UNIVERSIDAD AUTÓNOMA DE QUERÉTARO

24 DE OCTUBRE 2024

PRÁCTICA 3: Exploración en BMS y filtrado de señal

MATERIA: OPTATIVA II

Semestre VIII

INGENIERÍA MECÁNICA Y AUTOMOTRIZ

Estudiantes:

Pérez Urbano Brandon Josué	**301838**
Melendrez Robles Jesús Esteban	**301824**

I. INTRODUCCIÓN

En la presente práctica, se realiza la adquisición y análisis de datos provenientes del auto eléctrico “E-FACI” desarrollado por la universidad. El objetivo principal es la adquisiciónn de las señales de corriente mediante el uso del sistema de gestión de batería **Orion BMS**, con el fin de evaluar el comportamiento de la batería bajo condiciones de aceleracion y desaceleración. Para la adquisición de datos se empleó el protocolo **Controller Area Network (CAN)**, ampliamente utilizado en la industria automotriz para la comunicación entre los distintos módulos del vehículo eléctrico.

II. FUNDAMENTACION TEORICA

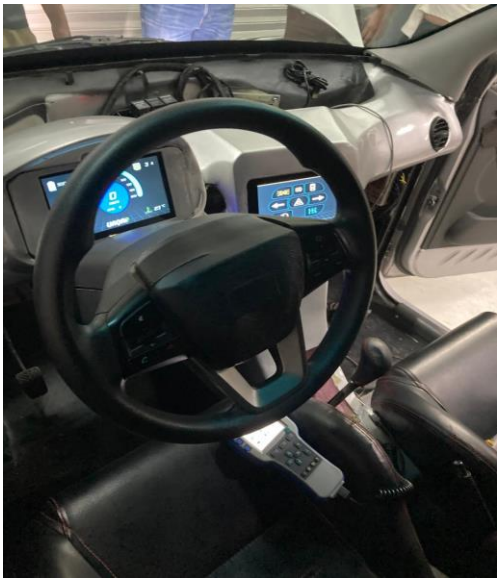
- *Protocolo CAN (Controller Area Network)*: Estándar de comunicación serie que fue desarrollado por Bosch en los años 80 para resolver los desafíos de comunicación entre múltiples dispositivos o nodos en sistemas distribuidos, especialmente en la industria automotriz. CAN permite que las unidades de control electrónico (ECU) de un vehículo se comuniquen sin necesidad de una computadora central. [1]
- *Modelo de comunicación en bus*: CAN utiliza un bus de comunicación de dos cables (CAN_H y CAN_L), donde todos los nodos están conectados y pueden comunicarse. La señal es diferencial, lo que mejora la inmunidad al ruido electromagnético, un factor crucial en entornos industriales o automotrices. Esto garantiza que la comunicación sea robusta incluso en entornos ruidosos. [2]
- *Orion BMS*: El Orion Battery Management System (BMS) es un sistema avanzado diseñado para monitorear y gestionar baterías de iones de litio en vehículos eléctricos. Este sistema controla parámetros como el voltaje de las celdas, la corriente de carga y descarga, y la temperatura para asegurar el funcionamiento seguro y eficiente de la batería. Además, el Orion BMS se comunica a través del bus CAN, permitiendo la integración con otros sistemas del vehículo para reportar el estado de la batería en tiempo real. [3]
- *Filtrado de señales:* El filtrado de señales es un proceso utilizado para eliminar o atenuar componentes indeseables (ruido) en una señal, permitiendo obtener una representación más clara de los datos de interés. En sistemas eléctricos, se utilizan filtros digitales o analógicos para procesar las señales adquiridas y eliminar el ruido de alta frecuencia o cualquier otra distorsión que pueda afectar el análisis de las señales de corriente, voltaje, etc. [4]
- *MATLAB:* Es un entorno de programación utilizado para la computación numérica, visualización y análisis de datos. Es ampliamente empleado en la ingeniería para tareas como la simulación de sistemas, el procesamiento de señales y la creación de algoritmos. [5]
- *Filtro Pasa Bajas:* El filtro pasa bajas actúa como un bloqueador de frecuencias superiores a un cierto umbral llamado frecuencia de corte. Las señales con frecuencias por debajo de esta frecuencia de corte pasan sin atenuación significativa, mientras que las señales con frecuencias más altas son debilitadas o eliminadas. Este tipo de filtro es fundamental en muchas aplicaciones de procesamiento de señales, como la eliminación de ruido en

- sistemas de adquisición de datos, el suavizado de señales, o en sistemas de comunicación para evitar aliasing.

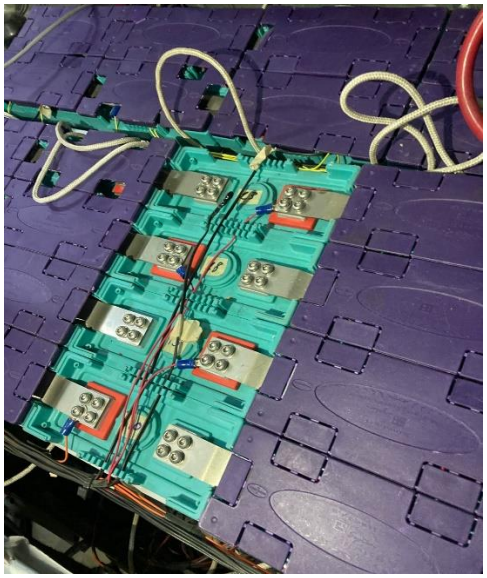
¿Qué es el proyecto EFACI?

Figura 1. EFACI

El vehículo EFACI (figura 1) es un proyecto vehículo, el cual tiene la conversión de un auto ATOS de combustión interna a un vehículo con motor completamente eléctrico y con una apariencia totalmente distinta.



a)



b)

c)

d)

Figura 2. Modificaciones EFACI

Dentro de las modificaciones mas notorias del proyecto EFACI se resalta:

- Interior dos pantallas una mostrando parámetros de velocidad, revoluciones, temperatura, y porcentaje de carga disponible (Figura 2 a).
- Alimentación con 32 celdas de batería conectadas en serie logrando una alimentación de 102.4V (Figura 2 b).
- Conversión física modificación de estructura original (Figuras 2 c y d)

III. OBJETIVO

Adquirir una señal del auto eléctrico y después procesarla mediante procesamiento matemático por medio del software Matlab

IV. DESARROLLO

Encendido del vehículo.

Para iniciar con la práctica es necesario encender el vehículo y para esto se debe colocar la llave del auto en el sensor de proximidad que se encuentra entres los asiente del piloto y copiloto, se logra identificar si se coloca de manera adecuada porque el vehículo empieza a emitir una alerta auditiva (similar a cuando se dejan las llaves pegadas a un carro normal y se tiene el motor apagado). Una vez las llaves están colocadas en su lugar, se debe presionar el freno y oprimir el botón de encendido (el cual está ubicado detrás del volante del lado derecho) y al realizar esto, se escuchan que actúan algunos relevadores, el auto empieza a hacer algunos zumbidos y las pantallas del automóvil encienden.

Conexión al BMS.

Para adquirir datos directo en la computadora, se debe instalar el programa “Orion BMS Utilitu 2”, que se encuentra disponible en la página oficial del proveedor, ya teniéndolo instalado, la computadora se conecta al cable gris que se encuentra del lado del copiloto y antes de empezar a usarlo se debe desactivar la conexión bluetooth ya que se pueden producir problemas e conexión al tenerlo activo durante el uso del programa. Al iniciar el programa se muestra la interfaz que se aprecia en la figura 3.

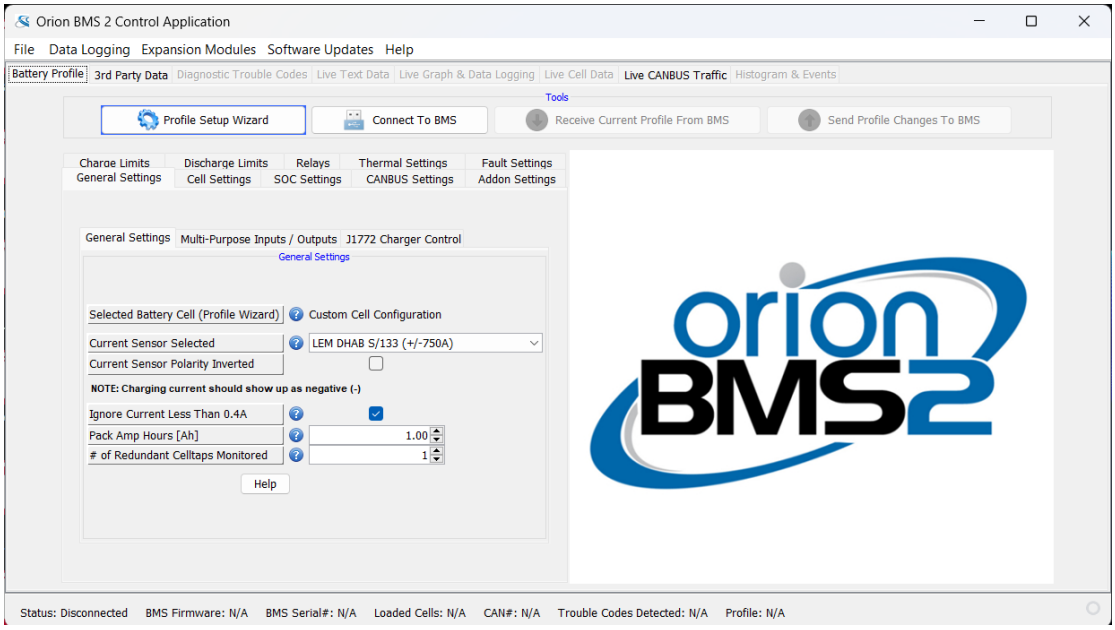


Figura 3. Interfaz del programa “Orion BMS 2 Utility”.

Para hacer una correcta conexión con el BMS, se hace clic en la opción “Connect to BMS” y se abre una ventana como la que se muestra en la figura 4, en la cual para la primera opción “Selected CANdapter”, se debe seleccionar la opción del puerto en la que señale que se ha detectado el sistema de comunicación del BMS, y en la segunda opción, donde se lee “CAN Baud Rate” se selecciona la opción “250 kbit/sec” y una vez teniendo ambas opciones como se indican se da clic en el botón “Connect”.

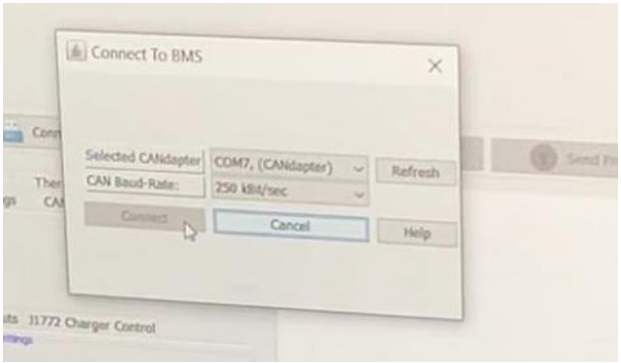
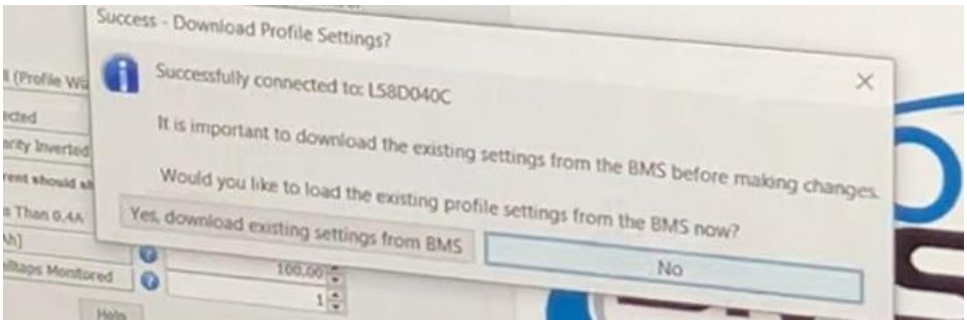


Figura 4. Configuración de la conexión.

Posteriormente hay que esperar unos segundos hasta que salga la ventana en la que indica que el programa de la computadora se ha conectado satisfactoriamente al modulo BMS que está instalado en el carro además que preguntara si se desean cargar las configuraciones de perfil, a lo que simplemente se contesta que no. (Figura 5)



Figuran 5. Ventana para cargar configuraciones existentes

Después hay que situarse en la segunda pestaña que ofrece el programa llamada “3rd Party Device”, que en español se puede traducir como “datos de terceros” o de manera más entendible “datos de dispositivos que no son de la marca del controlador”, en donde se muestra una pantalla como la mostrada en la figura 6a) y estando ahí, se da clic en el botón “Connect to Device” una vez hecho esto se muestra la pantalla mostrada en la figura 6b), en la cual se puede observar que ya se encuentran algunos datos desplegados en la tabla, los cuales son datos en vivo.

a) b)

Figura 6. Pestaña “3rd Party Device”, a) antes de conectarse al dispositivo y b) después de conectarse

Si se da clic en la tercer pestaña, en la que se lee “Diagnostic Trouble Codes”, se muestra una ventana como la que se observa en la figura 7, en donde se pueden ver tres recuadros, el de la izquierda enumera códigos de falla y algunas letras, de las cuales se puede saber su significado ya que están desglosadas en la parte inferior, en el recuadro de en medio se ofrece información más detallada del código de falla, para que se muestre dicha información se debe seleccionar el código del que se quiera saber más y por último, el recuadro de la derecha en lista las celdas de batería. A si mismo, hasta abajo se puede ver un botón en el que se lee “Clear All Codes” el cual sirve para eliminar dichos códigos de falla.

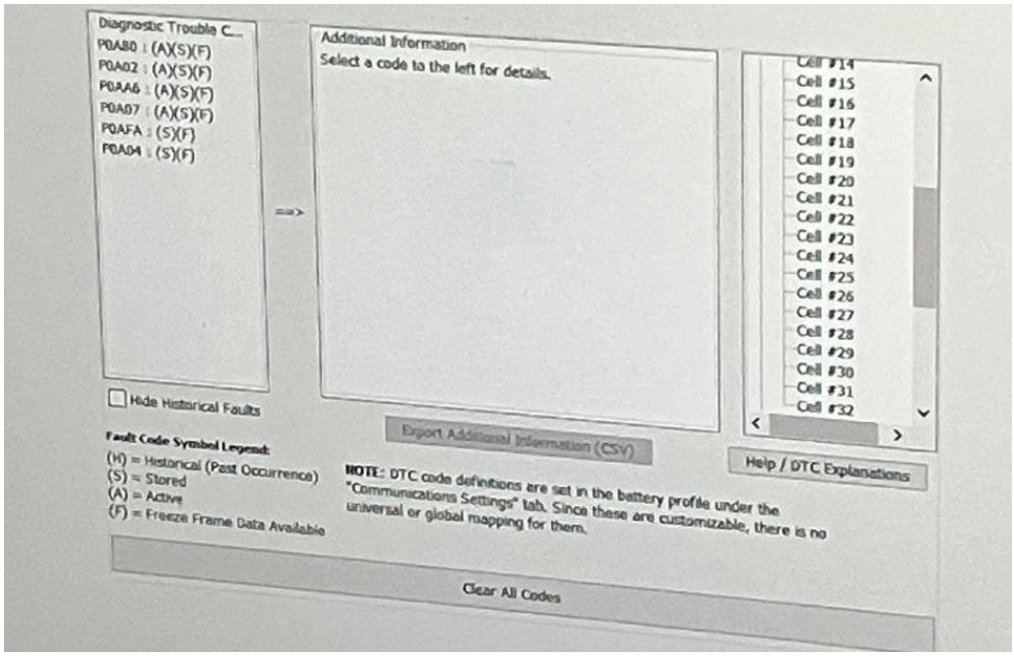


Figura 7. Información mostrada en la pestaña “Diagnostic Trouble Codes”

Continuando con la exploración de las pestañas, se abre la llamada “Live Text Data”, en la cual, como su nombre indica, se pueden ver datos en vivo tales como el estado de carga, el valor de voltaje mas bajo de las celdas, así como también el mas alto, la corriente consumida, el tiempo transcurrido desde su encendido, etc. Esta información viene mostrada en una tabla de tres columnas, en la primera se señala el parámetro medido, en la segunda el valor y en la tercer las unidades de cada uno de estos (en caso de que aplique), además en la parte inferior se puede ver que se pueden seleccionar distintos tipos de parámetros y otros botones para mostrar los comandos activos, para exportar los parámetros en vivo y para obtener ayuda. (Figura 8)

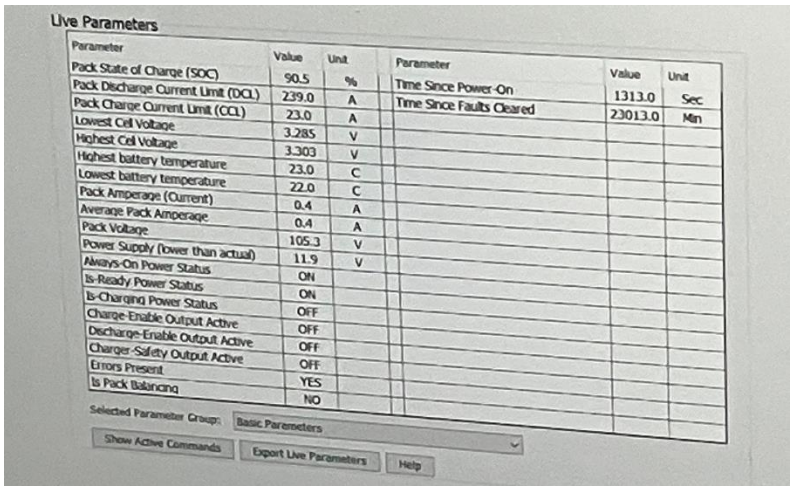


Figura 8. Información mostrada en la pestaña “Live Text Data”

La siguiente ventana por explorar es “Live Graph & Data Logging”, en la cual se muestra una ventana donde se grafican los datos seleccionados y para seleccionarlos hay que hacer un clic en el botón que se lee “Select Parameters to Graph” y se puede ver la gráfica en la ventana mencionada, en esta pestaña es de donde se obtienen los datos para posteriormente procesarlos y para empezar a grabarlos se da clic en el botón “Start Loggong” que se encuentra del lado izquierdo y una vez que ha pasado el tiempo que se desea monitorear la o las señales seleccionadas se da clic en el botón “Stop Logging” que se encuentra a un costado del presionado anteriormente. Cabe mencionar que para esta práctica se seleccionaron os datos con los nombres “Pack Amperage (Current)” y “Pack Amp-hours” y durante la grabación de estos datos en la parte inferior se puede notar una barra verde como de cargando y mientras esta esté activa se sabe que se están grabando datos de las señales seleccionadas. Para poder guardar un archivo con los datos, después de detener la grabación se da clic en el botón “Export Collected Data” y se selecciona la carpeta y el nombre con el que se quiera guardar. (Figura 9)

Figura 9. Grabación de datos de parámetros seleccionados en la pestaña “Live Graph & Data Logging”

A continuación, se explora la pestaña “Live Cell Data”, en la que se muestra todo tipo de información referente a las celdas de las baterías, el voltaje mas alto, el mas bajo, el diferencial de voltaje, la temperatura, el porcentaje de carga, la resistencia, etc. Además, que muestra información por separado para cada una de las celdas con las que cuenta el automóvil, en este caso son 32, de igual manera como en la pestaña vista anteriormente, se pueden exportar los datos en vivo. Para un mejor entendimiento de lo explicado en este párrafo, refiérase a la figura 10.

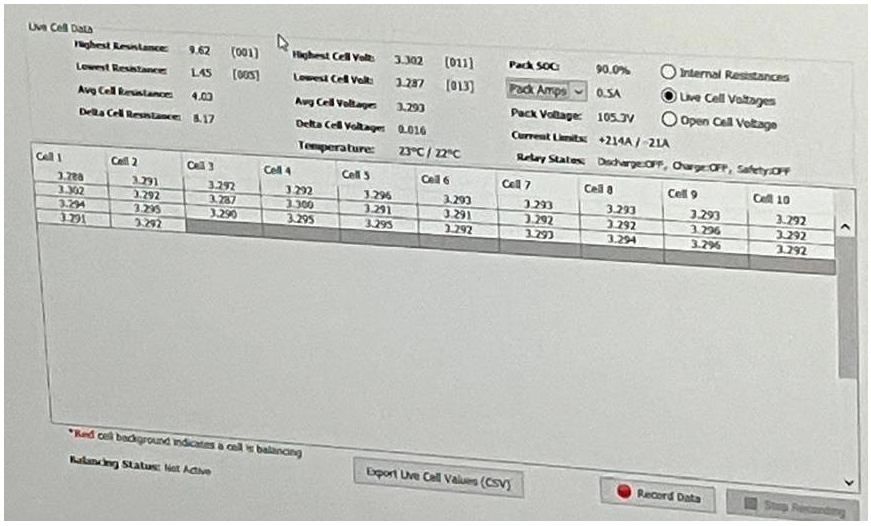


Figura 10. Pestaña “Live Cell Data”

Por último, se le echa un vistazo a la pestaña “Live CANBUS Traffic” en la cual se muestra una ventana donde se muestran los paquetes de datos que se envían del BMS a la computadora, es decir, son palabras que en lenguaje CAN para decirle todo lo que se pudo apreciar en las pestañas anteriores, es decir, la información como corriente, voltaje, temperatura, etc. Se manda mediante esos códigos mostrados en esta pantalla, sin olvidar mencionar que cada fila que se muestra en la figura 11 es un nodo distinto que está conectado al CANBUS.

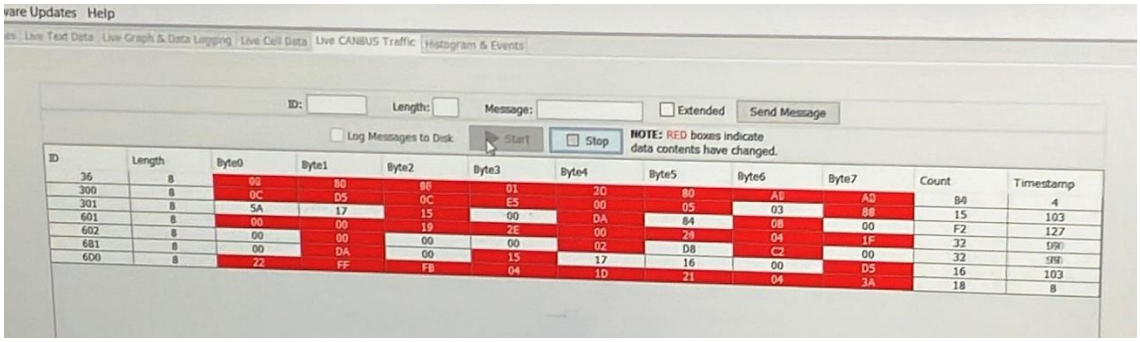


Figura 11. Paquetes de datos enviados a través de lenguaje CAN, vistos en la pestaña “Live CANBUS Traffic”

Finalmente para desconectar la computadora de manera segura del BMS, es necesario hacer clic en “Disconnect from BMS” para que se desconecte de manera segura y no haya ningún riesgo para ninguna de las partes, dicha opción se encuentra en la opción “File” ubicada en la parte superior derecha de la interfaz de la aplicación, tal y como se muestra en la figura 12.

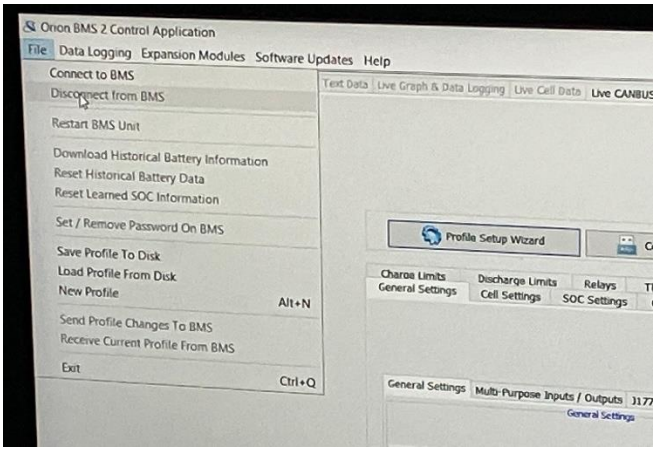


Figura 12. Desconexión segura del BMS

Procesado de la señal obtenida.

Ya que se ha explorado la aplicación y exportado una serie de datos, lo que falta es graficarlos y aplicarles un filtro para eliminar cualquier señal de interferencia o ruido, para esto es necesario abrir un script nuevo en MATLAB y como buena práctica, las primeras líneas de código serán las que se muestran en la figura 13, las cuales sirven para limpiar las variables, limpiar el "Command Window" y cerrar todas las figuras y de esta manera el código se ejecute de la manera más optima posible.

```
1    clc;
2    clear;
3    close all;
```
Figura 13. Primeras líneas del código

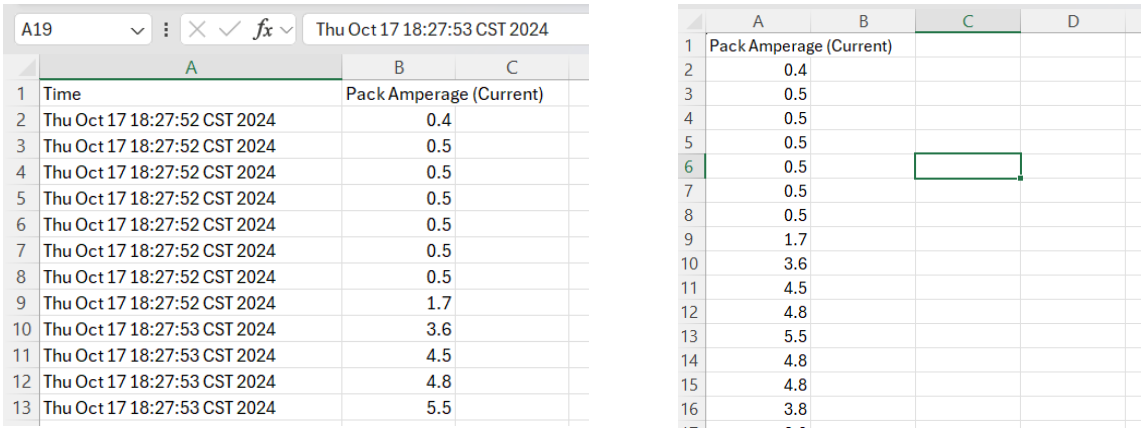
Posteriormente, para leer los datos en el archivo donde se guardaron los valores dados por el BMS, se usa la siguiente línea de código:

```
A=readtable("orionbmsBRANDON_log_2024-10-17-18-20-51.csv", 'VariableNamingRule',
'preserve');
```

La cual guarda los datos encontrados en el archivo en una variable llamada "A" usando la función "readtable" a la cual se le dan tres parámetros, tales como el nombre del archivo (el cual debe estar guardado en la misma carpeta que el scrip), el "VariableNamingRule", el cual indica que la tabla que se lee tiene un encabezado que designa el nombre del conjunto de datos de la columna y "preserve" el cual indica que se guarde dicho nombre tal y como se tiene en el archivo.

Hasta ahora todo parece ir como debe de ser, pero al correr el código marca un error y esto se debe a los datos contenidos en el archivo, al abrirlo se puede notar que no solo guardó los valores que se le solicitó en su momento si no que el archivo también almacena la fecha y

hora a la que se obtuvo cada dato, esto ayuda a saber la frecuencia de muestreo que ofrece el módulo y es de ayuda a la hora de graficar, y analizando cuantas muestras hay por segundo se puede saber que se toman 10 muestras por segundo, dato que mas adelante es de importancia, por lo pronto, desde el archivo original abierto con Excel, se elimina dicha columna y se guardan los cambios. (Figura 14)



A19 | Thu Oct 17 18:27:53 CST 2024

	A	B	C
1	Time	Pack Amperage (Current)	
2	Thu Oct 17 18:27:52 CST 2024	0.4	
3	Thu Oct 17 18:27:52 CST 2024	0.5	
4	Thu Oct 17 18:27:52 CST 2024	0.5	
5	Thu Oct 17 18:27:52 CST 2024	0.5	
6	Thu Oct 17 18:27:52 CST 2024	0.5	
7	Thu Oct 17 18:27:52 CST 2024	0.5	
8	Thu Oct 17 18:27:52 CST 2024	0.5	
9	Thu Oct 17 18:27:52 CST 2024	1.7	
10	Thu Oct 17 18:27:53 CST 2024	3.6	
11	Thu Oct 17 18:27:53 CST 2024	4.5	
12	Thu Oct 17 18:27:53 CST 2024	4.8	
13	Thu Oct 17 18:27:53 CST 2024	5.5	

a)

	A	B	C	D
1	Pack Amperage (Current)			
2	0.4			
3	0.5			
4	0.5			
5	0.5			
6	0.5			
7	0.5			
8	0.5			
9	1.7			
10	3.6			
11	4.5			
12	4.8			
13	5.5			
14	4.8			
15	4.8			
16	3.8			

b)

Figura 14. Archivo obtenido de la grabación de datos en vivo, a) original y b) con fecha y hora eliminadas

Ya que se han guardado los cambios y se vuelve a correr el código, esta vez se leen los datos sin problemas e indica una cantidad de datos de 677, con lo cual ya se tienen los datos suficientes para generar el vector de tiempo y poder graficar los datos, si se tienen 677 datos y se tomaban 10 en un segundo, quiere decir que se obtuvieron datos durante 67.7 segundos. Por lo tanto se genera un vector de tiempo con el siguiente código:

```
t=0.1:0.1:67.7;
```

Donde el primero 0.1 es desde donde se inicia la cuenta, el segundo 0.1 indica los saltos que va tomando, es decir va de 0.1 en 0.1 y finalmente el 67.7 es el valor final por alcanzar, ahora se deben guardar los datos de la columna en otra variable a la cual se la ha llamado “Amp”, para eso se usa la siguiente línea:

```
Amp=table2array(A(:,1));
```

Dicha línea, convierte la tabla en un arreglo numérico, y los valores que se les da es primero la tabla de donde se van a sacar los datos, las filas, que en este caso como se quiere obtener todas se escriben dos puntos, y después la columna de la que se quieren tomar los datos.

Resta filtrar la señal obtenida con un filtro pasa bajas con la siguiente línea de Código:

```
Amp_F = lowpass(Amp, 15*pi/180,10 );
```

En donde, “Amp_F” es la variable en donde se guardan los datos de la señal filtrada y la función “lospass” aplica un filtro pasa bajas según los datos que se le den, en este caso, se le dan “Amp” que es la señal por filtrar, “15*pi/180” es la frecuencia de corte y “10” es la frecuencia de muestreo de la señal. Y finalmente se grafican con las líneas de código de la figura 15.

```
13    subplot(2,1,1);
14    plot(t,Amp);
15    xlabel('Time (s)');
16    ylabel('Ampers');
17    title('Original Pack Amperage (Current)');
18    subplot(2,1,2);
19    plot(t,Amp_F,'g');
20    xlabel('Time (s)');
21    ylabel('Ampers');
22    title('Filtred Pack Amperage (Current)')
23
```

Figura 15. Código para graficar la señal original y la señal filtrada.

En la figura 15 se observan líneas de código que prácticamente se repiten esto se debe a que son dos graficas las que se quieren generar y en si se usan los mismos comandos, en los cuales “subplot(2,1,1)” y “subplot(2,1,2)” (líneas 13 y 18) es para generar varias gráficas en la misma ventana, donde el 2, indica el número de graficas ordenas en las filas, el primer 1, se refiere a la cantidad de gráficas ordenadas en columnas en la ventana y el número del final 1 y 2 (línea 13 y 18), respectivamente indica el orden de las gráficas; para “plot(t,Amp)” y “plot(t,Amp_F, ’g’)” son para hacer la gráfica y se le indican los datos del eje X, que sería “t” y los datos del eje Y, que serían “Amp” y “Amp_F” pero en el plot de la línea 19 se ve un dato de mas el cual es ‘g’, el cual solamente es para darle el color verde a la gráfica. Las siguientes líneas (15-17 y 20-22) son para darle etiquetas a los ejes y titulo a la gráfica.

V. RESULTADOS

Como resultados se obtienen las gráficas mostradas en la figura 16.

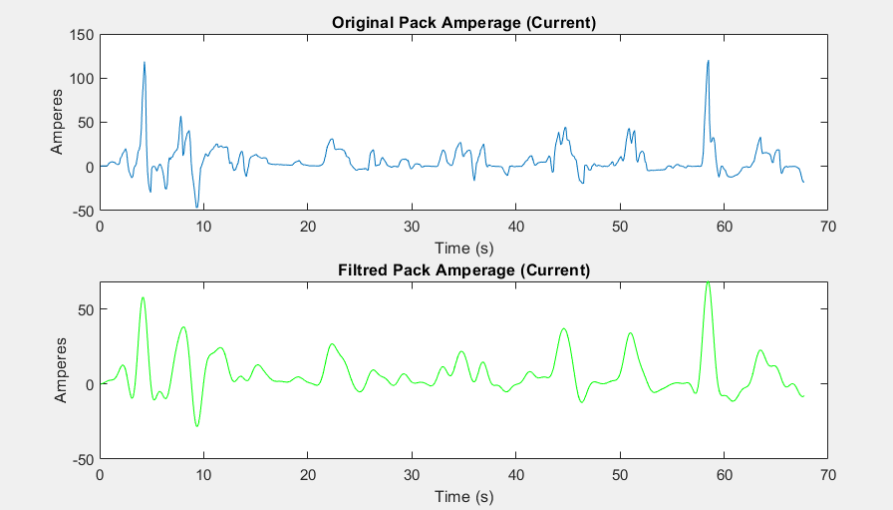


Figura 16. Graficas de la señal y de la señal filtrada.

En la figura 16 se muestran la gráfica original en azul y la gráfica filtrada y se pueden observar como se suaviza la señal y los picos máximos no toman valores tan altos, en si, una gráfica más suave. Es preciso indicar que se puede identificar los momentos de tiempo en los que se presentó una aceleración observando los picos de la señal y los momentos en los que hubo una desaceleración en las bajadas de corriente.

NOTA: En sí, se debieron haber exportado dos señales distintas con ayuda del BMS, pero hubo un problema al exportar los datos y solo se exportó una con éxito.

VI. CONCLUSIONES

Brandon: En esta práctica se experimentó lo visto en clase, además de aprovechar las ventajas del coche eléctrico EFACI el cual nos permitió obtener datos para posteriormente analizarlos en Matlab y filtrar la señal obtenida.
Se concluye de manera exitosa esta práctica examen esperando tener más prácticas con el uso de CAN para lograr entenderlo de mejor forma

Esteban. Se concluye con éxito la práctica ya que se adquirieron datos para el monitoreo del comportamiento de la batería del Vehículo Eléctrico EFACI, utilizando el Orion BMS y el protocolo CAN. La adquisición de señales de corriente a través del CAN bus permitió obtener información valiosa sobre el uso o consumo de corriente de las baterías mediante aceleraciones en un estado estático del vehículo.
Mediante el procesamiento de las señales en MATLAB, se aplicaron técnicas de filtrado digital, como el filtro pasa bajas, para eliminar el ruido presente en las señales adquiridas. Esto permitió obtener una representación más precisa y suave de los datos, mejorando así el análisis del consumo de corriente bajo condiciones de aceleración y desaceleración.
Aunado a lo mencionado, fue interesante ver cómo se presentan en vivo la comunicación CAN y la velocidad en la que varían los paquetes de datos enviados.

VII. REFERENCIAS

[1] R. Bosch GmbH, *CAN Specification Version 2.0*, Robert Bosch GmbH, 1991.

[2] M. Migliavacca, R. Obermaisser, and H. Kopetz, "A comparison of CAN and TTP," in *Proc. 2005 IEEE Int. Conf. Ind. Informatics*, 2005, pp. 39-44.

[3] A. Orion BMS, "Battery Management System - Product Overview," Available: https://www.orionbms.com. [Accessed: 24-October-2024].

[4] S. K. Mitra, *Digital Signal Processing: A Computer-Based Approach*, 4th ed. New York: McGraw-Hill, 2011.

[5] MathWorks, "MATLAB - The Language of Technical Computing," Available: https://www.mathworks.com/products/matlab.html. [Accessed: 24-October-2024].

[6] V. A. W. Hillier, *Fundamentals of Motor Vehicle Technology*. Nelson Thornes, 2006.