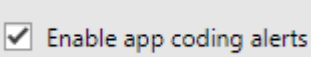


Figura 3.3. Ventana de Código de App Designer y paneles de trabajo a) Editor de código, b) Navegador de código, c) Diseño de la aplicación, d) Propiedades del botón, e) Barra de herramientas.

Tabla 3.3. Barra de herramientas de la Ventana de Código.

Elemento	Nombre	Descripción
	***Save***	Guarda la aplicación y cambios en ella.
	***Compare***	Realiza una comparación entre dos códigos existentes.
	***Callback***	Permite la creación automática de un *Callback* sobre un objeto seleccionado.
	***Function***	Crea una nueva función de utilidad.
	***Property***	Crea una propiedad para el objeto seleccionado.
	***App Input Arguments***	Permite a la aplicación recibir argumentos de entrada.
	***Go to***	Localiza un elemento o función creada.
	***Find***	Busca en el código las palabras que sean ingresadas.
	***Comment***	Al seleccionar líneas de código las convierte a comentario o los elimina para ser líneas de código activo.
	***Indent***	Añade o retira sangría a las líneas de código seleccionadas.
	***Enable app coding alerts***	Activa o desactiva los mensajes de alerta o error en el código.
	***Zoom***	Aumenta o disminuye el ángulo de visión del código.
	***Show tips***	Permite obtener consejos o ayuda en la vista del código.
	***Run***	Ejecuta la aplicación y guarda los cambios realizados.

Una sección especial es el denominado Navegador de componentes, mostrado en la Figura 3.4, el cual forma parte del panel de trabajo tanto en la Ventana de Diseño como en la de Código. El Navegador de componentes enlista los componentes de la interfaz. El orden por defecto señala el nombre del objeto creado antecedido de la palabra "app."; sin embargo, éste puede ser modificado para hacer más sencillo su reconocimiento.

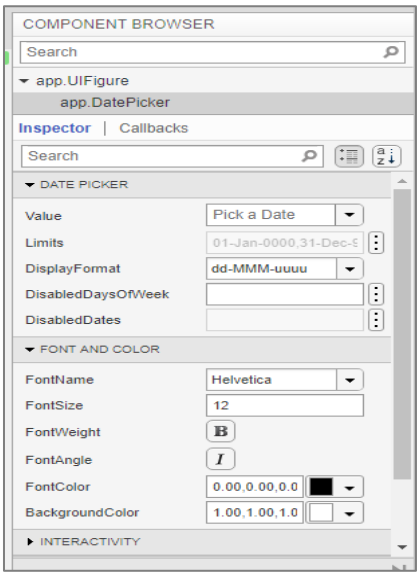


Figura 3.4. Navegador de componentes, espacio común entre las ventanas de Código y Diseño, muestra propiedades de los objetos seleccionados.

3.2.3. Estructura del código

En el Editor de código de la Ventana de Código se muestra el *script* de la aplicación o interfaz. Una ventaja que ofrece *App Designer* es que gran parte del código es generado automáticamente al arrastrar los componentes deseados en el Editor de diseño, al añadir *Callbacks* o al cambiar valores predeterminados en *Button Properties*; lo cual permite una programación más sencilla. Una importante ayuda adicional es que parte del código que resulta esencial para la ejecución de los *Callbacks* y funciones estará protegido para evitar ser modificado o borrado en la edición el código.

App Designer hace una diferenciación entre cuatro tipos de funciones básicas:

a) Funciones *Start Up*

Son las funciones que se ejecutan al abrir la aplicación por primera vez y son las definidas en los objetos por defecto para ser mostradas al usuario; un ejemplo son los ejes de los gráficos. Pueden ser modificadas al cambiar sus propiedades o al dar la opción al usuario con controles adicionales.

b)* Funciones de interacción con objetos o *Callbacks

Es posible definir a los *Callbacks* como funciones predefinidas en respuesta a una acción realizada por el usuario en los objetos de la aplicación, como por ejemplo activar un Switch (Cid Espinosa, 2018).

En *App Designer* el uso de los *Callbacks* son el eje principal de la funcionalidad de la aplicación y la creación de ellos es bastante sencilla. Se debe seleccionar al objeto o acceder al Navegador de componentes, en seguida dar *click* derecho en el mouse sobre el nombre del objeto deseado y al abrirse una pestaña de opciones seleccionar *Callbacks* y posteriormente *Add Callback*. La Figura 3.5 muestra la creación de un *Callback* desde el Navegador de componentes.

Al crearse un *Callback*, *App Designer* redirigirá automáticamente a la Ventana de Código en el espacio definido para la nueva función.

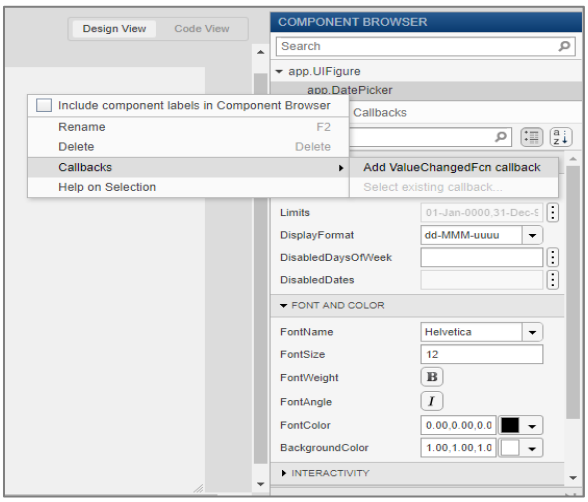


Figura 3.5. Creación de un Callback desde el Navegador de componentes en la Ventana de Diseño

c) Funciones privadas

Son las utilizadas dentro de la aplicación para compartir instrucciones entre *Callbacks*. Su ejecución ocurre únicamente dentro de la aplicación y su principal utilidad es evitar la repetición de líneas de código.

d) Funciones de utilidad pública

Son funciones especiales que permiten compartir utilidades con otras aplicaciones o con otros dispositivos u periféricos fuera de la interfaz.

El comportamiento de las funciones es influenciado por su ubicación en la estructura del código, en *App Designer* se pueden observar tres partes o secciones, las cuales son explicadas en los siguientes incisos:

a) Sección de propiedades de la aplicación

En esta sección se establecen las características de la aplicación y de los componentes, así como también la relación entre los objetos y sus propiedades, como se puede observar en la Figura 3.6.

```
classdef app1 < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure         matlab.ui.Figure
        DatePickerLabel  matlab.ui.control.Label
        DatePicker       matlab.ui.control.DatePicker
        Button           matlab.ui.control.Button
        UIAxes           matlab.ui.control.UIAxes
    end
```

Figura 3.6. Sección predeterminada de código de App Designer para propiedades de aplicación.

b) Sección de funciones

Esta sección es el cuerpo central del código donde se añaden las funciones de utilidad de la aplicación, comúnmente relacionadas a los *Callback* de los objetos de control. Un ejemplo es mostrado en la Figura 3.7.

```
    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: Button
        function ButtonPushed(app, event)
            for i = 1:20
           h=[1,2,3]
           namestr = ['s' num2str(i) '=h'];
           eval(namestr);
           endfor i = 1:20
           h=[1,2,3]
           namestr = ['s' num2str(i) '=h'];
           eval(namestr);
            end

        end
    end
```

Figura 3.7. Sección de funciones del código de App Designer.

c) Sección de inicialización de la aplicación

En esta zona del código se declaran las propiedades u opciones predefinidas de los objetos de la interfaz, las cuales son ejecutadas de forma automática al abrir la aplicación. La Figura 3.8 muestra un ejemplo de código de la sección de inicialización.

```
    % Component initialization
    methods (Access = private)

        % Create UIFigure and components
        function createComponents(app)

            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Position = [100 100 640 480];
            app.UIFigure.Name = 'UI Figure';

            % Create DatePickerLabel
            app.DatePickerLabel = uilabel(app.UIFigure);
            app.DatePickerLabel.HorizontalAlignment = 'right';
            app.DatePickerLabel.Position = [41 314 68 22];
            app.DatePickerLabel.Text = 'Date Picker';

            % Create DatePicker
            app.DatePicker = uidatepicker(app.UIFigure);
            app.DatePicker.Position = [124 314 150 22];

            % Create Button
            app.Button = uibutton(app.UIFigure, 'push');
            app.Button.ButtonPushedFcn = createCallbackFcn(app, @ButtonPushed, true);
            app.Button.Position = [41 266 100 22];
```

Figura 3.8. Sección del código de App Designer dedicada a la inicialización de componentes.

Detección y corrección de errores en código

App Designer ofrece dos herramientas para detectar errores de sintaxis en el código:

- Alertas de codificación
- Mensajes de *Code Analyzer*

Alertas de codificación

Al escribir el código en el espacio de edición, *App Designer* detecta errores de sintaxis y emite mensajes de error combinados con un icono de advertencia. El mensaje incluirá una breve descripción del tipo de error sobre la línea de código que contenga el problema. En la Figura 3.9 se puede observar un ejemplo de alerta de codificación.

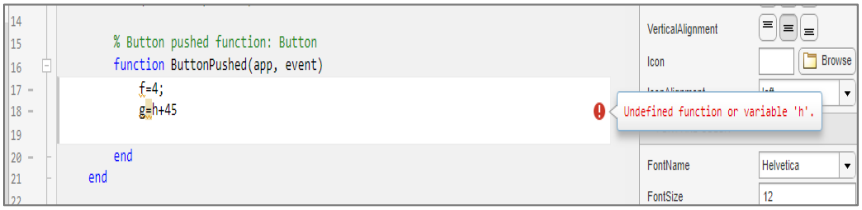


Figura 3.9. Alerta de codificación propio de App Designer.

Mensaje de *Code Analyzer*

Es la herramienta de Matlab® para señalar al usuario los errores en la sintaxis del código; *Code Analyzer* ocupa el color de subrayado naranja o rojo, para indicar advertencia o error respectivamente.

No todas las características del *Code Analyzer* están disponibles en el editor de *App Designer*, por lo que algunos de los mensajes de error pueden ser más reducidos y no contar con una descripción detallada del tipo de fallo causado. En la Figura 3.10 se muestra un ejemplo de mensaje de error detectado por el *Code Analyzer*.

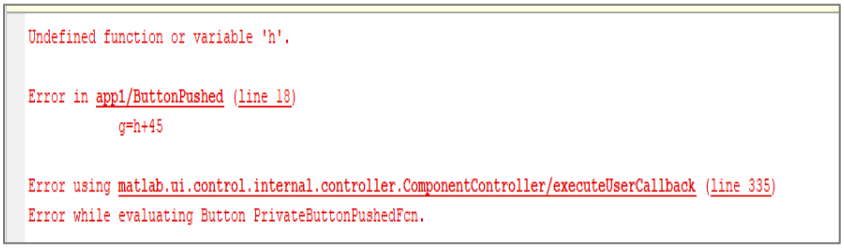


Figura 3.10. Mensaje de error mostrado con Code Analyzer de Matlab

3.2.4. Componentes

La Tabla 3.4, Tabla 3.5, Tabla 3.6 y Tabla 3.7 ofrecen una descripción de los componentes que ofrece *App Designer* para la creación de las aplicaciones y de interfaces.

Tabla 3.4. Tabla de componentes de App Designer para la representación de datos.

Elemento	Nombre	Descripción
	***Axes***	Es el objeto empleado para la representación de datos en diversos diagramas. Permite gráficos de dispersión, histogramas, circulares, etc. Da la opción de manejar escalas, marcadores, mallado, colores, así como la presentación de imágenes.
	***Table***	Permite la presentación de datos de forma ordenada como tablas e igualmente permite obtener y almacenar datos desde archivos como .xlsx, .txt, .csv. Los datos almacenados en una tabla pueden ser manipulados (mediante funciones o *Callbacks*) y presentarse nuevamente en un objeto *Table* o *Axes*.
	***Image***	Objeto que permite la visualización de imágenes estáticas o modificadas por el código de la aplicación.
   	***Gauge***	Los medidores resultan particularmente útiles para representar variables físicas como pueden ser la presión o temperatura, ya que permiten al usuario evaluar la reproductibilidad y repetitividad de las variables con mayor estimulo visual. En función de la presentación de los datos deseada conviene hacer la selección entre un medidor completo, de noventa grados, lineal o semicircular.
	***Lamp***	Es el objeto predefinido para señalar el estado del proceso, al utilizar un *Callback* en ellos es posible asociar un cambio de color que muestre la modificación de la ejecución de las tareas de la aplicación.
	***Label***	Permite nombrar o denominar a los elementos dentro de la aplicación y presentarlos al usuario de la aplicación.

Tabla 3.5. Componentes de App Designer para la entrada de datos.

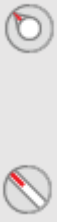
Elemento	Nombre	Descripción
	***Edit Field Numeric***	Es el objeto que permite al usuario introducir un valor numérico por medio del teclado. Sus propiedades pueden ser modificadas dentro del Navegador de componentes, lo que permite establecer rangos, formato de visualización, valor inicial o bloqueo de edición.
	***Spinner***	Permite al usuario introducir por medio del ratón cantidades precisas de datos numéricos dentro de intervalos predefinidos y con un rango establecido máximo. Las propiedades de salto de intervalo deben ser definidas previamente en el Navegador de componentes.
	***Slider***	Representa visualmente todo el rango de valores controlables por el usuario, está pensado para ser usado en aplicaciones de cambio gradual mediante el uso de *Callbacks*.
	***Knob***	Presenta el rango máximo de datos y su desplazamiento a través de diversos valores intermedios. Resulta principalmente útil para definir puntos intermedios o categorías de valores asociados a una acción particular de la aplicación.
	***Edit Field Text***	Permite introducir líneas de texto a la interfaz. Está creado para cadenas cortas de caracteres.
	***HTML***	Crea un componente que permite incrustar contenido HTML, JavaScript ® o CSS, en la aplicación. Función permitida a partir de la versión R2021a de *App Designer*.
	***Hyperlink***	Permite asociar un hipervínculo a un objeto en particular de la interfaz para acceder a su contenido.
	***Text Area***	Es un objeto que hace posible al usuario introducir varias líneas de texto y al hacerlo se muestra una barra de desplazamiento para la visualización completa.

Tabla 3.6. Componentes de App Designer para la ejecución de acciones.

Elemento	Nombre	Descripción
	***Button***	Es el objeto básico para la ejecución de acciones. Normalmente su *Callback* está asociado a la realización de tareas particulares al ser pulsados por el usuario.
	***Checkbox***	Elemento de interacción que permite al usuario hacer la selección de una opción al marcar o desmarcar una casilla.
	***State Button***	Es utilizado principalmente para representar la ejecución de la operación. Al ser presionado dará la impresión de comenzar la tarea y servirá de estímulo visual al usuario.
	***Switch, Toggle Switch, Rocker Switch***	Sirven para controlar dos estados mutuamente excluyentes por el usuario al cambiar su posición.
	***Drop Down, List Box, Radio Button***	Selectores que tienen la particularidad de permitir la ejecución de tareas al elegir entre múltiples opciones (excluyentes entre ellas) al poder asociar un *Callback* independiente a cada una de ellas.
	***Menu Bar***	Permite añadir barras de menús en la parte superior de la interfaz o aplicación, con lo cual se reduce considerablemente el espacio dentro del área de diseño de la interfaz al poder añadir opciones al usuario y asociarlas a un *Callback*.
	***Tool Bar***	Crea en la parte superior de la aplicación una fila de iconos que pueden ser relacionados a una acción en particular mediante un *Callback*.
	***Context Menu***	Al unir este objeto a otro elemento de *App Designer* es posible acceder a un menú adicional de comandos o acciones al posicionar el cursor y hacer *click* derecho.

Tabla 3.7. Componentes de App Designer para la organización de elementos.

Elemento	Nombre	Descripción
	***Panel***	Tienen como función principal contener a otros objetos o servir de organizadores de los que tengan una función similar.
	***Grid Layout***	Objeto creado principalmente para desplegar gráficos o figuras con una mejor disposición que el objeto *Axes*.
	***Tab Group***	Es usado para crear pestañas organizadoras que contengan objetos por separado y que sean visibles únicamente los necesitados, lo cual ayuda a reducir el espacio de la aplicación.
	***Tree***	Permite formar una estructura jerárquica de directorios u opciones.

3.2.5. Compilador

Las interfaces o aplicaciones creadas en *App Designer* pueden ser guardadas y posteriormente ejecutadas por cualquier usuario de Matlab®, pero, al mismo tiempo ofrecen la posibilidad de utilizar *Matlab Compiler*, una herramienta que permite compartir la interfaz como una aplicación autónoma que no requiera del entorno Matlab®.

El procedimiento de *Matlab Compiler* tiene la siguiente secuencia de pasos:

1) Ir a la barra de herramientas de diseño y dar *click* en el botón Compartir o *Share*.
2) Al abrirse una nueva ventana se selecciona el archivo con extensión ".mlapp" que se requiere compilar (normalmente el de trabajo actual es señalado como predeterminado), y se da click a *Standalone Application*.
3) Se selecciona la forma en que se obtendrán las librerías de instalación, denominados *Matlab Compiler Runtime*, existen dos tipos:
   a) *Runtime downloaded from the web*, las librerías se descargan de internet.
   b) *Runtime included in package*, las librerías se almacenan en el ejecutable.
4) Incluir datos para personalizar la aplicación

Las opciones que ofrece *App Designer* incluyen el colocar una imagen y nombre al ejecutable, establecer una imagen que se muestre durante el proceso de instalación, colocar los datos del creador de programa, así como los datos de contacto, descripción del programa y su número de versión. A partir de la versión R2020b de *App Designer* es posible cambiar el icono de la aplicación. En la Figura 3.11 se presenta un ejemplo de la ventana de personalización.

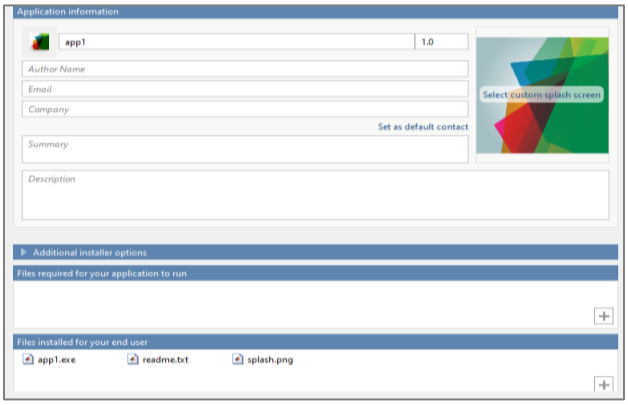


Figura 3.11. Ventana de personalización al compilar la aplicación de App Designer.

5) Añadir archivos necesarios para la aplicación.

   En caso de que la interfaz requiera archivos extra para su funcionamiento esta opción permite incluirlos en el archivo principal y que puedan ser empaquetados. Por defecto se incluye el ejecutable de la aplicación y un archivo de texto con instrucciones básicas para instalar y configurar el archivo compilador de Matlab®.

6) Dar *click* en el botón *Package* y seleccionar la ubicación para guardar el archivo.
   Una vez terminado el proceso de compilación se crearan las siguientes carpetas de archivos:

* *for_redistribution*, dirigido a no usuarios de Matlab®, el archivo .exe iniciara el proceso de instalación y los archivos necesarios para el funcionamiento autónomo.
* *for_redistribution_files_only*, está orientado para usuarios del entorno Matlab®, el archivo .exe no requiere de un archivo instalador para su ejecución.

CAPÍTULO IV

4. RESULTADOS Y DISCUSIONES

Descritas las operaciones para el cálculo de Ruido Electroquímico y los elementos disponibles para la creación de la Interfaz Gráfica de Usuario en Matlab®, se procede en el presente capitulo a explicar el desarrollo de la interfaz y los resultados obtenidos con su implementación.

Para facilitar la comprensión de los procesos efectuados por la interfaz se presenta su esquema estructural por medio de un diagrama de flujo. La interfaz tiene por objetivo ser amigable al usuario, por ello se plantean pocos pasos para obtener los datos y guardar los resultados. En la Figura 4.1 se muestra el diagrama de flujo de la aplicación y se esquematizan los procesos generales que serán efectuados.

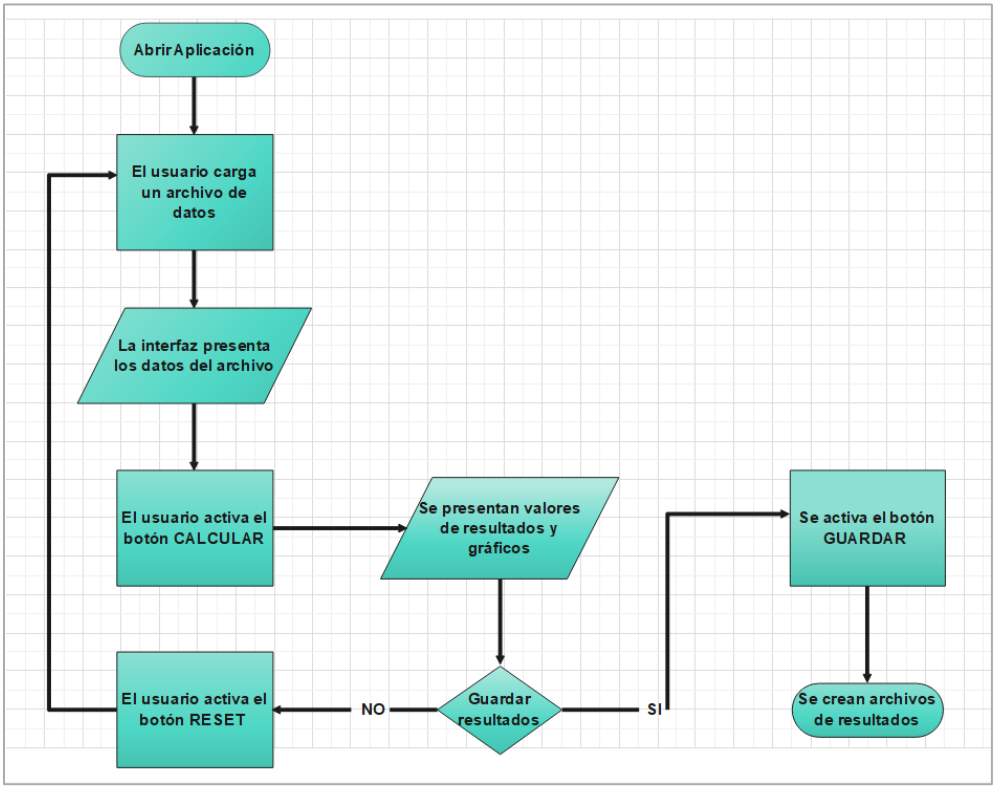


Figura 4.1. Diagrama de flujo de la aplicación.

De acuerdo con el diagrama de flujo mostrado, la interfaz depende esencialmente de cinco funciones de utilidad:

- Carga de datos
- Cálculo de operaciones
- Visualización de resultados
- Limpieza de resultados
- Guardado de resultados

4.1. Carga de datos

Es la función básica del código, ya que permitir obtener la base de datos con la que inicia el funcionamiento de la interfaz. Las extensiones de archivo permitidas por la interfaz son:

- Archivo tipo texto (.txt)
- Archivo de Valores Separados por Comas (.csv, por sus siglas en ingles de *comma separated values*)
- Archivo Excel (.xlsx)

Las condiciones de la interfaz para cargar y utilizar los datos son tres: la primera, las columnas de los datos deben ser ordenadas en tiempo, voltaje y corriente. La segunda condición, es que no haya espacios vacíos entre los datos o diferentes a valores numéricos. Como última condición, se deben retirar los encabezados de las columnas de datos.

La interfaz ofrece dos opciones para cargar los archivos: la primera desde el Menú superior, al pulsar en la opción “Cargar datos” →” Cargar Archivo”. La segunda opción es pulsar en el botón “CARGAR ARCHIVO” ubicado en la parte inferior derecha de la interfaz. La cantidad de datos de entrada predefinidos es de 1024 (número de valores ofrecidos en las bases de datos de prueba). Sin embargo, se ofrece la posibilidad de reducir o aumentar la cantidad de datos mediante un elemento de control tipo *Spinner*. La Figura 4.2 señala la posición de las opciones para la carga de archivos y de control de número de datos.

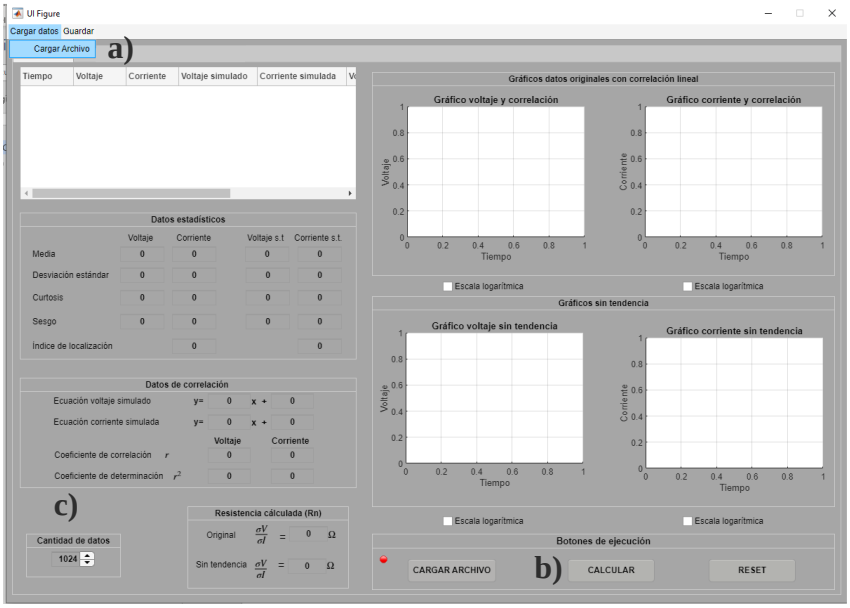


Figura 4.2. Opciones para cargar archivo de datos en la interfaz a) Menú superior; b) Botón de acción. c) Control de cantidad de datos.

El usuario al elegir alguna de las opciones para cargar archivos activará un *Callback*, el cual hace uso de la función *uigetfile***,** la cual permite abrir una ventana del navegador de *Windows* para buscar y seleccionar el archivo deseado. En la Figura 4.3 se observa la ventana de búsqueda y selección de archivo.

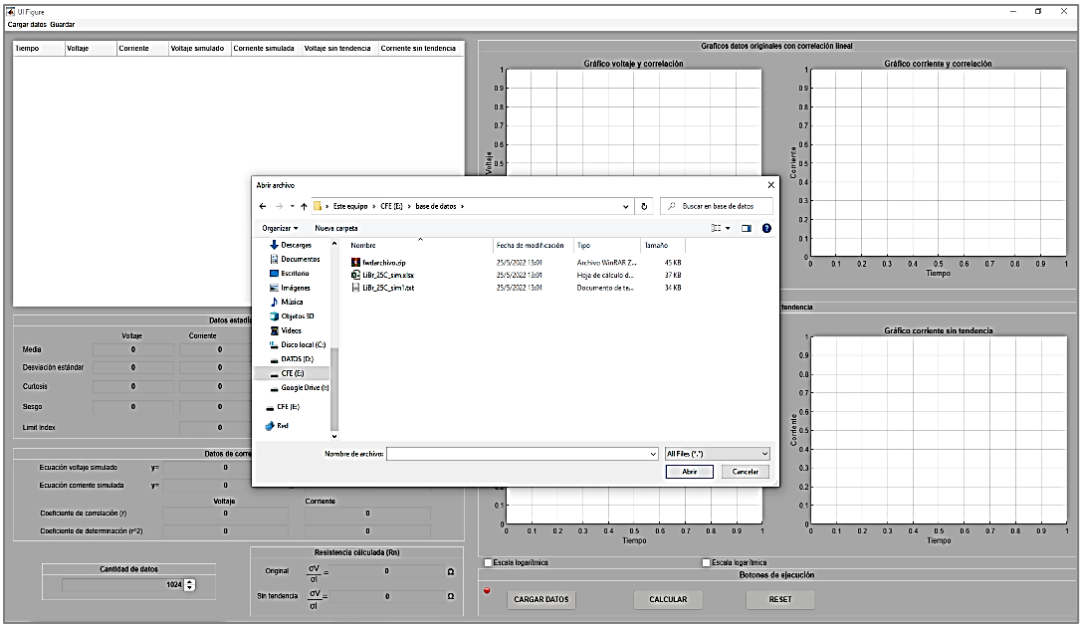


Figura 4.3. Interfaz con despliegue de ventana para seleccionar archivo de base de datos

La función *uigetfile* permite obtener y guardar como variables el nombre y la ruta del archivo que será abierto.

```
[nombre,ruta]=uigetfile({'*.*'},'Abrir archivo');
if nombre== 0
return;
else
```

Para almacenar la información de la ubicación del documento contenedor de la base de datos, las variables de nombre y ruta de archivo serán concatenadas por medio de la función *strcat*. Posteriormente la información será guardada en un objeto tipo tabla de *App Designer* al usar la función *readtable*.

```
datos= strcat(ruta,nombre);
app.Tabladat.Data=readtable(datos);
```

La tabla de datos creada (denominada como *Tabladat*) será de uso exclusivo del programa y no será mostrada al usuario. Para lograr esta característica, desde el Navegador de componentes de la Ventana de Diseño se selecciona la casilla *Enable* en la opción de Interactividad del componente, lo cual es mostrado en la Figura 4.4.

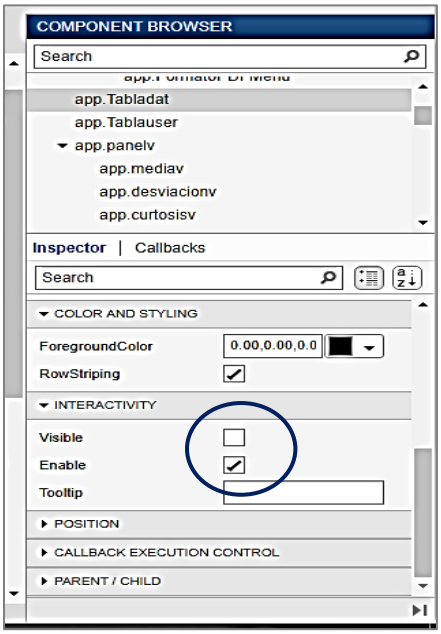


Figura 4.4. Opción de inhabilitado de visualización de objeto desde el Navegador de componentes.

Al no poder visualizar de forma inmediata la carga de datos en la tabla de valores, se propone, como estímulo visual del tiempo de proceso, un objeto *Lamp* (led) en el área de Botones de ejecución. El led en su estado inicial es color rojo y cambia a color verde una vez sea finalizado el proceso de carga, este cambio puede observase en la Figura 4.5.

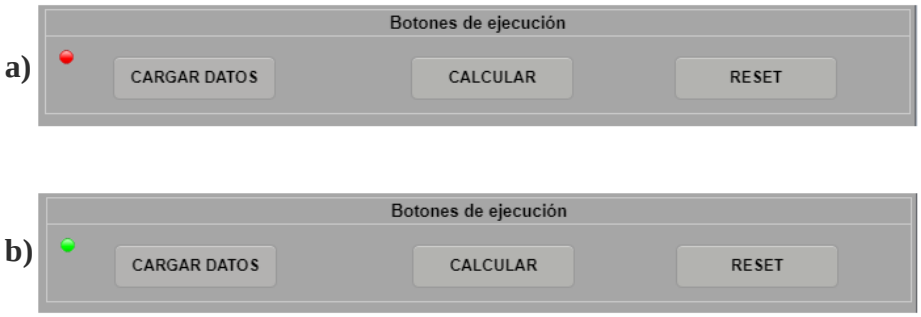


Figura 4.5. Cambio realizado en el objeto Lamp, a) Color rojo, sin datos cargados, b) Color verde, con base de datos.

Para conseguir la transición de color en el objeto led fue definido un estado inicial desde el Navegador de componentes, y posteriormente en el código se estableció una condición para su cambio. En este caso la condición es la existencia de datos en una de las columnas (la respectiva al tiempo) de la tabla de valores, para lograrlo se emplea a la función *isempty* en combinación con un ciclo if para su control.

```
ind0=(app.Tabladat.Data(:,1));
ind1=table2array(ind0);
if isempty(ind1)
app.lampcar.Color='red';
else
app.lampcar.Color='green';
end
end
```

4.2. Procesamiento de datos

Obtenida la base de datos, el usuario puede utilizar el botón “CALCULAR”, acción que activara el *Callback* correspondiente al cuerpo principal del programa en el que se realizan las siguientes acciones:

1) Preparación de datos

La preparación de datos hace referencia al tratamiento de éstos como variables u objetos de *App Designer* en caso de ser necesario.

El tratamiento de los datos como variable es necesario en muchas ocasiones en *App Designer* dado que algunas funciones pueden señalar error al no reconocer a los objetos como entradas válidas.

```
vmat=app.Tabladat.Data;
```

El tratamiento de datos ofrece de igual manera la posibilidad de realizar operaciones con mayor facilidad, como es el caso de la conversión de tipos de dato. Ejemplo de lo anterior es la función *table2array* que permite pasar de operar datos tipo tabla a datos tipo arreglo. Este nuevo formato permite trabajar de forma separada las columnas de una tabla y posteriormente guardarlas como una variable.

```
vmat1=table2array(vmat);

% (tiempo)
ti=vmat1(1:ntot,1);
% (voltaje)
vo=vmat1(1:ntot,2);
% (corriente)
co=vmat1(1:ntot,3);
```

Al contar con las variables definidas y separadas es posible obtener de forma más sencilla el resultado de las operaciones estadísticas.

2) Cálculo de resultados

En el código fue empleado un ciclo *for* para iniciar nuevas variables y guardar en ellas los datos de voltaje (*ntv*) y corriente (*ntc*) con la remoción de tendencia, para ello se hace uso de la Ecuación (2).

```
%% Retirar la tendencia al voltaje y corriente
for ii=1:1:(ntot-1)
ntv(ii+1)=volt1(ii+1)-volt1(ii);
```

```
ntc(ii+1)=corr1(ii+1)-corr1(ii);
end
```

En el código los valores de la media, obtenidos con el uso de la Ecuación (3) son almacenados en las variables *medvlib* (voltaje), *medclib* (corriente) y sus respectivas versiones sin tendencia *medvnt medcnt*.

```
% Media voltaje
medvlib=sum(volt1)/nf;
medvnt=sum(ntv)/nf;
% Media corriente
medclib=sum(corr1)/nf;
medcnt=sum(ntc)/nf;
```

En el código empleamos las variables *desvvlib* (voltaje) y *desvclib* (corriente) para almacenar los valores de la desviación obtenidos con base en la Ecuación (4).

```
% Desviación estándar voltaje
desvvlib=sqrt((sum((volt1-medvlib).^2))/nf);
% Desviación voltaje sin tendencia
desvnt=sqrt((sum((ntv-medvnt).^2))/nf);
% Desviación estándar corriente
desvclib=sqrt((sum((corr1-medclib).^2))/nf);
% Desviación corriente sin tendencia
descnt=sqrt((sum((ntc-medcnt).^2))/nf);
```

Dentro del código los valores correspondientes a la curtosis, definida por la Ecuación (5), son asignados a las variables *curtvlib* y *curtclib*.

```
% Curtosis voltaje
curtvlib=(nf*(sum((volt1-medvlib).^4)))/((sum((volt1-medvlib).^2))^2);
% Curtosis sin tendencia voltaje
curvnt=(nf*(sum((ntv-medvnt).^4)))/((sum((ntv-medvnt).^2))^2);
```

```
% Curtosis corriente
curtclib=(nf*(sum((corr1-medclib).^4)))/((sum((corr1-medclib).^2))^2);
% Curtosis sin tendencia corriente
curcnt=(nf*(sum((ntc-medcnt).^4)))/((sum((ntc-medcnt).^2))^2);
```

En el código el valor del sesgo, calculado usando la Ecuación (6) se guardará dentro de las variables *sesgovlib* (voltaje) y *sesgoclib* (corriente).

```
% Sesgo voltaje
sesgovlib=(sqrt(nf)*sum((volt1-medvlib).^3))/((sum((volt1-medvlib).^2))^(3/2));
% Sesgo voltaje sin tendencia
sesgovnt=(sqrt(nf)*sum((ntv-medvnt).^3))/((sum((ntv-medvnt).^2))^(3/2));
% Sesgo corriente
sesgoclib=(sqrt(nf)*sum((corr1-medclib).^3))/((sum((corr1-medclib).^2))^(3/2));
% Sesgo corriente sin tendencia
sesgocnt=(sqrt(nf)*sum((ntc-medcnt).^3))/((sum((ntc-medcnt).^2))^(3/2));
```

Con el uso de la Ecuación (7) se obtiene el valor para la variable denominada *liclib* que hace referencia al Índice de localización.

```
% Índice de localización voltaje
livlib=desvvlib/(sqrt((sum(corr1.^2))/nf));
% Índice de localización voltaje sin tendencia
lrmsvnt= desvnt/(sqrt((sum(ntv.^2))/nf);
% Índice de localización corriente
liclib= desvvlib/(sqrt((sum(corr1.^2))/nf));
% Índice de localización corriente sin tendencia
lrmscnt=descnt/(sqrt((sum(ntc.^2))/nf)/;
```

Los datos necesarios para el método de regresión lineal (pendiente m y ordenada al origen b) son calculados con las Ecuaciones (8) y (9), los resultado obtenidos se almacenan en las variables para voltaje (*mvlib,bvlib*) y corriente (*mclib,bclib*).

```
% Valor de m (pendiente) método de mínimos cuadrados (voltaje)
mvlib=((nf*(sum(tiemp1.*volt1)))-(sum(tiemp1)*sum(volt1)))/((nf*(sum(tiemp1.^2)))-(sum(tiemp1)^2));
% Valor de b (intersección) en voltaje
bvlib=((sum(volt1))-(mvlib*(sum(tiemp1))))/nf;
```

```
% Valor de m (pendiente) método de mínimos cuadrados (corriente)
mclib=((nf*(sum(tiemp1.*corr1)))-(sum(tiemp1)*sum(corr1)))/((nf*(sum(tiemp1.^2)))-(sum(tiemp1)^2));
% Valor de b (intersección) en corriente
bclib=((sum(corr1))-(mclib*(sum(tiemp1))))/nf;
```

Los resultados de la pendiente y ordenada en el origen son sustituidos en la Ecuación (10), con lo que se obtiene el modelo de regresión lineal para voltaje y corriente. Estos datos serán presentados posteriormente al usuario en la tabla de visualización.

```
% Voltaje simulado regresion
voltsimlib=(tiemp1*mvlib)+bvlib;
% Corriente simulado regresion
corrsimlib=(tiemp1*mclib)+bclib;
```

El coeficiente de correlación r es calculado en base a la Ecuación (11) y guardado en las variables rv para voltaje y rc para corriente.

```
% Coeficiente de correlación en voltaje
rv=(sum((tiemp1-medtlib).*(volt1-medvlib)))/(sqrt((sum((tiemp1-medtlib).^2))*(sum((volt1-medvlib).^2))));
% Coeficiente de correlación en corriente
rc=(sum((tiemp1-medtlib).*(corr1-medclib)))/(sqrt((sum((tiemp1-medtlib).^2))*(sum((corr1-medclib).^2))));
```

Obtenido el coeficiente de correlación el coeficiente de determinación es calculado siguiendo la Ecuación (12).

```
% Coeficiente de determinación (voltaje)
r2v=rv^2;
% Coeficiente de determinación (corriente)
r2c=rc^2;
```

Con los datos estadísticos calculados es posible utilizar la Ecuación (1) para obtener la Resistencia de Polarización al Ruido (R_n), la cual relaciona los valores de la desviación estándar de voltaje y corriente en analogía a la ley de Ohm. De esta forma se presenta a la variable *resistlib*.

```
% Resistencia de Polarización al Ruido
resistlib=desvvlib/desvclib;
```

4.3. Visualización de resultados

Obtenidos los resultados del cálculo de los parámetros estadísticos se procede a presentarlos al usuario. La visualización de los resultados en la interfaz es en tres formas:

1) Tabla de valores

Se utiliza un objeto *Table* (denominada *Tablauser)* y su propiedad *Data* para almacenar y mostrar los valores obtenidos en las operaciones. La tabla se compone de siete columnas, la primera de ellas para el tiempo, y las siguientes para los datos de voltaje y corriente, así como los respectivos valores simulados y con remoción de la tendencia.

```
app.Tablauser.Data(:,1)=(tiemp1);
app.Tablauser.Data(:,2)=(volt1);
app.Tablauser.Data(:,3)=(corr1);
app.Tablauser.Data(:,4)=(voltsimlib);
app.Tablauser.Data(:,5)=(corrsimlib);
app.Tablauser.Data(:,6)=(ntv);
```

```
app.Tablauser.Data(:,7)=(ntc);
```

Los valores de las columnas son guardados directamente desde las variables almacenadas. Los encabezados de las columnas son posibles de predefinir desde las opciones del Navegador de componentes, y, por lo tanto, no requieren manipulación adicional mediante código. Para realizar lo anterior, solo es necesario seleccionar el componente y posteriormente ir a la sección *ColumnName*, se deben introducir los nombres de cada columna separados por una coma simple. Un ejemplo del cambio de encabezado es mostrado en la Figura 4.6.

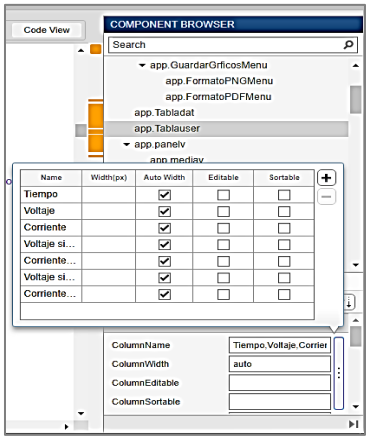


Figura 4.6. Añadir encabezados a columnas de objeto Table desde el Navegador de componentes.

Debido a la base de datos (regularmente de valores muy pequeños), los resultados de las operaciones se presentan en la tabla en formato *long***,** el cual permite una notación de hasta 15 dígitos decimales. En *App Designer* no es posible aplicar desde las opciones del Navegador de componentes el formato de visualización en celdas. Este cambio debe ser aplicado vía código usando la propiedad del objeto *Table* denominada *ColumnFormat*, y establecer por cada columna el formato deseado.

```
%%-------Formato a los datos de la tabla-------%%
app.Tablauser.ColumnFormat=({'long','long','long','long','long','long','long'});
```

La presentación de la tabla al inicio y después de la ejecución de la aplicación es mostrada en la Figura 4.7.

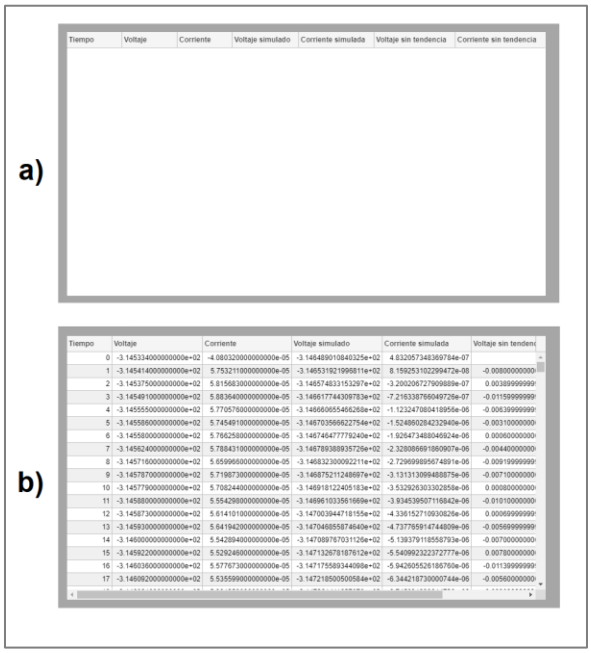


Figura 4.7. Tabla de visualización a) Sin valores al iniciar la aplicación; b) Presentación de los datos calculados al usuario.

2) Campos numéricos

Los campos numéricos son empleados para mostrar al usuario un dato en concreto de las operaciones. Como se puede observar en la Figura 4.8, en la interfaz éstos están agrupados y colocados inmediatamente después a la tabla de visualización. Dentro de *App Desiner* se utiliza al objeto *Edit File Numeric* y su propiedad *Value*, para almacenar el valor de la variable respectiva.

```
% Visualización de valores de voltaje
app.mediav.Value = medvlib;
app.desviacionv.Value = desvvlib;
app.curtosisv.Value = curtvlib;
app.sesgv.Value = sesgovlib;
```

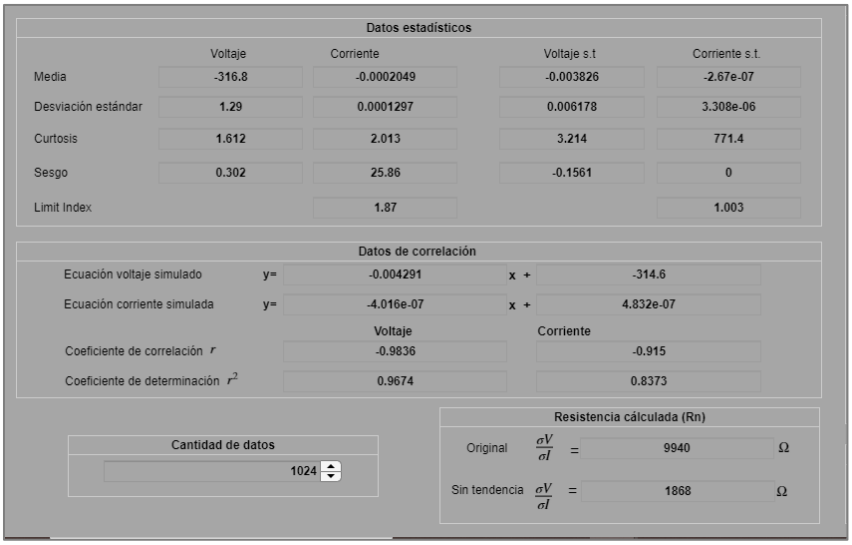


Figura 4.8. Visualización de valores numéricos en interfaz.

Para la presentación de los resultados en la interfaz, se colocan en la Ventana de Diseño objetos *Label* antecediendo al campo numérico, esto permite una mayor facilidad de personalización. La edición de las etiquetas se realiza de forma sencilla desde el Navegador de componentes, el cual permite opciones de edición de texto como seleccionar el tipo de fuente, tamaño el uso de cursivas o negritas, hasta la introducción de caracteres especiales.

Para ajustar las etiquetas de forma precisa es posible utilizar la opción *Position* del Navegador de componentes. En la Figura 4.9 se observan parte de las opciones de edición que ofrece el Navegador de componentes.

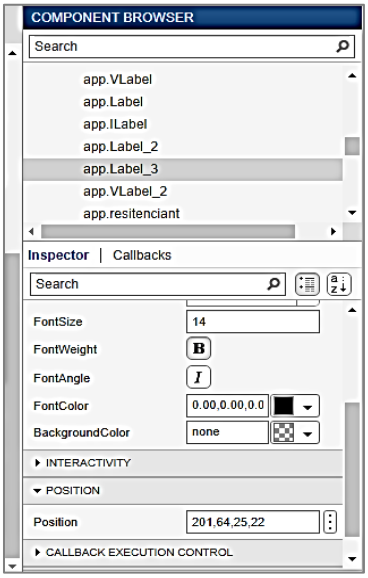


Figura 4.9. Opciones de formato y posición del objeto Label en el Navegador de componentes.

3) Gráficos

Los gráficos que se presentan al usuario son de dos tipos: los primeros contienen la información de corriente y voltaje considerando el método de regresión lineal; mientras que los segundos presentan los datos de voltaje y corriente con el tratamiento de remoción de tendencia.

Para mostrar los gráficos al usuario se hace uso del objeto de *App Designer* denominado *UIAxes***.** Estos objetos presentan la visualización básica del gráfico, sin embargo, más características adicionales puedes ser manipulado mediante código.

La función *plot* de Matlab® permite graficar los datos, no obstante, para ser mostrado es necesaria su vinculación al objeto tipo *UIAxes* en el cual será presentado, de esta forma la estructura de la función es la siguiente:

```
g1=plot(app.Graficov,tiemp1,volt1,'-k');
```

Con la sintaxis propuesta anteriormente se obtienen gráficos de líneas en 2-D útiles para observar el comportamiento del voltaje y corriente con respecto al tiempo, los cuales son observados en la Figura 4.10. Opciones como la activación de una malla dentro del gráfico, tipo de marcadores, colores, etc., pueden ser modificadas mediante las propiedades del objeto *UIAxes* en el Navegador de componentes, como es observable en la Figura 4.11. Sin embargo, en caso de características específicas o de mayor detalle (como la delimitación de valores en los ejes) es recomendable la edición por código.

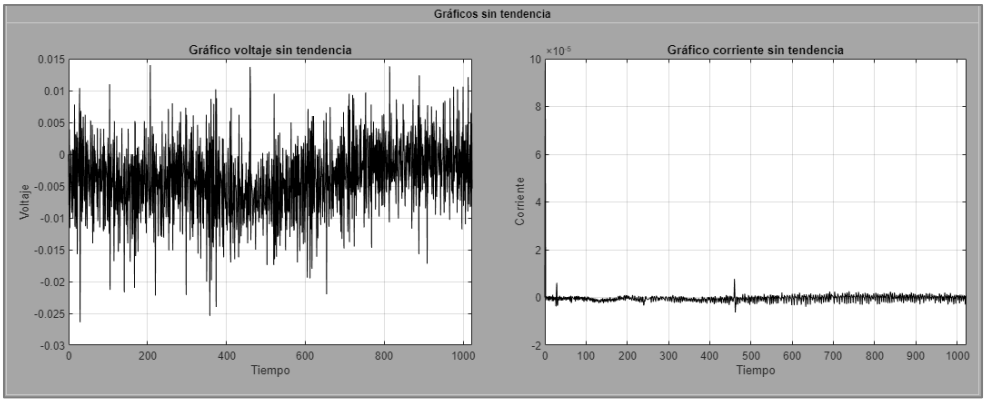


Figura 4.10. Gráficos generados por la interfaz. Izquierda, gráfica de voltaje sin tendencia; Derecha, gráfico de corriente sin tendencia.

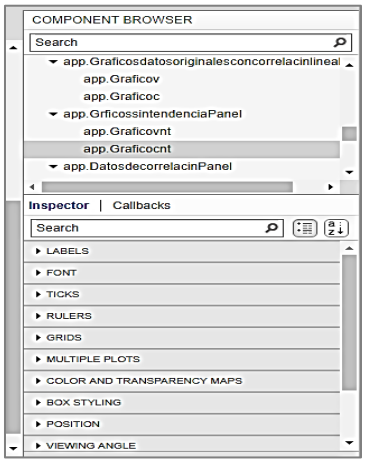


Figura 4.11. Opciones del objeto UIAxes en el Navegador de componentes.

Para evitar conflicto entre los gráficos generados inicialmente en el código y los que fuesen generados después de varias ejecuciones, se optó por utilizar sintaxis adicional a las condiciones predefinidas en el Navegador de componentes. Una de las funciones introducidas es *cla***,** la cual permite de limpiar el espacio del objeto *UIAxes* antes de generar una visualización, el uso de esta función es aún más efectivo combinado con la propiedad *reset* para volver a las condiciones predefinidas.

```
cla(app.Graficov,'reset');
```

La función *hold* al igual que en Matlab®, permite sobreponer dos gráficos en una misma figura. Esta utilidad se hace necesaria en la interfaz para la evaluación del modelo de regresión lineal y su línea de tendencia. Ejemplo de un gráfico de valores con la línea de tendencia es mostrado en la Figura 4.12.

```
hold(app.Graficov);
```

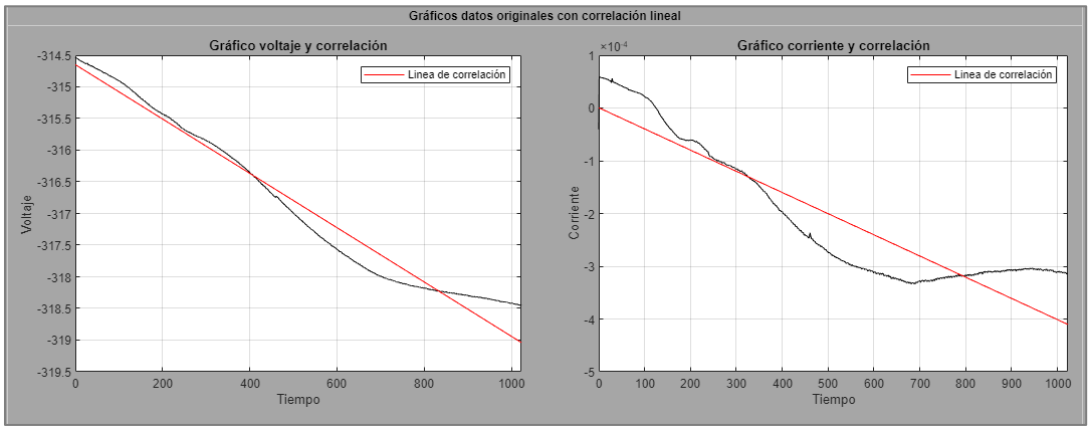


Figura 4.12. Gráficos de voltaje y corriente con línea de tendencia generados por la interfaz.

Adicionalmente se debió recurrir vía código a las funciones *legend* (para el uso separado de las leyendas en los gráficos), *xlim* (delimita el rango de valores del eje x) y *grid* (permite conservar una malla dentro de los gráficos) para conservar un formato de visualización estándar al usuario el cual no debía ser afectado por la acción del botón RESET que será explicado más adelante.

```
% leyenda del grafico
leg=legend(app.Graficoc,g4,'Linea de correlación');
% limites en eje x
xlim(app.Graficoc,[tiemp1(1) tiemp1(ntot)]);
% malla del gráfico
grid(app.Graficoc,"on");
```

Debido a la naturaleza de los datos es necesario colocar la opción de una escala logarítmica en los gráficos. Para activar esta opción se recurre al objeto *Checkbox* situado al pie cada una de las gráficas (mostrado en la Figura 4.13)**,** el cual al ser marcado ofrecerá al usuario la posibilidad de cambiar la escala de los ejes. La implementación dentro del código fue posible al proponer un ciclo *if* relacionado al valor booleano del *Checkbox* el cual al activarse la función *plot* será sustituida por *loglog* para generar el grafico. El resultado de estas operaciones es ejemplificado con los gráficos de la Figura 4.14.

```
if app.Esclog2.Value==0;
cla(app.Graficoc,"reset");
g4=plot(app.Graficoc,tiemp1,corrsimlib,'-r');
else
% ----------Escala logarítmica----------
cla(app.Graficoc,"reset");
g3=loglog(app.Graficoc,tiemp1,corr1,'-k');
end
```

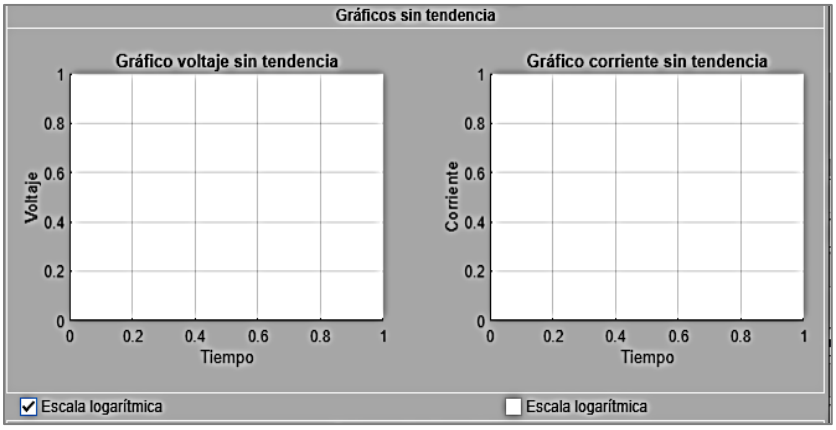


Figura 4.13. CheckBox activado en gráfico de voltaje (izquierda) y desactivado en el gráfico de corriente (derecha).

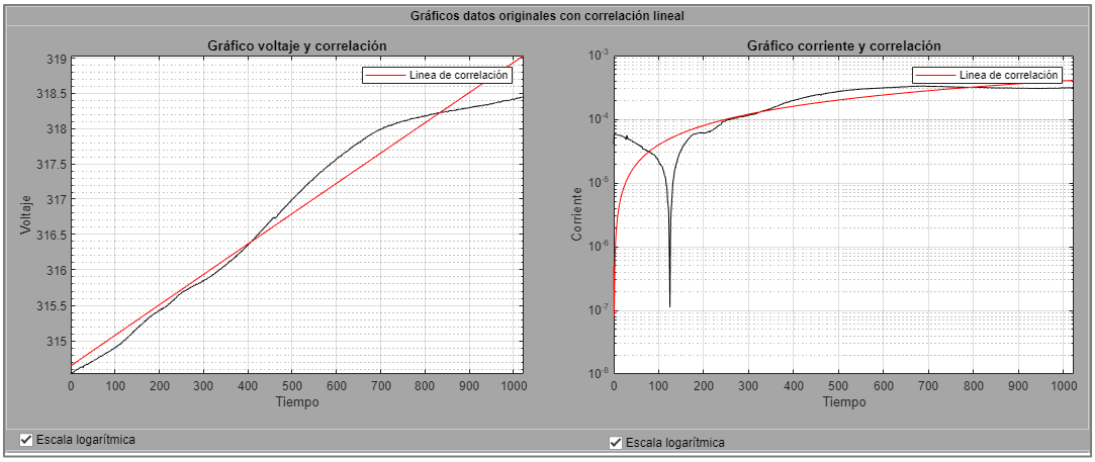


Figura 4.14. Gráficos de voltaje y corriente con escala logarítmica generados por la interfaz.

4.4. Limpieza de resultados

Para conseguir que la interfaz se mantenga en operación, para cargar una nueva base de datos sin forzar su cierre, se implementó una función para limpiar los objetos de la interfaz. De esta forma se hace uso de un botón de acción denominado RESET, el cual al activarse vincula a las funciones asociadas a un *Callback* para la limpieza de variables (o el regreso a un valor default) como de objetos de visualización.

Entre las funciones de depuración se encuentra *cla*, descrita con anterioridad y la cual hace una depuración del objeto *UIAxes* seleccionado.

```
% cla reset para graficos de voltaje y corriente
cla(app.Graficov);
cla(app.Graficoc);
```

En el caso de objetos de campo numérico o de tipo tabla es posible aplicar corchetes vacíos a su propiedad *Data*, para ser devuelto el objeto con valores nulos.

```
% Tabla a valores nulos
app.Tabladat.Data=[];
app.Tablauser.Data=[];
```

Las funciones que devuelven a un valor default al objeto pueden ser observadas en los *Edit File Numeric* y en *Lamp*. En el primer caso al activar la acción del *Callback* los valores regresaran a 0, esto al emplear la propiedad *Value*, por su parte al utilizar la propiedad *Color* del objeto *Lamp* es posible devolverlo al color (la sintaxis debe hacerse escribiendo el nombre de color en ingles entre comillas simples) anterior a la carga de datos en la interfaz, quedando este indicador nuevamente disponible.

```
% Limpieza de valores en Edit File de voltaje
app.mediav.Value = 0;
app.desviacionv.Value =0;
% Led a color inicial
app.lampcar.Color='red'
```

4.5. Guardado de resultados

La interfaz permite guardar los resultados presentados en la tabla de valores, campos de edición numérica y gráficas. Para poder ejecutar la acción de guardado se debe seleccionar en el menú superior la opción “Guardar” y posteriormente elegir entre Guardar Datos numéricos o Guardar Gráficos. En ambos casos el usuario puede optar por dos diferentes tipos de formato de archivo. Para los datos numéricos es posible tener archivos tipo texto (.txt) o de *Microsoft Excel* (.xlsx); por su parte, los gráficos permiten su guardado como

archivos PNG o PDF. En la Figura 4.15 se observan las opciones que ofrece la interfaz para guardar archivos.

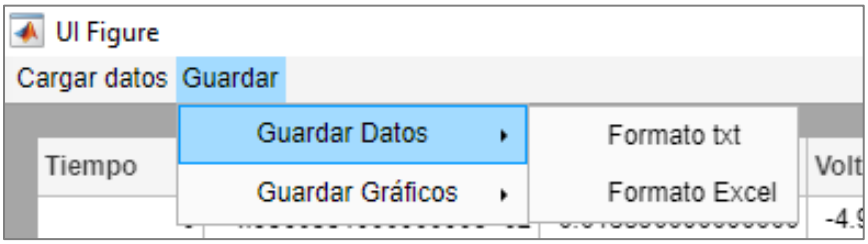


Figura 4.15. Guardado de datos desde el menú superior de la interfaz.

Al activar alguna de las opciones del menú superior para guardar datos o gráficos se activará el *Callback* con la función correspondiente. El activar la opción de guardar hace necesario que algunas de las variables dentro del código sean compartidas entre *Callbacks*. Para conseguirlo en *App Designer*, es necesario que las variables requeridas sean declaradas en la sección *Properties* de la Ventana de Código con el carácter de públicas.

```
%% Se declaran variables para que puedan ser usadas entre Callbacks
properties (Access = public)
graficov
g4
Tab_res
end
%% Se utilizara un objeto app para la variable global
Tab_res=app.Tab_res;
% La variable global se guarda en un el mismo objeto app para poder utilizarse entre otros Callbacks
app.Tab_res=Tab_res;
```

Los archivos generados al elegir la opción Guardar Datos serán dos: el primero, contiene los valores de la tabla de usuario, y, el segundo que almacena los resultados visualizados en los *Edit Field Numeric*. Este último archivo será creado a partir de una variable tipo *table* de Matlab®.

```
Tab_res=table(Datos,Voltaje_Volt,Corriente_Amper,Resistencia_Ohm);
```

La interfaz hace uso de la función *uigetdir*, la cual permite al usuario seleccionar la carpeta destino para el guardado de los archivos. La función al ser activada abre el directorio de carpetas de *Windows* (mostrado en la Figura 4.16), y obtiene al mismo tiempo la ruta de la carpeta, la cual es guardada dentro de una variable.

```
direci=uigetdir;
```

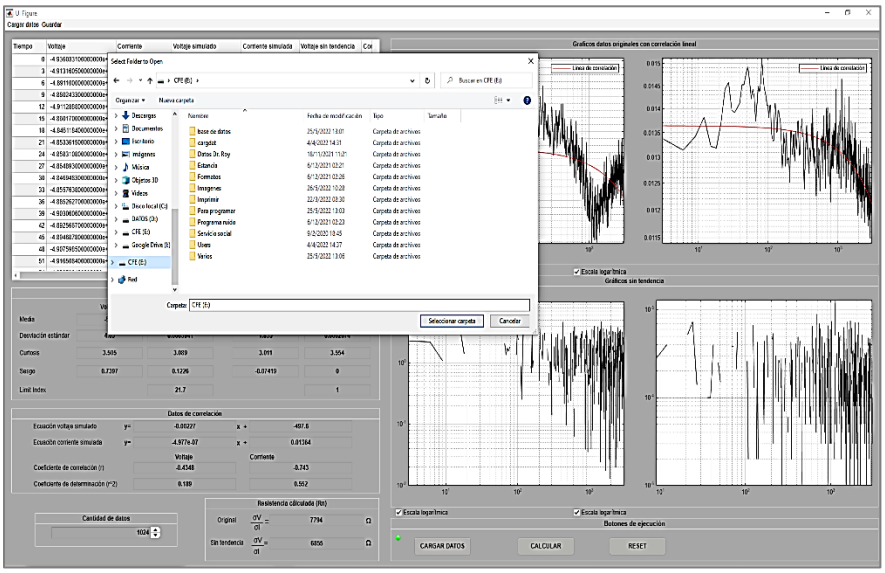


Figura 4.16. Ventana de selección de carpeta para guardar resultados y gráficos de la interfaz.

La variable obtenida por medio de la función *uigetdir* se utiliza para construir la ruta de guardado, que sirve como parte del argumento de entrada para las funciones *writematrix* y *writetable*. Estas funciones permiten guardar los datos de la tabla de los valores de usuario y los de la tabla de resultados estadísticos.

```
writematrix(data, filename);
writetable(vres,filename2);
```

Las funciones *writetable* y *writematrix* requieren como argumentos de entrada a la variable con los datos numéricos, y, al nombre del archivo combinado con la ruta de destino. Por ruta de destino debe entenderse a la dirección *URL* interna de la computadora (obtenida anteriormente con la función *uigetdir*). En el presente trabajo la creación del nombre para los archivos a guardar se estableció en cuatro partes para evitar conflictos de sobreescritura: en primer lugar, la etiqueta o identificador del archivo; en segundo lugar, la fecha de guardado

(obtenida mediante la función de Matlab® *date* convertida a dato tipo carácter); en tercer lugar, el dato del reloj del cpu (conseguida por la función *cputime* de Matlab® y posteriormente convertida a variable tipo carácter); y en último lugar la extensión del archivo (.txt o .xlsx). Se concatenan los elementos con la función *strcat* para obtener el nombre completo del archivo. En la Figura 4.17 se puede observar un ejemplo de archivos guardados por la interfaz.

```
marv=date;
marv2=datestr(marv);
hor=cputime;
hor1=num2str(hor);
% Se presenta el nombre general del documento
marv3='Valores tabla ';
marv4='Resultados ';
% Se coloca la extensión del archivo a guardar
nom1='.txt';
% Los strings se unen para formar el nombre completo del archivo
nomap=strcat(marv3,hor3,nom1);
nomap2=strcat(marv4,hor3,nom1);
filename=strcat(direci,nomap);
filename2=strcat(direci,nomap2);
```

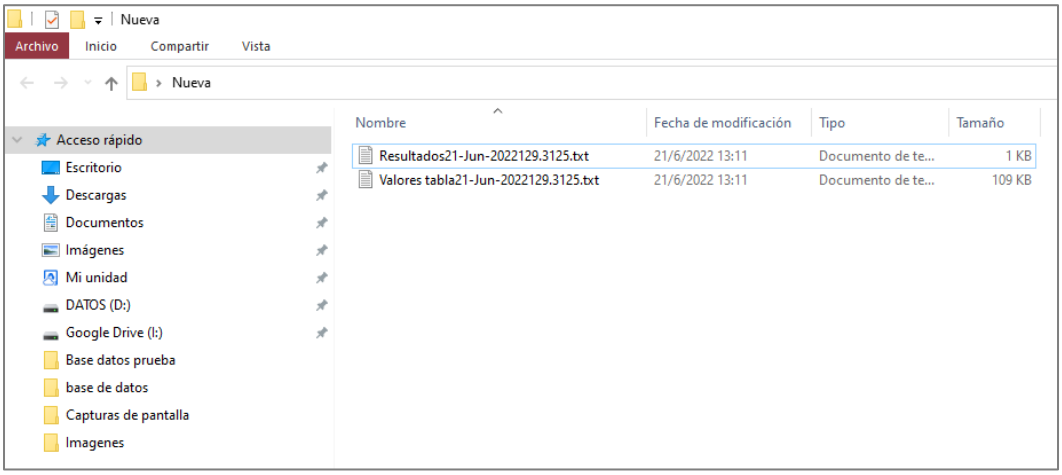


Figura 4.17. Archivos de datos numéricos guardados por la interfaz en formato txt.

Para guardar los gráficos generados por la interfaz se utiliza la función *exportgraphics*. Los argumentos de entrada de la función son una variable contenedora de un elemento gráfico y el nombre del archivo combinado con la ruta de destino. El procedimiento para el guardado es idéntico al utilizado para los datos numéricos (a excepción de las extensiones válidas para los archivos, en este caso .png o pdf.). Ejemplo de gráficos guardados pueden ser observados en la Figura 4.18.

```
GV = app.Graficov;
margv='Gráfico voltaje ';
horg=cputime;
horg2=num2str(horg);
horg3=strcat(margv,horg2);
extg='.png';
direci2=uigetdir;
% Los strings se unen para formar el nombre completo del archivo
nompgv=strcat(direci2,horg3,extg);
%% Se utiliza la funcion exportgraphics para guardar la imagen
exportgraphics(GV,nompgv);
```

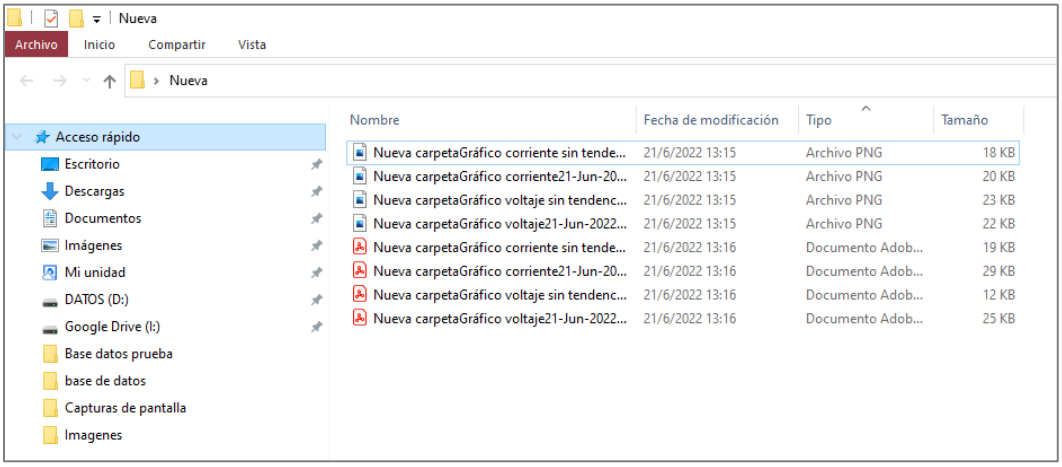


Figura 4.18. Archivos de gráficos guardados por la interfaz en formato png y pdf.

Es importante mencionar que el primer argumento de entrada para las funciones de salvado (referentes a la variable a guardar) no pude ser asociado directamente a un objeto de *App Designer*.

Finalmente, con lo anteriormente descrito fue posible la asociación del algoritmo de programación a los elementos de *App Designer* para la creación de la Interfaz Gráfica de Usuario. Consiguiendo de esta manera presentar al usuario en forma sencilla los resultados del método estadístico para el análisis de Ruido Electroquímico.

5. CONCLUSIONES

El objetivo del presente trabajo fue conseguir la automatización del análisis de Ruido Electroquímico, objetivo que pudo cumplirse a partir de la creación de una Interfaz Gráfica de Usuario.

La aplicación creada permite a los usuarios cargar una base de datos y aplicar un algoritmo para el cálculo de las variables necesarias en el desarrollo del método estadístico.

Los resultados obtenidos por la interfaz pueden ser visualizados por medio de indicadores numéricos y gráficas, los cuales pueden ser guardados por el usuario.

La interfaz presentada consiguió ser un entorno amigable que permite en pocos pasos, la visualización, tratamiento, procesamiento y guardado de datos provenientes de Ruido Electroquímico.

5.1. RECOMENDACIONES

A lo largo de la bibliografía consultada se encontró que existen diferentes posturas entre los autores para la consideración de los parámetros de la corrosión, los cuales en gran medida son observables mediante un tratamiento más avanzado de los datos y su correspondiente representación gráfica.

De esta forma se hace la recomendación de anexar a la interfaz los métodos de Análisis Espectral y por Transformada de Wavelet para la comparativa de resultados y obtener una respuesta más concluyente del fenómeno observado con los datos de Ruido Electroquímico.

6. REFERENCIAS BIBLIOGRAFÍCAS

Albright, R. E. (2003). Roadmapping in the corporation. *Research Technology*, 31-40.

Andrade, M. C., & Feliu, S. (1991). *Corrosion y Proteccion Metalicas Vol. 1*. Madrid: CSIC.

Autolab, M. (28 de enero de 2022). *Metrohm Autolab*. Obtenido de https://www.metrohm-autolab.com/

Bautista, A., & Vergara, A. (1997). Comparación del ruido electroquímico con las técnicas de impedancia y resistencia de polarización en el sistema acero/hormigón. *Revista De Metalurgia*, 113-119.

Botana, J. (2002). *Ruido Electroquímico: Métodos de Análisis.* Cadiz España: Septem Ediciones.

Casquete García , A. (2016). *Técnica de Ruido Electroquímico para el estudio del comportamiento frente a la corrosión de aceros inoxidables (Wrought y Pulvimetalúrgicos).* Valladolid España: Universidad de Valladolid Escuela de Ingenierías Industriales.

Cid Espinosa, G. (2018). *Programación de Interfaz Gráfica en App Designer para el control vectorial de motores de imanes permanentes.* Madrid: Escuela Tecnica Superior de Ingenieros Industriales, Universidad Politecnica de Madrid.

Davis, J. R. (2000). *Corrosion: Understanding the Basics.* Ohio: ASM International.

Dingle, J. (20 de Enero de 2012). *Electrochemical Noise Analysis*. Obtenido de http://www.electrochemicalnoise.com/ecn%20analysis.html

Drake Moyano, J. (2005). Instrumentación electrónica de telecomunicaciones. En J. M. Drake Moyano, *Instrumentación electrónica de telecomunicaciones* (pág. 1). Santander: Universidad de Cantabria.

ECCA. (10 de septiembre de 2022). *Prepaintedmetal The Basics of Corrosion. TechnicalPaper.* Obtenido de http://www.prepaintedmetal.eu/repository/Annina/Basic%20of%20corrosion%2021211.pdf.

Fajardo González, C. M., & Ruballo García, J. L. (2022). *Evalución de la Corrosión Natural y Acelerada en Acero A36 con sistema de Recubrimiento Anticorrosivo Epoxico.* San Salvador: Universidad del Salvador.

Gallego Carrillo, M. (2005). *Interfaces gráficas en Java.* Madrid: Editorial Universitaria Ramón Areces.

Galvele, J. (2011). *Materiales y Materias Primas. Los Materiales y la Humanidad.* Buenos Aires: Ministerio de de Educación - Instituto Nacional de Educación Tecnológica.

Gamry Instruments. (22 de Febrero de 2022). *gamry.com*. Obtenido de https://www.gamry.com/

Goellner, J. (2004). Elektrochemisches Rauschen bei der Korrosion. *Werkstoffe und Korrosion*, 727-734.

Gutierrez, A., Serna, M., Parada, J., & Torres, O. (2014). *Mapa de Ruta para IOT*. Distrito Federal, México: CANIETI.

Ivium Technologies. (27 de febrero de 2022). *Innovative Electrochemical Instrumentation from Ivium Technologies*. Obtenido de https://www.ivium.com/iviumsoft/

Javaherdashti, R. (2008). *Microbiologically Influenced Corrosion - An Engineering Insight.* Londres: Springer.

Luengas, L. A., & Toloza, D. C. (2020). Análisis frecuencial y de la densidad espectral de potencia de la estabilidad de sujetos amputados. *TecnoLógicas*, 1-16.

Malo Tamayo, J., & Uruchurtu Chavarín, J. (2000). *La Tecnica de Ruido Electroquímico para el estudio de la Corrosión.* Cuernavaca, Morelos: Instituto de Investigaciones Eléctricas.

Mariaca, L., & Bautista, A. (1997). Use of electrochemical noise for studying the rate of corrosion of reinforcements embedded in concrete. *Materials and Structures*, 613-617.

Medina, M. (7 de Enero de 2020). *Linkedin*. Obtenido de ¿Cuál es la utilidad de la serie galvánica en la protección catódica? Serie galvánica: clasificación de los metales : https://es.linkedin.com/pulse/cu%C3%A1l-es-la-utilidad-de-serie-galv%C3%A1nica-en-protecci%C3%B3n-cat%C3%B3dica-medina

Moore, H. (2007). *Matlab para ingenieros.* Salt Lake City , Utah: Pearson Prentice Hall.

Obando Ramírez, A. (2013). *Propuesta de procedimientos de las técnicas: Ruido electroquímico, Resistencia a la polarización e impedancia electroquímica usadas en la medición de la corrosión del refuerzo en el concreto reforzado.* Bogotá: Universidad Nacional de Colombia.

Qiao, G., & Ou, J. (2007). Corrosion monitoring of reinforcing steel in cement mortar by EIS and ENA. *Electrochimica Acta*, (págs. 8008-8019).

Revie, W. (2011). *Uhlig's Corrosion Handbook.* Ottawa, Ontario, Canada: John Wiley & Sons, Inc.

Rivera, N. (20 de Junio de 2015). *hipertextual*. Obtenido de https://hipertextual.com/2015/06/internet-of-things

Rosa María González Muradás, P. M. (2014). *Química: Serie Universitaria Patria.* Grupo Editorial Patria, 2014. doi:6074389357, 9786074389357

Rubio Foces, M. (2019). *Técnica del Ruido Electroquímico para el estudio del comportamiento frente a la corrosión en medio biológico de los aceros inoxidables Wrought.* Valladolid España: Universidad de Valldolid Escuela de Ingenierías Insdustriales.

Ruiz Enciso, J., & Rojas Salinas, A. (2007). Sistema que supervisa la corrosión en tiempo real por análisis del ruido electroquímico. *Scientia et Technica*, 937-940.

Salazar Jiménez, J. (2015). Introduction to Corrosion Phenomena: Types, Influencing Factors and Control for Material's Protection. *Revista Tecnología en Marcha*, 127-136.

Santander Morales, C. B. (2008). *Estudio experimental de Corrosión en métales de uso industrial por Desulfovidrio Desulfuricans.* Santiago de Chile: Universidad de Chile Facultad de Ciencias Físicas y Matemáticas.

Sarmiento Klapper, H., & Heyn , A. (2007). Utilización de la técnica de ruido electroquímico para la investigación y monitoreo de la corrosión. *Ingeniería y Desarrollo*, 57-72.

Shi, Y., & Zhang, Z. (2008). Dimensional analysis applied to pitting corrosion measurements. *Electrochimica Acta*, 2688-2698.

Surendra, P. (2005). *Estadística Básica para el manejo de datos experimentales: Aplicación en la Geoquímica (Geoquimetría).* Temixco, Morelos, México: Centro de Investigaciones en Energía, Universidad Nacional Autonoma de México.

Toledano Vivar, L. (2019). *Desarrollo de App en Matlab para la rehabilitación de espasticidad con ayud del robot colaborativo Kuka LBR IIWA.* Leganés: Universidad Carlos III de Madrid.

Uhlig, H., & Revie, W. (2008). *Corrosion and Corrosion Control: An Introduction to Corrosion Science and Engineerin.* New Jersey, Estados Unidos: John Wiley & Sons.

Zhao, B., & Li, J.-H. (2007). Study on the corrosion behavior of reinforcing steel in cement mortar by electrochemical noise measurements. *Electrochimica Acta* (págs. 3976-3984). Fujian: Department of Chemistry, College of Chemistry and Chemical Engineering, Xiamen University.

APENDICE

Se incluye el código que establece el funcionamiento de la interfaz.

```
classdef cargdat < matlab.apps.AppBase


    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                matlab.ui.Figure
        CargardatosMenu         matlab.ui.container.Menu
        CargarArchivoMenu       matlab.ui.container.Menu
        GuardarMenu             matlab.ui.container.Menu
        menuguardar             matlab.ui.container.Menu
        FormatotxtMenu          matlab.ui.container.Menu
        FormatoExcelMenu        matlab.ui.container.Menu
        GuardarGrficosMenu      matlab.ui.container.Menu
        FormatoPNGMenu          matlab.ui.container.Menu
        FormatoPDFMenu          matlab.ui.container.Menu
        Tabladat                matlab.ui.control.Table
        Tablauser               matlab.ui.control.Table
        panelv                  matlab.ui.container.Panel
        mediav                  matlab.ui.control.NumericEditField
        desviacionv             matlab.ui.control.NumericEditField
        curtosisv               matlab.ui.control.NumericEditField
        sesgv                   matlab.ui.control.NumericEditField
        VoltajeLabel            matlab.ui.control.Label
        CorrienteLabel          matlab.ui.control.Label
        MediaLabel              matlab.ui.control.Label
        DesviacinestndarLabel   matlab.ui.control.Label
        CurtosisLabel           matlab.ui.control.Label
        SesgoLabel              matlab.ui.control.Label
        mediac                  matlab.ui.control.NumericEditField
        curtosisc               matlab.ui.control.NumericEditField
        desviacionc             matlab.ui.control.NumericEditField
        sesgc                   matlab.ui.control.NumericEditField
        LimitIndexLabel         matlab.ui.control.Label
        limin                   matlab.ui.control.NumericEditField
        VoltajestLabel          matlab.ui.control.Label
        CorrientestLabel        matlab.ui.control.Label
        mediavnt                matlab.ui.control.NumericEditField
        mediacnt                matlab.ui.control.NumericEditField
```

desvvcnt matlab.ui.control.NumericEditField
desvvnt matlab.ui.control.NumericEditField
curtcnt matlab.ui.control.NumericEditField
curtvnt matlab.ui.control.NumericEditField
segcnt matlab.ui.control.NumericEditField
sesgvnt matlab.ui.control.NumericEditField
limitnt matlab.ui.control.NumericEditField
CantidaddedatosPanel matlab.ui.container.Panel
datpprueb matlab.ui.control.Spinner
ResistenciaclculadaRnPanel matlab.ui.container.Panel
resitencia matlab.ui.control.NumericEditField
Label_2 matlab.ui.control.Label
resitenciant matlab.ui.control.NumericEditField
Label_5 matlab.ui.control.Label
OriginalLabel matlab.ui.control.Label
SintendenciaLabel matlab.ui.control.Label
fracsigmaVsigmaILabel matlab.ui.control.Label
fracsigmaVsigmaILabel_2 matlab.ui.control.Label
OmegaLabel_2 matlab.ui.control.Label
OmegaLabel_3 matlab.ui.control.Label
GrficosdatosoriginalesconcorrelacinlinealPanel matlab.ui.container.Panel
Graficov matlab.ui.control.UIAxes
Graficoc matlab.ui.control.UIAxes
GrficossintendenciaPanel matlab.ui.container.Panel
Graficovnt matlab.ui.control.UIAxes
Graficocnt matlab.ui.control.UIAxes
DatosdecorrelacinPanel matlab.ui.container.Panel
mvs matlab.ui.control.NumericEditField
mcs matlab.ui.control.NumericEditField
bvs matlab.ui.control.NumericEditField
bcs matlab.ui.control.NumericEditField
EcuacinvoltajesimuladoLabel matlab.ui.control.Label
EcuacincorrientesimuladaLabel matlab.ui.control.Label
yLabel matlab.ui.control.Label
yLabel_2 matlab.ui.control.Label
xLabel matlab.ui.control.Label
xLabel_2 matlab.ui.control.Label
CoeficientedecorrelacinLabel matlab.ui.control.Label
CoeficientededeterminacinLabel matlab.ui.control.Label
VoltajeLabel_2 matlab.ui.control.Label
CorrienteLabel_2 matlab.ui.control.Label
rvs matlab.ui.control.NumericEditField
rcs matlab.ui.control.NumericEditField
rrvs matlab.ui.control.NumericEditField
rrcs matlab.ui.control.NumericEditField
r2Label matlab.ui.control.Label
rLabel matlab.ui.control.Label

```
    BotonesdeejecucinPanel          matlab.ui.container.Panel
    Buscaxls                        matlab.ui.control.Button
    CALCULARButton                  matlab.ui.control.Button
    RESETButton                     matlab.ui.control.Button
    lampcar                         matlab.ui.control.Lamp
    Esclog1                         matlab.ui.control.CheckBox
    Esclog2                         matlab.ui.control.CheckBox
    Esclog3                         matlab.ui.control.CheckBox
    Esclog4                         matlab.ui.control.CheckBox
end
properties (Access = public)
    volt1
    corr1
    voltsimlib
    corrsimlib
    desvclib
    graficov
    g4
    Tab_res

end

% Callbacks that handle component events
methods (Access = private)
    % Button pushed function: Buscaxls
    function BuscaxlsButtonPushed(app, event)

     [nombre,ruta]=uigetfile({'*.*'},'Abrir archivo');
        if nombre== 0
            return;
        else
     datos= strcat(ruta,nombre);
     app.Tabladat.Data=readtable(datos);
     ind0=(app.Tabladat.Data(:,1));
     ind1=table2array(ind0);

        if isempty(ind1)
         app.lampcar.Color='red';
        else
         app.lampcar.Color='green';
        end

        end
    end
    % Button pushed function: CALCULARButton
    function CALCULARButtonPushed(app, event)
```

```
Tab_res=app.Tab_res;
ndat=app.datpprueb.Value;
ntot=ndat;
vmat=app.Tabladat.Data;
vmat1=table2array(vmat);
ti=vmat1(1:ntot,1);
vo=vmat1(1:ntot,2);
co=vmat1(1:ntot,3);
[nf,nc]=size(ti);
tiemp1=ti;
volt1=vo;
corr1=co;
medv=mean(volt1);
medc=mean(corr1);
medvlib=sum(volt1)/nf;
medclib=sum(corr1)/nf;
medtlib=sum(tiemp1)/nf;
desvv=std(volt1);
desvc=std(corr1);
desvvlib=sqrt((sum((volt1-medvlib).^2))/nf);
desvclib=sqrt((sum((corr1-medclib).^2))/nf);
curtv=kurtosis(volt1);
curtc=kurtosis(corr1);
curtvlib=(nf*(sum((volt1-medvlib).^4)))/((sum((volt1-medvlib).^2))^2);
curtclib=(nf*(sum((corr1-medclib).^4)))/((sum((corr1-medclib).^2))^2);
sesgov=skewness(volt1);
sesgoc=skewness(corr1);
sesgovlib=(sqrt(nf)*sum((volt1-medvlib).^3))/((sum((volt1-medvlib).^2))^(3/2));
sesgoclib=(sqrt(nf)*sum((corr1-medclib).^3))/((sum((corr1-medclib).^2))^(3/2));
lrmsv=sqrt((sum(volt1.^2))/ntot);
lrmsc=sqrt((sum(corr1.^2))/ntot);
liv=desvv/lrmsv;
lic=desvc/lrmsc;
lrmsvlib=sqrt((sum(volt1.^2))/nf);
lrmsclib=sqrt((sum(corr1.^2))/nf);
livlib=desvvlib/lrmsvlib;
liclib=desvclib/lrmsclib;

resist=desvv/desvc;
resistlib=desvvlib/desvclib;

mvlib=((nf*(sum(tiemp1.*volt1)))-
(sum(tiemp1)*sum(volt1)))/((nf*(sum(tiemp1.^2)))-(sum(tiemp1)^2));
bvlib=((sum(volt1))-(mvlib*(sum(tiemp1))))/nf;
rv=(sum((tiemp1-medtlib).*(volt1-medvlib)))/(sqrt((sum((tiemp1-
medtlib).^2))*(sum((volt1-medvlib).^2))));
r2v=rv^2;
```

```
        mclib=((nf*(sum(tiemp1.*corr1)))-
(sum(tiemp1)*sum(corr1)))/((nf*(sum(tiemp1.^2)))-(sum(tiemp1)^2));
        bclib=((sum(corr1))-(mclib*(sum(tiemp1))))/nf;
        rc=(sum((tiemp1-medtlib).*(corr1-medclib)))/(sqrt((sum((tiemp1-
medtlib).^2))*(sum((corr1-medclib).^2))));
        r2c=rc^2;

        voltsimlib=(tiemp1*mvlib)+bvlib;
        corrsimlib=(tiemp1*mclib)+bclib;

        app.mediav.Value = medvlib;
        app.desviacionv.Value = desvvlib;
        app.curtosisv.Value = curtvlib;
        app.sesgv.Value = sesgovlib;
        app.mediac.Value = medclib;
        app.desviacionc.Value = desvclib;
        app.curtosisc.Value = curtclib;
        app.sesgc.Value = sesgoclib;
        app.limin.Value=liclib;
        app.resitencia.Value=resistlib;
        app.mvs.Value=mvlib;
        app.bvs.Value=bvlib,
        app.rvs.Value=rv;
        app.rrvs.Value=r2v;
        app.mcs.Value=mclib;
        app.bcs.Value=bclib;
        app.rcs.Value=rc;
        app.rrcs.Value=r2c;
       app.Tablauser.ColumnFormat=({'long','long','long','long','long','long','long'});
        app.Tablauser.Data(:,1)=(tiemp1);
        app.Tablauser.Data(:,2)=(volt1);
        app.Tablauser.Data(:,3)=(corr1);
        app.Tablauser.Data(:,4)=(voltsimlib);
        app.Tablauser.Data(:,5)=(corrsimlib);

        if app.Esclog1.Value==0;
        cla(app.Graficov,'reset');
        g1=plot(app.Graficov,tiemp1,volt1,'-k');
        hold(app.Graficov);
        g2=plot(app.Graficov,tiemp1,voltsimlib,'-r');
        leg=legend(app.Graficov,g2,'Linea de correlación');
        xlim(app.Graficov,[tiemp1(1) tiemp1(ntot)]);
        grid(app.Graficov,"on");
        xlabel(app.Graficov, 'Tiempo');
        ylabel(app.Graficov, 'Voltaje');
        title(app.Graficov, 'Gráfico voltaje y correlación');
        else
```

```
cla(app.Graficov,"reset");
volt1log=abs(volt1);
voltsimliblog=abs(voltsimlib);
g1=semilogy(app.Graficov,tiemp1,volt1log,'-k');
hold(app.Graficov);
g2=plot(app.Graficov,tiemp1,voltsimliblog,'-r');
leg=legend(app.Graficov,g2,'Linea de correlación');
xlim(app.Graficov,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficov,"on");
xlabel(app.Graficov, 'Tiempo');
ylabel(app.Graficov, 'Voltaje');
title(app.Graficov, 'Gráfico voltaje y correlación');
end
if app.Esclog2.Value==0;
cla(app.Graficoc,"reset");
g3=plot(app.Graficoc,tiemp1,corr1,'-k');
hold(app.Graficoc);
g4=plot(app.Graficoc,tiemp1,corrsimlib,'-r');
leg=legend(app.Graficoc,g4,'Linea de correlación');
xlim(app.Graficoc,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficoc,"on");
xlabel(app.Graficoc, 'Tiempo');
ylabel(app.Graficoc, 'Corriente');
title(app.Graficoc, 'Gráfico corriente y correlación');
else
cla(app.Graficoc,"reset");
corr1log=abs(corr1);
corrsimliblog=abs(corrsimlib);
g3=semilogy(app.Graficoc,tiemp1,corr1log,'-k');
hold(app.Graficoc);
g4=plot(app.Graficoc,tiemp1,corrsimliblog,'-r');
leg=legend(app.Graficoc,g4,'Linea de correlación');
xlim(app.Graficoc,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficoc,"on");
xlabel(app.Graficoc, 'Tiempo');
ylabel(app.Graficoc, 'Corriente');
title(app.Graficoc, 'Gráfico corriente y correlación');
end

for ii=1:1:(ntot-1)
ntv(ii+1)=volt1(ii+1)-volt1(ii);
ntc(ii+1)=corr1(ii+1)-corr1(ii);
end

medvnt=sum(ntv)/nf;
medcnt=sum(ntc)/nf;
desvnt=sqrt((sum((ntv-medvnt).^2))/nf);
```

```
descnt=sqrt((sum((ntc-medcnt).^2))/nf);
curvnt=(nf*(sum((ntv-medvnt).^4)))/((sum((ntv-medvnt).^2))^2);
curcnt=(nf*(sum((ntc-medcnt).^4)))/((sum((ntc-medcnt).^2))^2);
sesgovnt=(sqrt(nf)*sum((ntv-medvnt).^3))/((sum((ntv-medvnt).^2))^(3/2));
sesgocnt=(sqrt(nf)*sum((ntc-medcnt).^3))/((sum((ntc-medcnt).^2))^(3/2));
lrmsvnt=sqrt((sum(ntv.^2))/nf);
lrmscnt=sqrt((sum(ntc.^2))/nf);
livnt=desvnt/lrmsvnt;
licnt=descnt/lrmscnt;
resistnt=desvnt/descnt;

app.mediavnt.Value = medvnt;
app.desvvnt.Value = desvnt;
app.curtvnt.Value = curvnt;
app.sesgvnt.Value = sesgovnt;
app.mediacnt.Value = medcnt;
app.desvvcnt.Value = descnt;
app.curtcnt.Value = curcnt;
app.segcnt.Value = sesgocnt;
app.limitnt.Value=licnt;
app.resitenciant.Value=resistnt;
app.Tablauser.Data(:,6)=(ntv);
app.Tablauser.Data(:,7)=(ntc);

if app.Esclog3.Value==0;
cla(app.Graficovnt,"reset");
g5=plot(app.Graficovnt,tiemp1,ntv,'-k');
xlim(app.Graficovnt,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficovnt,"on");
xlabel(app.Graficovnt, 'Tiempo');
ylabel(app.Graficovnt, 'Voltaje');
title(app.Graficovnt, 'Gráfico voltaje sin tendencia');
else
cla(app.Graficovnt,"reset");
ntvlog=abs(ntv);
g5=semilogy(app.Graficovnt,tiemp1,ntvlog,'-k');
xlim(app.Graficovnt,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficovnt,"on");
xlabel(app.Graficovnt, 'Tiempo');
ylabel(app.Graficovnt, 'Voltaje');
title(app.Graficovnt, 'Gráfico voltaje sin tendencia');
end
if app.Esclog4.Value==0;
cla(app.Graficocnt,"reset");
g6=plot(app.Graficocnt,tiemp1,ntc,'-k');
xlim(app.Graficocnt,[tiemp1(1) tiemp1(ntot)]);
grid(app.Graficocnt,"on");
```

```
        xlabel(app.Graficocnt, 'Tiempo');
        ylabel(app.Graficocnt, 'Corriente');
        title(app.Graficocnt, 'Gráfico corriente sin tendencia');
        else
        cla(app.Graficocnt,"reset");
        ntclog=abs(ntc);
        g6=semilogy(app.Graficocnt,tiemp1,ntclog,'-k');
        xlim(app.Graficocnt,[tiemp1(1) tiemp1(ntot)]);
        grid(app.Graficocnt,"on");
        xlabel(app.Graficocnt, 'Tiempo');
        ylabel(app.Graficocnt, 'Corriente');
        title(app.Graficocnt, 'Gráfico corriente sin tendencia');
        end

        Datos=["Media","Media sin tendencia", "Desviación", "Desviación sin tendencia"...
        "Curtosis","Curtosis sin tendencia","Sesgo", "Sesgo sin tendencia"...
        "Índice de localización","Índice de localización sin tendencia", "Coeficiente de
Correlación","Coeficiente de Determinación"...
        "Resistencia calculada","Resistencia sin tendencia"]';
        Voltaje_Volt=[medvnt,medvlib,desvnt,desvvlib,curvnt,curtvlib,sesgovnt,sesgovlib...
        "NA","NA",rv,r2v,"NA","NA"]';

Corriente_Amper=[medcnt,medclib,descnt,desvclib,curcnt,curtclib,sesgocnt,sesgoclib...
        licnt,liclib,rc,r2c,"NA","NA"]';

Resistencia_Ohm=["NA","NA","NA","NA","NA","NA","NA","NA","NA","NA","NA","N
A",resistnt,resistlib]';
        Tab_res=table(Datos,Voltaje_Volt,Corriente_Amper,Resistencia_Ohm);
        app.Tab_res=Tab_res;

        end
        % Cell edit callback: Tabladat
        function TabladatCellEdit(app, event)
        indices = event.Indices;
        newData = event.NewData;
        app.Tabladat.ColumnFormat={'numeric','numeric'};
        end
        % Menu selected function: menuguardar
        function menuguardarSelected(app, event)
        end
        % Button pushed function: RESETButton
        function RESETButtonPushed(app, event)
        app.lampcar.Color='red'
        app.Tabladat.Data=[];
        app.Tablauser.Data=[];
        app.mediav.Value = 0;
        app.desviacionv.Value =0;
```

```
app.curtosisv.Value =0;
app.sesgv.Value = 0;
app.mediac.Value = 0;
app.desviacionc.Value= 0;
app.curtosisc.Value = 0;
app.sesgc.Value = 0;
app.limin.Value=0;
app.resitencia.Value=0;
app.mvs.Value=0;
app.bvs.Value=0,
app.rvs.Value=0;
app.rrvs.Value=0;
app.mcs.Value=0;
app.bcs.Value=0;
app.rcs.Value=0;
app.rrcs.Value=0;
app.mediavnt.Value = 0;
app.desvvnt.Value = 0;
app.curtvnt.Value = 0;
app.sesgvnt.Value = 0;
app.mediacnt.Value = 0;
app.desvvcnt.Value = 0;
app.curtcnt.Value = 0;
app.sesgc.Value = 0;
app.limitnt.Value=0;
app.resitenciant.Value=0;
cla(app.Graficov);
cla(app.Graficoc);
cla(app.Graficovnt);
cla(app.Graficocnt);
end
% Menu selected function: GuardarGrficosMenu
function GuardarGrficosMenuSelected(app, event)
end
% Menu selected function: FormatotxtMenu
function FormatotxtMenuSelected(app, event)
marv=date;
marv2=datestr(marv);
hor=cputime;
hor1=num2str(hor);
hor3=strcat(marv2,hor1);
marv3='\Valores tabla ';
marv4='\Resultados ';
nom1='.txt';
nomap=strcat(marv3,hor3,nom1);
nomap2=strcat(marv4,hor3,nom1);
direci=uigetdir;
```

```
data=app.Tablauser.Data;
vres=app.Tab_res;
filename=strcat(direci,nomap);
filename2=strcat(direci,nomap2);
writematrix(data, filename);
writetable(vres,filename2);
end
% Menu selected function: FormatoExcelMenu
function FormatoExcelMenuSelected(app, event)

marv=date;
marv2=datestr(marv);
hor=cputime;
hor1=num2str(hor);
hor3=strcat(marv2,hor1);
marv3='\Valores tabla ';
marv4='\Resultados ';
nom1='.xlsx';
nomap=strcat(marv3,hor3,nom1);
nomap2=strcat(marv4,hor3,nom1);
direci=uigetdir;
data=app.Tablauser.Data;
vres=app.Tab_res;
filename=strcat(direci,nomap);
filename2=strcat(direci,nomap2);
writematrix(data, filename);
writetable(vres,filename2);

end
% Menu selected function: FormatoPNGMenu
function FormatoPNGMenuSelected(app, event)
GV = app.Graficov;
margv='\Gráfico voltaje ';
marv=date;
marv2=datestr(marv);
horg=cputime;
horg2=num2str(horg);
horg3=strcat(margv,marv2,horg2);
extg='.png';
direci2=uigetdir;
nompgv=strcat(direci2,horg3,extg);
exportgraphics(GV,nompgv);
GC = app.Graficoc;
margc='\Gráfico corriente ';
horg4=strcat(margc,marv2,horg2);
nompgc=strcat(direci2,horg4,extg);
exportgraphics(GC,nompgc);
```

```
GVNT = app.Graficovnt;
margvnt='\Gráfico voltaje sin tendencia ';
horg5=strcat(margvnt,marv2,horg2);
nompgvnt=strcat(direci2,horg5,extg);
exportgraphics(GVNT,nompgvnt);
GCNT = app.Graficocnt;
margcnt='\Gráfico corriente sin tendencia ';
horg6=strcat(margcnt,marv2,horg2);
nompgcnt=strcat(direci2,horg6,extg);
exportgraphics(GCNT,nompgcnt);
end
% Menu selected function: FormatoPDFMenu
function FormatoPDFMenuSelected(app, event)
GV = app.Graficov;
margv='\Gráfico voltaje ';
marv=date;
marv2=datestr(marv);
horg=cputime;
horg2=num2str(horg);
horg3=strcat(margv,marv2,horg2);
extg2='.pdf';
direci2=uigetdir;
nompgv=strcat(direci2,horg3,extg2);
exportgraphics(GV,nompgv);
GC = app.Graficoc;
margc='\Gráfico corriente ';
horg4=strcat(margc,marv2,horg2);
nompgc=strcat(direci2,horg4,extg2);
exportgraphics(GC,nompgc);
GVNT = app.Graficovnt;
margvnt='\Gráfico voltaje sin tendencia ';
horg5=strcat(margvnt,marv2,horg2);
nompgvnt=strcat(direci2,horg5,extg2);
exportgraphics(GVNT,nompgvnt);
GCNT = app.Graficocnt;
margcnt='\Gráfico corriente sin tendencia ';
horg6=strcat(margcnt,marv2,horg2);
nompgcnt=strcat(direci2,horg6,extg2);
exportgraphics(GCNT,nompgcnt);
end

% Menu selected function: CargarArchivoMenu
function CargarArchivoMenuSelected(app, event)

[nombre,ruta]=uigetfile({'*.*'},'Abrir archivo');
if nombre== 0
return;
```

```
        else
        datos= strcat(ruta,nombre);
        app.Tabladat.Data=readtable(datos);
        ind0=(app.Tabladat.Data(:,1));
        ind1=table2array(ind0);
        if isempty(ind1)
        app.lampcar.Color='red';
        else
        app.lampcar.Color='green';
        end
        end
        end
    end
    % Component initialization
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure and hide until all components are created
            app.UIFigure = uifigure('Visible', 'off');
            app.UIFigure.Color = [0.651 0.651 0.651];
            app.UIFigure.Position = [100 100 1173 737];
            app.UIFigure.Name = 'UI Figure';
            % Create CargardatosMenu
            app.CargardatosMenu = uimenu(app.UIFigure);
            app.CargardatosMenu.Text = 'Cargar datos';
            % Create CargarArchivoMenu
            app.CargarArchivoMenu = uimenu(app.CargardatosMenu);
            app.CargarArchivoMenu.MenuSelectedFcn = createCallbackFcn(app,
@CargarArchivoMenuSelected, true);
            app.CargarArchivoMenu.Text = 'Cargar Archivo';
            % Create GuardarMenu
            app.GuardarMenu = uimenu(app.UIFigure);
            app.GuardarMenu.Text = 'Guardar';
            % Create menuguardar
            app.menuguardar = uimenu(app.GuardarMenu);
            app.menuguardar.MenuSelectedFcn = createCallbackFcn(app,
@menuguardarSelected, true);
            app.menuguardar.Text = 'Guardar Datos';
            % Create FormatotxtMenu
            app.FormatotxtMenu = uimenu(app.menuguardar);
            app.FormatotxtMenu.MenuSelectedFcn = createCallbackFcn(app,
@FormatotxtMenuSelected, true);
            app.FormatotxtMenu.Text = 'Formato txt';
            % Create FormatoExcelMenu
            app.FormatoExcelMenu = uimenu(app.menuguardar);
            app.FormatoExcelMenu.MenuSelectedFcn = createCallbackFcn(app,
@FormatoExcelMenuSelected, true);
```

```
            app.FormatoExcelMenu.Text = 'Formato Excel';
            % Create GuardarGrficosMenu
            app.GuardarGrficosMenu = uimenu(app.GuardarMenu);
            app.GuardarGrficosMenu.MenuSelectedFcn = createCallbackFcn(app,
@GuardarGrficosMenuSelected, true);
            app.GuardarGrficosMenu.Text = 'Guardar Gráficos';
            % Create FormatoPNGMenu
            app.FormatoPNGMenu = uimenu(app.GuardarGrficosMenu);
            app.FormatoPNGMenu.MenuSelectedFcn = createCallbackFcn(app,
@FormatoPNGMenuSelected, true);
            app.FormatoPNGMenu.Text = 'Formato PNG';
            % Create FormatoPDFMenu
            app.FormatoPDFMenu = uimenu(app.GuardarGrficosMenu);
            app.FormatoPDFMenu.MenuSelectedFcn = createCallbackFcn(app,
@FormatoPDFMenuSelected, true);
            app.FormatoPDFMenu.Text = 'Formato PDF';
            % Create Tabladat
            app.Tabladat = uitable(app.UIFigure);
            app.Tabladat.ColumnName = {'Column 1'; 'Column 2'; 'Column 3'; 'Column 4'};
            app.Tabladat.RowName = {};
            app.Tabladat.CellEditCallback = createCallbackFcn(app, @TabladatCellEdit, true);
            app.Tabladat.Visible = 'off';
            app.Tabladat.Position = [497 495 10 43];
            % Create Tablauser
            app.Tablauser = uitable(app.UIFigure);
            app.Tablauser.ColumnName = {'Tiempo'; 'Voltaje'; 'Corriente'; 'Voltaje simulado';
'Corriente simulada'; 'Voltaje sin tendencia'; 'Corriente sin tendencia'};
            app.Tablauser.RowName = {};
            app.Tablauser.Position = [15 534 480 190];
            % Create panelv
            app.panelv = uipanel(app.UIFigure);
            app.panelv.TitlePosition = 'centertop';
            app.panelv.Title = 'Datos estadísticos';
            app.panelv.BackgroundColor = [0.651 0.651 0.651];
            app.panelv.FontName = 'Arial';
            app.panelv.FontWeight = 'bold';
            app.panelv.Position = [15 312 480 209];
            % Create mediav
            app.mediav = uieditfield(app.panelv, 'numeric');
            app.mediav.Editable = 'off';
            app.mediav.HorizontalAlignment = 'center';
            app.mediav.FontWeight = 'bold';
            app.mediav.BackgroundColor = [0.651 0.651 0.651];
            app.mediav.Position = [321 139 63 22];
            % Create desviacionv
            app.desviacionv = uieditfield(app.panelv, 'numeric');
            app.desviacionv.Editable = 'off';
```

```
app.desviacionv.HorizontalAlignment = 'center';
app.desviacionv.FontWeight = 'bold';
app.desviacionv.BackgroundColor = [0.651 0.651 0.651];
app.desviacionv.Position = [322 109 63 22];
% Create curtosisv
app.curtosisv = uieditfield(app.panelv, 'numeric');
app.curtosisv.Editable = 'off';
app.curtosisv.HorizontalAlignment = 'center';
app.curtosisv.FontWeight = 'bold';
app.curtosisv.BackgroundColor = [0.651 0.651 0.651];
app.curtosisv.Position = [322 77 63 22];
% Create sesgv
app.sesgv = uieditfield(app.panelv, 'numeric');
app.sesgv.Editable = 'off';
app.sesgv.HorizontalAlignment = 'center';
app.sesgv.FontWeight = 'bold';
app.sesgv.BackgroundColor = [0.651 0.651 0.651];
app.sesgv.Position = [322 43 63 22];
% Create VoltajeLabel
app.VoltajeLabel = uilabel(app.panelv);
app.VoltajeLabel.Position = [154 162 42 22];
app.VoltajeLabel.Text = 'Voltaje';
% Create CorrienteLabel
app.CorrienteLabel = uilabel(app.panelv);
app.CorrienteLabel.Position = [223 162 55 22];
app.CorrienteLabel.Text = 'Corriente';
% Create MediaLabel
app.MediaLabel = uilabel(app.panelv);
app.MediaLabel.Position = [18 139 119 22];
app.MediaLabel.Text = 'Media';
% Create DesviacinestndarLabel
app.DesviacinestndarLabel = uilabel(app.panelv);
app.DesviacinestndarLabel.Position = [18 109 119 22];
app.DesviacinestndarLabel.Text = 'Desviación estándar';
% Create CurtosisLabel
app.CurtosisLabel = uilabel(app.panelv);
app.CurtosisLabel.Position = [18 77 63 22];
app.CurtosisLabel.Text = 'Curtosis';
% Create SesgoLabel
app.SesgoLabel = uilabel(app.panelv);
app.SesgoLabel.Position = [18 43 63 22];
app.SesgoLabel.Text = 'Sesgo';
% Create mediac
app.mediac = uieditfield(app.panelv, 'numeric');
app.mediac.Editable = 'off';
app.mediac.HorizontalAlignment = 'center';
app.mediac.FontWeight = 'bold';
```

```
app.mediac.BackgroundColor = [0.651 0.651 0.651];
app.mediac.Position = [396 139 63 22];
% Create curtosisc
app.curtosisc = uieditfield(app.panelv, 'numeric');
app.curtosisc.Editable = 'off';
app.curtosisc.HorizontalAlignment = 'center';
app.curtosisc.FontWeight = 'bold';
app.curtosisc.BackgroundColor = [0.651 0.651 0.651];
app.curtosisc.Position = [396 77 63 22];
% Create desviacionc
app.desviacionc = uieditfield(app.panelv, 'numeric');
app.desviacionc.Editable = 'off';
app.desviacionc.HorizontalAlignment = 'center';
app.desviacionc.FontWeight = 'bold';
app.desviacionc.BackgroundColor = [0.651 0.651 0.651];
app.desviacionc.Position = [396 109 63 22];
% Create sesgc
app.sesgc = uieditfield(app.panelv, 'numeric');
app.sesgc.Editable = 'off';
app.sesgc.HorizontalAlignment = 'center';
app.sesgc.FontWeight = 'bold';
app.sesgc.BackgroundColor = [0.651 0.651 0.651];
app.sesgc.Position = [396 43 63 22];
% Create LimitIndexLabel
app.LimitIndexLabel = uilabel(app.panelv);
app.LimitIndexLabel.Position = [18 8 64 22];
app.LimitIndexLabel.Text = 'Índice de localización';
% Create limin
app.limin = uieditfield(app.panelv, 'numeric');
app.limin.Editable = 'off';
app.limin.HorizontalAlignment = 'center';
app.limin.FontWeight = 'bold';
app.limin.BackgroundColor = [0.651 0.651 0.651];
app.limin.Position = [396 8 63 22];
% Create VoltajestLabel
app.VoltajestLabel = uilabel(app.panelv);
app.VoltajestLabel.Position = [324 162 58 22];
app.VoltajestLabel.Text = 'Voltaje s.t';
% Create CorrientestLabel
app.CorrientestLabel = uilabel(app.panelv);
app.CorrientestLabel.Position = [391 162 74 22];
app.CorrientestLabel.Text = 'Corriente s.t.';
% Create mediavnt
app.mediavnt = uieditfield(app.panelv, 'numeric');
app.mediavnt.Editable = 'off';
app.mediavnt.HorizontalAlignment = 'center';
app.mediavnt.FontWeight = 'bold';
```

```
app.mediavnt.BackgroundColor = [0.651 0.651 0.651];
app.mediavnt.Position = [143 139 63 22];
% Create mediacnt
app.mediacnt = uieditfield(app.panelv, 'numeric');
app.mediacnt.Editable = 'off';
app.mediacnt.HorizontalAlignment = 'center';
app.mediacnt.FontWeight = 'bold';
app.mediacnt.BackgroundColor = [0.651 0.651 0.651];
app.mediacnt.Position = [217 139 63 22];
% Create desvvcnt
app.desvvcnt = uieditfield(app.panelv, 'numeric');
app.desvvcnt.Editable = 'off';
app.desvvcnt.HorizontalAlignment = 'center';
app.desvvcnt.FontWeight = 'bold';
app.desvvcnt.BackgroundColor = [0.651 0.651 0.651];
app.desvvcnt.Position = [217 109 63 22];
% Create desvvnt
app.desvvnt = uieditfield(app.panelv, 'numeric');
app.desvvnt.Editable = 'off';
app.desvvnt.HorizontalAlignment = 'center';
app.desvvnt.FontWeight = 'bold';
app.desvvnt.BackgroundColor = [0.651 0.651 0.651];
app.desvvnt.Position = [143 109 63 22];
% Create curtcnt
app.curtcnt = uieditfield(app.panelv, 'numeric');
app.curtcnt.Editable = 'off';
app.curtcnt.HorizontalAlignment = 'center';
app.curtcnt.FontWeight = 'bold';
app.curtcnt.BackgroundColor = [0.651 0.651 0.651];
app.curtcnt.Position = [217 77 63 22];
% Create curtvnt
app.curtvnt = uieditfield(app.panelv, 'numeric');
app.curtvnt.Editable = 'off';
app.curtvnt.HorizontalAlignment = 'center';
app.curtvnt.FontWeight = 'bold';
app.curtvnt.BackgroundColor = [0.651 0.651 0.651];
app.curtvnt.Position = [143 77 63 22];
% Create segcnt
app.segcnt = uieditfield(app.panelv, 'numeric');
app.segcnt.Editable = 'off';
app.segcnt.HorizontalAlignment = 'center';
app.segcnt.FontWeight = 'bold';
app.segcnt.BackgroundColor = [0.651 0.651 0.651];
app.segcnt.Position = [217 43 63 22];
% Create sesgvnt
app.sesgvnt = uieditfield(app.panelv, 'numeric');
app.sesgvnt.Editable = 'off';
```

```
app.sesgvnt.HorizontalAlignment = 'center';
app.sesgvnt.FontWeight = 'bold';
app.sesgvnt.BackgroundColor = [0.651 0.651 0.651];
app.sesgvnt.Position = [143 43 63 22];
% Create limitnt
app.limitnt = uieditfield(app.panelv, 'numeric');
app.limitnt.Editable = 'off';
app.limitnt.HorizontalAlignment = 'center';
app.limitnt.FontWeight = 'bold';
app.limitnt.BackgroundColor = [0.651 0.651 0.651];
app.limitnt.Position = [217 8 63 22];
% Create CantidaddedatosPanel
app.CantidaddedatosPanel = uipanel(app.UIFigure);
app.CantidaddedatosPanel.TitlePosition = 'centertop';
app.CantidaddedatosPanel.Title = 'Cantidad de datos';
app.CantidaddedatosPanel.BackgroundColor = [0.651 0.651 0.651];
app.CantidaddedatosPanel.FontWeight = 'bold';
app.CantidaddedatosPanel.Position = [67 41 135 62];
% Create datpprueb
app.datpprueb = uispinner(app.CantidaddedatosPanel);
app.datpprueb.Step = 1024;
app.datpprueb.FontWeight = 'bold';
app.datpprueb.BackgroundColor = [0.651 0.651 0.651];
app.datpprueb.Position = [36 15 64 22];
app.datpprueb.Value = 1024;
% Create ResistenciaclculadaRnPanel
app.ResistenciaclculadaRnPanel = uipanel(app.UIFigure);
app.ResistenciaclculadaRnPanel.TitlePosition = 'centertop';
app.ResistenciaclculadaRnPanel.Title = 'Resistencia cálculada (Rn)';
app.ResistenciaclculadaRnPanel.BackgroundColor = [0.651 0.651 0.651];
app.ResistenciaclculadaRnPanel.FontWeight = 'bold';
app.ResistenciaclculadaRnPanel.Position = [263 14 230 117];
% Create resitencia
app.resitencia = uieditfield(app.ResistenciaclculadaRnPanel, 'numeric');
app.resitencia.Editable = 'off';
app.resitencia.HorizontalAlignment = 'center';
app.resitencia.FontWeight = 'bold';
app.resitencia.BackgroundColor = [0.651 0.651 0.651];
app.resitencia.Position = [143 21 55 22];
% Create Label_2
app.Label_2 = uilabel(app.ResistenciaclculadaRnPanel);
app.Label_2.FontSize = 16;
app.Label_2.Position = [131 62 25 22];
app.Label_2.Text = '=';
% Create resitenciant
app.resitenciant = uieditfield(app.ResistenciaclculadaRnPanel, 'numeric');
app.resitenciant.Editable = 'off';
```

```
            app.resitenciant.HorizontalAlignment = 'center';
            app.resitenciant.FontWeight = 'bold';
            app.resitenciant.BackgroundColor = [0.651 0.651 0.651];
            app.resitenciant.Position = [145 67 55 22];
            % Create Label_5
            app.Label_5 = uilabel(app.ResistenciaclculadaRnPanel);
            app.Label_5.FontSize = 16;
            app.Label_5.Position = [129 22 25 22];
            app.Label_5.Text = '=';
            % Create OriginalLabel
            app.OriginalLabel = uilabel(app.ResistenciaclculadaRnPanel);
            app.OriginalLabel.Position = [27 65 47 22];
            app.OriginalLabel.Text = 'Original';
            % Create SintendenciaLabel
            app.SintendenciaLabel = uilabel(app.ResistenciaclculadaRnPanel);
            app.SintendenciaLabel.Position = [12 23 78 22];
            app.SintendenciaLabel.Text = 'Sin tendencia';
            % Create fracsigmaVsigmaILabel
            app.fracsigmaVsigmaILabel = uilabel(app.ResistenciaclculadaRnPanel);
            app.fracsigmaVsigmaILabel.FontSize = 16;
            app.fracsigmaVsigmaILabel.Position = [96 55 30 42];
            app.fracsigmaVsigmaILabel.Text = '$\frac{\sigma V}{\sigma I}$';
            % Create fracsigmaVsigmaILabel_2
            app.fracsigmaVsigmaILabel_2 = uilabel(app.ResistenciaclculadaRnPanel);
            app.fracsigmaVsigmaILabel_2.FontSize = 16;
            app.fracsigmaVsigmaILabel_2.Position = [95 4 42 47];
            app.fracsigmaVsigmaILabel_2.Text = '$\frac{\sigma V}{\sigma I}$';
            % Create OmegaLabel_2
            app.OmegaLabel_2 = uilabel(app.ResistenciaclculadaRnPanel);
            app.OmegaLabel_2.FontSize = 18;
            app.OmegaLabel_2.Position = [197 20 25 24];
            app.OmegaLabel_2.Text = '$\Omega$';
            % Create OmegaLabel_3
            app.OmegaLabel_3 = uilabel(app.ResistenciaclculadaRnPanel);
            app.OmegaLabel_3.FontSize = 18;
            app.OmegaLabel_3.Position = [199 64 32 24];
            app.OmegaLabel_3.Text = '$\Omega$';
            % Create GrficosdatosoriginalesconcorrelacinlinealPanel
            app.GrficosdatosoriginalesconcorrelacinlinealPanel = uipanel(app.UIFigure);
            app.GrficosdatosoriginalesconcorrelacinlinealPanel.TitlePosition = 'centertop';
            app.GrficosdatosoriginalesconcorrelacinlinealPanel.Title = 'Gráficos datos
originales con correlación lineal';
            app.GrficosdatosoriginalesconcorrelacinlinealPanel.BackgroundColor = [0.651
0.651 0.651];
            app.GrficosdatosoriginalesconcorrelacinlinealPanel.FontWeight = 'bold';
            app.GrficosdatosoriginalesconcorrelacinlinealPanel.Position = [511 434 660 290];
            % Create Graficov
```

```
app.Graficov = uiaxes(app.GrficosdatosoriginalesconcorrelacinlinealPanel);
title(app.Graficov, 'Gráfico voltaje y correlación')
xlabel(app.Graficov, 'Tiempo')
ylabel(app.Graficov, 'Voltaje')
app.Graficov.XGrid = 'on';
app.Graficov.YGrid = 'on';
app.Graficov.BackgroundColor = [0.651 0.651 0.651];
app.Graficov.Position = [11 20 300 240];
% Create Graficoc
app.Graficoc = uiaxes(app.GrficosdatosoriginalesconcorrelacinlinealPanel);
title(app.Graficoc, 'Gráfico corriente y correlación')
xlabel(app.Graficoc, 'Tiempo')
ylabel(app.Graficoc, 'Corriente')
app.Graficoc.XGrid = 'on';
app.Graficoc.YGrid = 'on';
app.Graficoc.BackgroundColor = [0.651 0.651 0.651];
app.Graficoc.Position = [350 20 300 240];
% Create GrficossintendenciaPanel
app.GrficossintendenciaPanel = uipanel(app.UIFigure);
app.GrficossintendenciaPanel.TitlePosition = 'centertop';
app.GrficossintendenciaPanel.Title = 'Gráficos sin tendencia';
app.GrficossintendenciaPanel.BackgroundColor = [0.651 0.651 0.651];
app.GrficossintendenciaPanel.FontWeight = 'bold';
app.GrficossintendenciaPanel.Position = [511 114 660 300];
% Create Graficovnt
app.Graficovnt = uiaxes(app.GrficossintendenciaPanel);
title(app.Graficovnt, 'Gráfico voltaje sin tendencia')
xlabel(app.Graficovnt, 'Tiempo')
ylabel(app.Graficovnt, 'Voltaje')
app.Graficovnt.XGrid = 'on';
app.Graficovnt.YGrid = 'on';
app.Graficovnt.BackgroundColor = [0.651 0.651 0.651];
app.Graficovnt.Position = [12 20 300 240];
% Create Graficocnt
app.Graficocnt = uiaxes(app.GrficossintendenciaPanel);
title(app.Graficocnt, 'Gráfico corriente sin tendencia')
xlabel(app.Graficocnt, 'Tiempo')
ylabel(app.Graficocnt, 'Corriente')
app.Graficocnt.XGrid = 'on';
app.Graficocnt.YGrid = 'on';
app.Graficocnt.BackgroundColor = [0.651 0.651 0.651];
app.Graficocnt.Position = [351 20 300 240];
% Create DatosdecorrelacinPanel
app.DatosdecorrelacinPanel = uipanel(app.UIFigure);
app.DatosdecorrelacinPanel.TitlePosition = 'centertop';
app.DatosdecorrelacinPanel.Title = 'Datos de correlación ';
app.DatosdecorrelacinPanel.BackgroundColor = [0.651 0.651 0.651];
```

```
app.DatosdecorrelacinPanel.FontWeight = 'bold';
app.DatosdecorrelacinPanel.Position = [15 140 480 157];
% Create mvs
app.mvs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.mvs.Editable = 'off';
app.mvs.HorizontalAlignment = 'center';
app.mvs.FontWeight = 'bold';
app.mvs.BackgroundColor = [0.651 0.651 0.651];
app.mvs.Position = [268 113 61 22];
% Create mcs
app.mcs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.mcs.Editable = 'off';
app.mcs.HorizontalAlignment = 'center';
app.mcs.FontWeight = 'bold';
app.mcs.BackgroundColor = [0.651 0.651 0.651];
app.mcs.Position = [268 83 61 22];
% Create bvs
app.bvs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.bvs.Editable = 'off';
app.bvs.HorizontalAlignment = 'center';
app.bvs.FontWeight = 'bold';
app.bvs.BackgroundColor = [0.651 0.651 0.651];
app.bvs.Position = [358 113 61 22];
% Create bcs
app.bcs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.bcs.Editable = 'off';
app.bcs.HorizontalAlignment = 'center';
app.bcs.FontWeight = 'bold';
app.bcs.BackgroundColor = [0.651 0.651 0.651];
app.bcs.Position = [358 83 61 22];
% Create EcuacinvoltajesimuladoLabel
app.EcuacinvoltajesimuladoLabel = uilabel(app.DatosdecorrelacinPanel);
app.EcuacinvoltajesimuladoLabel.Position = [48 113 144 22];
app.EcuacinvoltajesimuladoLabel.Text = 'Ecuación voltaje simulado';
% Create EcuacincorrientesimuladaLabel
app.EcuacincorrientesimuladaLabel = uilabel(app.DatosdecorrelacinPanel);
app.EcuacincorrientesimuladaLabel.Position = [48 83 156 22];
app.EcuacincorrientesimuladaLabel.Text = 'Ecuación corriente simulada';
% Create yLabel
app.yLabel = uilabel(app.DatosdecorrelacinPanel);
app.yLabel.FontWeight = 'bold';
app.yLabel.Position = [248 113 25 22];
app.yLabel.Text = 'y=';
% Create yLabel_2
app.yLabel_2 = uilabel(app.DatosdecorrelacinPanel);
app.yLabel_2.FontWeight = 'bold';
app.yLabel_2.Position = [248 83 25 22];
```

```
app.yLabel_2.Text = 'y=';
% Create xLabel
app.xLabel = uilabel(app.DatosdecorrelacinPanel);
app.xLabel.FontSize = 13;
app.xLabel.FontWeight = 'bold';
app.xLabel.Position = [330 113 28 22];
app.xLabel.Text = 'x  +';
% Create xLabel_2
app.xLabel_2 = uilabel(app.DatosdecorrelacinPanel);
app.xLabel_2.FontSize = 13;
app.xLabel_2.FontWeight = 'bold';
app.xLabel_2.Position = [330 83 28 22];
app.xLabel_2.Text = 'x  +';
% Create CoeficientedecorrelacinLabel
app.CoeficientedecorrelacinLabel = uilabel(app.DatosdecorrelacinPanel);
app.CoeficientedecorrelacinLabel.Position = [49 36 151 22];
app.CoeficientedecorrelacinLabel.Text = 'Coeficiente de correlación  ';
% Create CoeficientededeterminacinLabel
app.CoeficientededeterminacinLabel = uilabel(app.DatosdecorrelacinPanel);
app.CoeficientededeterminacinLabel.Position = [49 6 164 22];
app.CoeficientededeterminacinLabel.Text = 'Coeficiente de determinación ';
% Create VoltajeLabel_2
app.VoltajeLabel_2 = uilabel(app.DatosdecorrelacinPanel);
app.VoltajeLabel_2.FontWeight = 'bold';
app.VoltajeLabel_2.Position = [277 56 44 22];
app.VoltajeLabel_2.Text = 'Voltaje';
% Create CorrienteLabel_2
app.CorrienteLabel_2 = uilabel(app.DatosdecorrelacinPanel);
app.CorrienteLabel_2.FontWeight = 'bold';
app.CorrienteLabel_2.Position = [359 56 59 22];
app.CorrienteLabel_2.Text = 'Corriente';
% Create rvs
app.rvs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.rvs.Editable = 'off';
app.rvs.HorizontalAlignment = 'center';
app.rvs.FontWeight = 'bold';
app.rvs.BackgroundColor = [0.651 0.651 0.651];
app.rvs.Position = [268 36 61 22];
% Create rcs
app.rcs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
app.rcs.Editable = 'off';
app.rcs.HorizontalAlignment = 'center';
app.rcs.FontWeight = 'bold';
app.rcs.BackgroundColor = [0.651 0.651 0.651];
app.rcs.Position = [358 36 61 22];
% Create rrvs
app.rrvs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
```

```
            app.rrvs.Editable = 'off';
            app.rrvs.HorizontalAlignment = 'center';
            app.rrvs.FontWeight = 'bold';
            app.rrvs.BackgroundColor = [0.651 0.651 0.651];
            app.rrvs.Position = [268 6 61 22];
            % Create rrcs
            app.rrcs = uieditfield(app.DatosdecorrelacinPanel, 'numeric');
            app.rrcs.Editable = 'off';
            app.rrcs.HorizontalAlignment = 'center';
            app.rrcs.FontWeight = 'bold';
            app.rrcs.BackgroundColor = [0.651 0.651 0.651];
            app.rrcs.Position = [358 6 61 22];
            % Create r2Label
            app.r2Label = uilabel(app.DatosdecorrelacinPanel);
            app.r2Label.FontSize = 16;
            app.r2Label.Position = [210 7 30 25];
            app.r2Label.Text = '$r^2$';
            % Create rLabel
            app.rLabel = uilabel(app.DatosdecorrelacinPanel);
            app.rLabel.FontSize = 16;
            app.rLabel.Position = [193 38 25 22];
            app.rLabel.Text = '$r$';
            % Create BotonesdeejecucinPanel
            app.BotonesdeejecucinPanel = uipanel(app.UIFigure);
            app.BotonesdeejecucinPanel.TitlePosition = 'centertop';
            app.BotonesdeejecucinPanel.Title = 'Botones de ejecución';
            app.BotonesdeejecucinPanel.BackgroundColor = [0.651 0.651 0.651];
            app.BotonesdeejecucinPanel.FontWeight = 'bold';
            app.BotonesdeejecucinPanel.FontSize = 13;
            app.BotonesdeejecucinPanel.Position = [511 14 660 80];
            % Create Buscaxls
            app.Buscaxls = uibutton(app.BotonesdeejecucinPanel, 'push');
            app.Buscaxls.ButtonPushedFcn = createCallbackFcn(app,
@BuscaxlsButtonPushed, true);
            app.Buscaxls.BackgroundColor = [0.702 0.698 0.6941];
            app.Buscaxls.FontWeight = 'bold';
            app.Buscaxls.Position = [52 11 124 33];
            app.Buscaxls.Text = 'CARGAR DATOS';
            % Create CALCULARButton
            app.CALCULARButton = uibutton(app.BotonesdeejecucinPanel, 'push');
            app.CALCULARButton.ButtonPushedFcn = createCallbackFcn(app,
@CALCULARButtonPushed, true);
            app.CALCULARButton.BackgroundColor = [0.702 0.702 0.6902];
            app.CALCULARButton.FontWeight = 'bold';
            app.CALCULARButton.Position = [279 11 124 33];
            app.CALCULARButton.Text = 'CALCULAR';
            % Create RESETButton
```

```
            app.RESETButton = uibutton(app.BotonesdeejecucinPanel, 'push');
            app.RESETButton.ButtonPushedFcn = createCallbackFcn(app,
@RESETButtonPushed, true);
            app.RESETButton.BackgroundColor = [0.702 0.702 0.6902];
            app.RESETButton.FontWeight = 'bold';
            app.RESETButton.Position = [480 11 124 33];
            app.RESETButton.Text = 'RESET';
            % Create lampcar
            app.lampcar = uilamp(app.BotonesdeejecucinPanel);
            app.lampcar.Position = [11 38 11 11];
            app.lampcar.Color = [1 0 0];
            % Create Esclog1
            app.Esclog1 = uicheckbox(app.UIFigure);
            app.Esclog1.Text = 'Escala logarítmica';
            app.Esclog1.Position = [524 413 120 22];
            % Create Esclog2
            app.Esclog2 = uicheckbox(app.UIFigure);
            app.Esclog2.Text = 'Escala logarítmica';
            app.Esclog2.Position = [913 410 120 22];
            % Create Esclog3
            app.Esclog3 = uicheckbox(app.UIFigure);
            app.Esclog3.Text = 'Escala logarítmica';
            app.Esclog3.Position = [522 93 120 22];
            % Create Esclog4
            app.Esclog4 = uicheckbox(app.UIFigure);
            app.Esclog4.Text = 'Escala logarítmica';
            app.Esclog4.Position = [913 93 120 22];
            % Show the figure after all components are created
            app.UIFigure.Visible = 'on';
        end
    end
    % App creation and deletion
    methods (Access = public)
        % Construct app
        function app = cargdat
            % Create UIFigure and components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
```

```
        end
    end
end
```

VOTOS DE APROBATORIOS

Secretaria Ejecutiva del Instituto de Investigación en Ciencias Básicas Aplicadas de la Universidad Autónoma del Estado de Morelos.
P r e s e n t e.

Por medio de la presente le informamos que después de revisar la versión escrita de la tesis que realizó el C. **ANGELES CARLOS JORGE MISAEL** con número de matrícula **20164000521** cuyo título es:

"DESARROLLO DE INTERFAZ GRÁFICA DE SOFTWARE LIBRE PARA REALIZAR ANÁLISIS DE RUIDO ELECTROQUÍMICO".

Consideramos que **SI** reúne los méritos que son necesarios para continuar los trámites para obtener el título de **Licenciado en Tecnología Área Terminal en Física Aplicada.**

Cuernavaca, Mor a 18 de agosto de 2022

Atentamente
Por una universidad culta

Se adiciona página con la e-firma UAEM de los siguientes:

DR. JOSE GONZALO GONZÁLES RODRIGUEZ	**PRESIDENTE**
DR. OMAR PALILLERO SANDOVAL	**SECRETARIO**
DRA. ARIANNA PARRALES BAHENA	**VOCAL**
DR. ROY LOPEZ SESENES	**SUPLENTE**
DR. JOSE ALFREDO HERNANDEZ PEREZ	**SUPLENTE**

Se expide el presente documento firmado electrónicamente de conformidad con el ACUERDO GENERAL PARA LA CONTINUIDAD DEL FUNCIONAMIENTO DE LA UNIVERSIDAD AUTÓNOMA DEL ESTADO DE MORELOS DURANTE LA EMERGENCIA SANITARIA PROVOCADA POR EL VIRUS SARS-COV2 (COVID-19) emitido el 27 de abril del 2020.

El presente documento cuenta con la firma electrónica UAEM del funcionario universitario competente, amparada por un certificado vigente a la fecha de su elaboración y es válido de conformidad con los LINEAMIENTOS EN MATERIA DE FIRMA ELECTRÓNICA PARA LA UNIVERSIDAD AUTÓNOMA DE ESTADO DE MORELOS emitidos el 13 de noviembre del 2019 mediante circular No. 32.

Sello electrónico

JOSE GONZALO GONZALEZ RODRIGUEZ | Fecha:2022-11-04 12:07:10 | Firmante

Qlv9CsLNQaZJr/28dt1oQvdDjsYUK8EUxSuqmWA4bcNSy9yGQmfc1j4cDKKmjS+WpD+5wsxsTjyxmqK7/K7Vd7wuGYg3AjaEPEEIjBVgwWlFAwKZHa3e11nGBbF/3gm27/5rzpgPZjitrhhx8ukhpY2f6hhmjrBAl6E81/+3i2EjMSdUP7fV87r12j4gUqvK1Qui3X5aQJjd9KhdJop7157KdDUXm6ZK5QsvDanysOc34Z2xvfQOHMsPnYYlcR5kCd3Het7VB0a7hxU1hCiS7xDMREzRADzvsW131aiZmVhFD9seAnbpnDJr9kPX4PR5N2/r88s/ETlg6pAIEHSciw==

OMAR PALILLERO SANDOVAL | Fecha:2022-11-04 12:34:30 | Firmante

F7yxQz1OyIl+zfxpKUAFrV2MdQjsE2wFyHfvvp1P+qtnNjZsqrO+mucM2vWM7g5rttBrNDgTT545c+lyK/ve65ivEK5i1stqKNKq9MAhU6FBUxJKPw0podS0k21q6/xXI1MlXNXhMjDdiznhz1QssE3Ylx/v7HFIlKkq5ulQxu+/S95j9E/Rt6Kg6k86ggqHSx9aF1CBgoH4t0oyHjfPjyXbVrFe3BSHx2TC+6qXHv7+xJJiXO5UvGBgmEf5Xk30JG2fwfIHOJqkH8L8qiNTmE7GRpnb2Vw6igLD3Ux1hButHoIx07tlRTo4HM4N7QzwzrJnUvKFOZKAysbipCPOQw==

JOSE ALFREDO HERNANDEZ PEREZ | Fecha:2022-11-04 12:43:03 | Firmante

kAEtt0aR4cPR1LpgistHhamoKQeAySy7fU9D+ei05zm0eeS8sbiIiZvuvi+zxwbNt1J6rW/Ef/N5iv8TweQkbflrzTa4SScn7QDZQGiw9roIUyeZCDRMwvtHMlFcI6Cb5fuElyY4VTCJnm+BK35KY5kbzJul4O2B6X/VmkQxbikOCKI3rUjEqUeCDEbgNVUuEzasYXE17E6Ap6K2wQKmZwn0sojYiFIBx2xlWQmE37w2Msg3stgvXTKbLBUTUX2J0NYtSPC3BExb1JlCrBr9lKMsJ/fd4A3kLYGcct35xTY2n6CqpfkqC7K5YQHFP7B3s1fylvF98ODOJSh23NW8pg==

ROY LOPEZ SESENES | Fecha:2022-11-04 13:36:35 | Firmante

GB4DumFGbw4+vNhRFhWy3l8MgEIPedxqkFBNOOJ/4rUxXH8+C7Uk5OVnIi21MxEKxmMYjoRDBv02AMtcRSQcPBFlEJgytwrMdLI7Wy8MftMgOglSG8a2vefOPxgAfQ/Um8aFVvtaYDDTThVJJFl5cx8lMaQMTlNSZPjFBAcLPHVOQgkMcaFgFHo4Af10hRhMFsVZqa7D6EEP6XQ39tCivGzWuO7EHDDClMzSqZUWkBnjzg1iwqr5Sha0wABYQN4TIudwt9dxh/P4Hx7UOTcbty6itmUc1u2ZI5LqYEYyBCgPbBDXyjBA1ettgONY6fh6//CYm2w8CBMeAUbwRAQ0mw==

ARIANNA PARRALES BAHENA | Fecha:2022-11-07 07:52:38 | Firmante

lQ3cpLmfpz6xLbJC58NgvklzxRi5pbsR9U43nhkVae6STqZ7fgXZuyP4wj1cznjG4n+jg9XaqhYFSYuJF2Cw8u63lelWAAiYw40TVnPYzHlLHf3aRvGNP09qZBz3i50MtssFwuOJHNsMddlxijCF8C8og2vsDzIg7fQOZzbFWORjEku/vR3vbT917T/11zRUx6yjLJlpIRd4gPLPxnlsaaqcIw74ffvsBZHgJk1tHHM9MLt+QOEoZel5yvHxQa9By+KL02X63jw07f7Jtt3Y+FBDFCK2+QZ/xrvkZZqfDZLuRGQpjgJWgWfskhdvTyOTMkTjRiDrSw6keckUvAjLjg==

Puede verificar la autenticidad del documento en la siguiente dirección electrónica o escaneando el código QR ingresando la siguiente clave:

ewRPazyMH

https://efirma.uaem.mx/noRepudio/n0qxfVUB6A6gp46K5NmFlW9i0b59MCSy