Laboratorio Phyphox

Lina Fernanda Guio Cabrera
est.lina.guio@unimilitar.edu.co

***Resumen*—**Se utilizaron los sensores del teléfono Samsung A35 mediante la aplicación Phyphox para realizar dos experimentos. Primero, se probó el sensor de inclinación, se observaron sus gráficas y se exportaron los datos a .csv para analizarlos en Python y MATLAB, permitiendo comparar los resultados. Luego, se aplicó el experimento Optical Stopwatch, que mide intervalos de tiempo a partir de variaciones en la iluminación detectadas por el sensor de luz ambiental.

***Abstract*— The sensors of the Samsung A35 phone were used through the Phyphox application to carry out two experiments. First, the inclination sensor was tested, its graphs were observed, and the data were exported to .csv format for analysis in Python and MATLAB, allowing for result comparison. Then, the Optical Stopwatch experiment was conducted, which measures time intervals based on variations in illumination detected by the ambient light sensor.**

I. Introducción

En este informe se documentó el uso de la aplicación **Phyphox** en un teléfono Samsung A35 para realizar experimentos utilizando sus sensores integrados. Inicialmente, se exploraron las funcionalidades de la app mediante la prueba del sensor de inclinación, observando las gráficas generadas en tiempo real y exportando los datos en formato .csv para su posterior análisis en Python y MATLAB. Esto permitió comparar las representaciones gráficas entre las plataformas.

En la segunda parte del laboratorio, se realizó el experimento **Brightness** de la aplicación **Phyphox** para analizar cómo varía la luminancia en función del tiempo usando la cámara del teléfono móvil. Se ajustaron parámetros de exposición para evitar interferencias del procesamiento automático del dispositivo, y se utilizaron pausas de cinco segundos entre cambios de luz (oscuridad-luz) para garantizar una detección clara de cada evento. Los datos obtenidos se exportaron como archivo CSV y se graficaron tanto en **Python** como en **MATLAB**, asegurando la correcta lectura mediante el ajuste del delimitador a tabulación.

II. Desarrollo

A. Marco teórico

Python

Es un lenguaje de programación de **propósito general**, lo que significa que se puede utilizar para desarrollar una gran variedad de aplicaciones, desde análisis de datos hasta desarrollo web o inteligencia artificial. No está limitado a un área específica, lo que lo hace muy versátil.

Una de las razones por las que Python se ha convertido en uno de los lenguajes más populares es su **facilidad de uso**, especialmente para quienes están comenzando a programar. Su **sintaxis simple y clara** se asemeja al lenguaje natural, lo que facilita la lectura y escritura del código. Por esta razón, Python se considera un lenguaje de **alto nivel**.

Python es un lenguaje **interpretado**, es decir, no necesita pasar por un proceso de compilación como otros lenguajes. En su lugar, el código se ejecuta directamente mediante una **máquina virtual**, que interpreta las instrucciones y las convierte en código binario. Esto acelera el desarrollo, ya que permite probar y modificar el código con rapidez. [1]

Phyphox

Es una aplicación que aprovecha los sensores integrados en los teléfonos inteligentes para realizar experimentos científicos de manera accesible. Permite, por ejemplo, detectar la frecuencia de un péndulo utilizando el acelerómetro o medir el efecto Doppler con el micrófono del dispositivo. Una de sus grandes ventajas es la posibilidad de exportar los datos obtenidos en formatos comunes, lo que facilita su análisis en otros programas, además de permitir guardarlos o compartirlos directamente desde el teléfono. También ofrece la opción de control remoto, lo que significa que se puede manejar un experimento desde cualquier navegador web, permitiendo iniciar mediciones desde un computador y descargar los datos directamente al escritorio. Por último, Phyphox brinda la posibilidad de crear experimentos personalizados; si un experimento no está disponible en la aplicación, el usuario puede diseñar el suyo propio con ayuda de una wiki y un editor web, adaptando así la herramienta a sus necesidades específicas. [2]

Matlab

Está diseñado para adaptarse a la forma de pensar y trabajar de los usuarios, combinando un entorno de escritorio optimizado para el análisis iterativo y el diseño de procesos, con un lenguaje de programación que permite expresar operaciones matemáticas mediante matrices y arreglos de forma directa y sencilla. Sus **toolboxes** son desarrolladas profesionalmente, pasan por rigurosas pruebas de calidad y cuentan con documentación completa, lo que garantiza su confiabilidad y facilidad de uso. Además, MATLAB ofrece **apps interactivas** que permiten observar cómo distintos algoritmos actúan sobre los datos del usuario, facilitando la experimentación y el ajuste de parámetros hasta obtener los resultados deseados; posteriormente, se puede generar automáticamente un programa en MATLAB para automatizar el proceso. Otra característica destacada es su capacidad de escalamiento, ya que permite ejecutar análisis en clústeres, GPUs o en la nube con solo realizar cambios mínimos en el código, sin necesidad de reescribirlo ni de aprender programación especializada para Big Data o manejo de datos fuera de memoria. [3]

B. Procedimiento

Para este laboratorio, se utilizó un teléfono Samsung Galaxy A35, identificado con la referencia SM-A356E. Este dispositivo se empleó como herramienta principal para la recolección de datos experimentales mediante sus sensores integrados, los sensores integrados se pueden observar en la siguiente tabla

Sensor	Disponible	Nombre
Acelerómetro	Sí	Acelerómetro ICM42632M
Aceleración (sin g)	Sí	Sensor de aceleración lineal de Samsung
Giroscopio	Sí	Giroscopio ICM42632M
Magnetómetro	Sí	Magnetómetro AK09918C
Luz	Sí	Luz STK31610
Proximidad	Sí	Sensor de proximidad (ProToS)

Tabla 1. Sensores disponibles en A35

Las características principales de los sensores integrados en el dispositivo se presentan en la siguiente tabla.

Sensor	Rango	Resolución	Tasa (Hz)	Desviación estándar
Acelerómetro	78 m/s²	0,0024 m/s²	494,5	0,0061 m/s²
Aceleración (sin g)	78 m/s²	0,0024 m/s²	122,1	0,0071 m/s²
Giroscopio	17,5 rad/s	0,00053 rad/s	494,5	0,00089 rad/s
Magnetómetro	4900 µT	0,060 µT	123,3	0,44 µT
Luz	60000 lx	1,0 lx	—	—
Proximidad	—	—	—	—

Tabla 2. Características de los sensores disponibles en A35

Para la primera parte del laboratorio, se eligió trabajar con la característica de Aceleración (sin g), utilizando el sensor de aceleración lineal de Samsung, proporcionado por Samsung Electronics. Este sensor cuenta con las características que se presentan en la tabla 1 y 2, lo que lo hace adecuado para registrar variaciones precisas en la aceleración del dispositivo sin incluir el efecto de la gravedad.

Es importante tener en cuenta la diferencia entre "Aceleración con g" y "Aceleración sin g". Cuando el teléfono está en reposo, el sensor registra una aceleración de aproximadamente 9.81 m/s², correspondiente a la gravedad terrestre. A este valor se le denomina "aceleración con g". En cambio, la aceleración física real es cero cuando el teléfono está en reposo o se mueve a velocidad constante. Por ello, muchos teléfonos modernos utilizan un sensor virtual, que combina datos de múltiples sensores y resta el efecto constante de la gravedad, obteniendo así lo que se conoce como "aceleración sin g". Este sensor virtual reportará una aceleración cercana a cero mientras el dispositivo no esté experimentando un cambio real de velocidad. [4]

Se comenzó utilizando la aplicación Phyphox con el objetivo de medir la aceleración real. Para ello, se accedió a la opción que proporciona los datos brutos del acelerómetro, los cuales normalmente provienen de un sensor virtual, encargado de compensar la aceleración gravitacional a partir de la combinación de varios sensores internos del teléfono. Esta funcionalidad permite obtener datos más precisos sobre el movimiento real del dispositivo en el espacio.

Posteriormente, se realizó la toma de datos correspondiente en la aplicación para observar las gráficas obtenidas. Se registraron tres gráficas, cada una representando la aceleración lineal en los ejes X, Y y Z, respectivamente. Estas gráficas permiten analizar cómo varía la aceleración del dispositivo en cada dirección del espacio tridimensional durante el experimento.

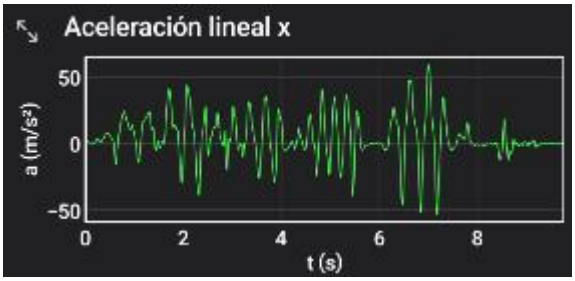

Figura 1. Grafico de aceleración lineal en x Phyphox

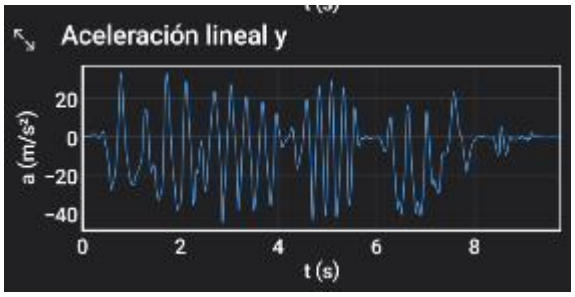

Figura 2. Gráfico de aceleración lineal en y Phyphox

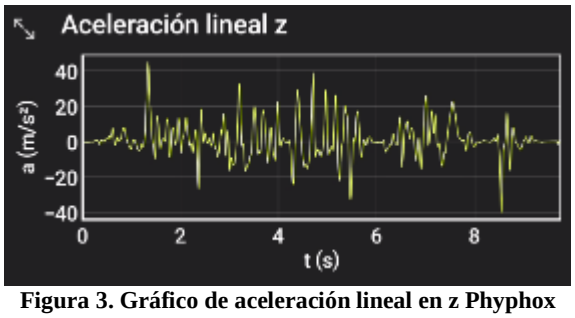

Figura 3. Gráfico de aceleración lineal en z Phyphox

Seguidamente, tras observar los gráficos obtenidos en la aplicación Phyphox, se realizó la exportación del archivo en formato CSV, el cual contiene los datos separados por tabulaciones. Posteriormente, este archivo fue editado utilizando el Bloc de notas, con el fin de renombrar las columnas a: t, x, y, z y magnitud. Este ajuste fue necesario para poder graficar correctamente los datos en Python.

```
import csv
import matplotlib.pyplot as plt
```

```
t_values = []
x_values = []
y_values = []
z_values = []
a_values = []
with open('data.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file,
delimiter='\t')
    for row in csv_reader:
        t_values.append(float(row['t']))
        x_values.append(float(row['x']))
        y_values.append(float(row['y']))
        z_values.append(float(row['z']))
        a_values.append(float(row['magnitud']))

# Graficar
plt.plot(t_values, x_values, label='x',
color='green')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración lineal en x')
plt.title('Aceleración lineal en x vs. Tiempo')
plt.grid(True)
plt.tight_layout()
plt.show()
# Graficar
plt.plot(t_values, y_values, label='y', color='blue')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración lineal en y')
plt.title('Aceleración lineal en y vs. Tiempo')
plt.grid(True)
plt.tight_layout()
plt.show()

# Graficar
plt.plot(t_values, z_values, label='z',
color='yellow')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración lineal en z')
plt.title('Aceleración lineal en z vs. Tiempo')
plt.grid(True)
plt.tight_layout()
plt.show()
# Graficar
plt.plot(t_values, a_values, label='a',
color='black')
plt.xlabel('Tiempo (s)')
plt.ylabel('Aceleración absoluta')
plt.title('Aceleración absoluta vs. Tiempo')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figura 4. Código de Python para graficar cada aceleración

Este código en Python sirve para leer y graficar datos de aceleración registrados por la aplicación Phyphox. Primero, importa las bibliotecas necesarias: csv para leer archivos CSV y matplotlib.pyplot para crear gráficos. Luego, inicializa listas vacías para almacenar los valores de tiempo (t_values), aceleración en los ejes X, Y y Z (x_values, y_values, z_values) y la aceleración total o magnitud (a_values). Después, abre el archivo data.csv y lo lee utilizando DictReader, que interpreta cada fila como un diccionario, donde las claves son los nombres de las columnas (t, x, y, z, magnitud) y los valores se convierten en números flotantes y se almacenan en sus respectivas listas. Finalmente, el código genera cuatro gráficos, uno para cada eje y otro para la aceleración total, mostrando cómo varían estas aceleraciones a lo largo del tiempo. Se muestra de forma individual con plt.show(). Las siguientes figuras muestran las gráficas obtenidas.

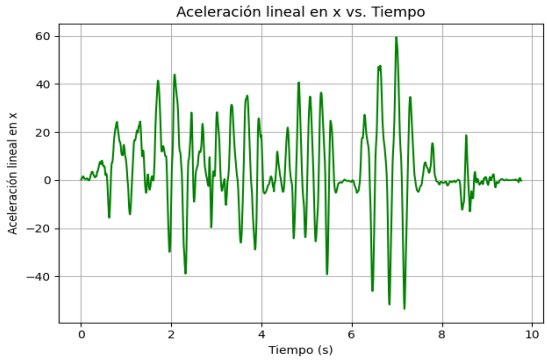

Figura 5. Gráfico de aceleración lineal en x Phyton

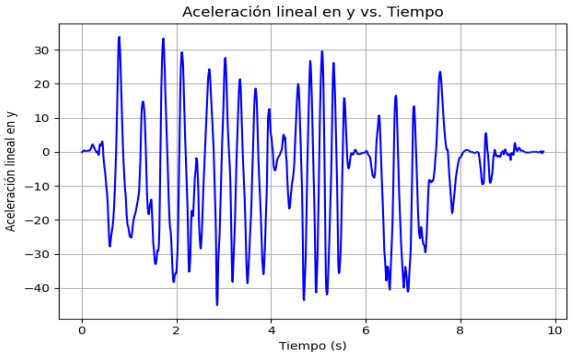

Figura 6. Gráfico de aceleración lineal en y Phyton

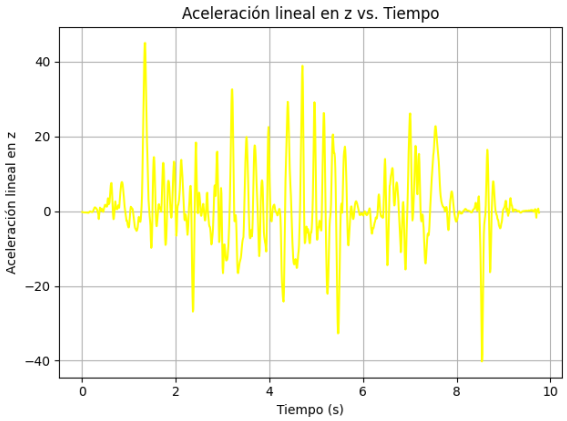

Figura 7. Gráfico de aceleración lineal en z Phyton

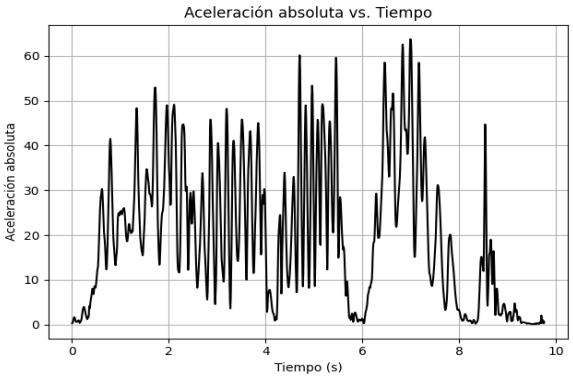

Figura 8. Gráfico de aceleración absoluta Phyton

Después de obtener las gráficas correspondientes utilizando Python, se procedió a realizar el mismo análisis en **MATLAB**. Para ello, se importó el archivo data.csv al **Workspace**, y fue necesario ajustar su

configuración de lectura. Por defecto, MATLAB interpreta los datos separados por comas, por lo que se modificó el delimitador a **tabulación** para que pudiera leer correctamente el archivo generado por Phyphox.

Una vez configurado el archivo, se utilizó el siguiente código para leer los datos y generar las gráficas correspondientes:

```
T = readtable('data.csv', 'Delimiter', '\t');  % Lee el archivo con separador tabulación
% Asignación correcta de columnas
col1 = T.t;     % tiempo
col2 = T.x;     % x
col3 = T.y;     % y
col4 = T.z;     % z
col4 = T.magnitud;     % mag
% Gráfica 1: x vs t
figure;
plot(col1, col2);
xlabel('Tiempo (s)');
ylabel('Aceleración lineal en x');
title('Gráfica de aceleración lineal en x vs tiempo');
grid on;
% Gráfica 2: y vs t
figure;
plot(col1, col3);
xlabel('Tiempo (s)');
ylabel('Aceleración lineal en y');
title('Gráfica de aceleración lineal en y vs tiempo');
grid on;
% Gráfica 3: z vs t
figure;
plot(col1, col4);
xlabel('Tiempo (s)');
ylabel('Aceleración lineal en z');
title('Gráfica de aceleración lineal en z vs tiempo');
grid on;

% Gráfica 4: aceleración absoluta vs t
figure;
plot(col1, col4);
xlabel('Tiempo (s)');
ylabel('Aceleración absoluta');
title('Gráfica de aceleración absoluta vs tiempo');
grid on;
```

Figura 9. Código de MATLAB para graficar cada aceleración

El código empleado para graficar los datos en MATLAB comienza con la instrucción `readtable`, que carga el archivo como una tabla estructurada. Luego, se asignaron las columnas correspondientes: `t` (tiempo), `x`, `y`, `z` y `magnitud`. A partir de estas variables, se generaron cuatro gráficas: la primera muestra la aceleración lineal en el eje X con respecto al tiempo, la segunda representa la aceleración en Y, la tercera en Z, y finalmente, se graficó la aceleración absoluta o magnitud en función del tiempo. Todas las gráficas incluyen etiquetas en los ejes, título descriptivo y una cuadrícula para facilitar la interpretación visual de los datos.

Luego de verificar que el código en **MATLAB** funciona correctamente y se ejecuta sin ningún error, se procede a correrlo. Como resultado, se obtienen las **cuatro gráficas correspondientes**: aceleración lineal en los ejes X, Y y Z en función del tiempo, aceleración absoluta.

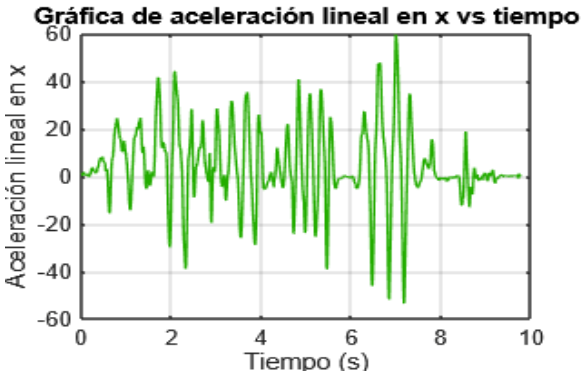

Figura 10. Gráfico de aceleración lineal en x MATLAB

Figura 11. Gráfico de aceleración lineal en y MATLAB

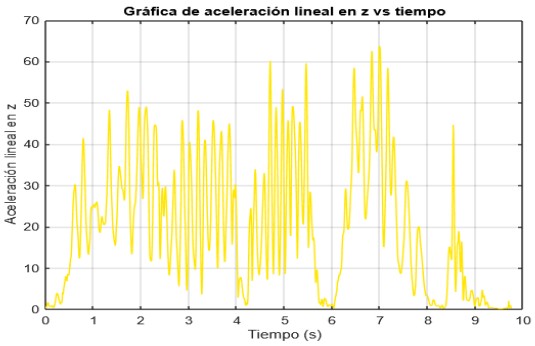

Figura 12. Gráfico de aceleración lineal en z MATLAB

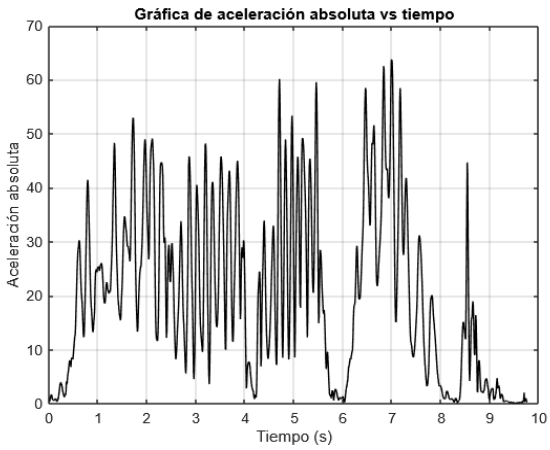

Figura 13. Gráfico de aceleración absoluta MATLAB

Con las gráficas obtenidas tanto en Python como en MATLAB, fue posible realizar una comparación entre ambas. Se observó que las

gráficas generadas en los dos softwares son prácticamente idénticas, lo cual indica que los datos fueron importados y graficados correctamente en ambos entornos. Además, estas representaciones se asemejan a las gráficas visualizadas originalmente en la aplicación Phyphox, lo que refuerza la consistencia y confiabilidad de los datos obtenidos durante el experimento. Estas gráficas muestran cómo varió la aceleración lineal del dispositivo en los ejes X, Y y Z, así como la aceleración total o magnitud, a lo largo del tiempo.

Para la segunda parte del laboratorio se quiso realizar el experimento Brightness de la aplicación Phyphox, el cual mide la luminancia relativa a partir del brillo promedio de la imagen capturada por la cámara del teléfono móvil [5]. Para llevarlo a cabo de manera más precisa, se ajustaron los parámetros predeterminados de la cámara del dispositivo, buscando una exposición intermedia, como recomienda la aplicación, con el fin de minimizar el impacto del procesamiento automático del teléfono (como el aumento de brillo en zonas oscuras o el ajuste dinámico de contraste). Durante el experimento, se observó que cuando había menos contraste o sombra, la aplicación marcaba el evento con color azul.

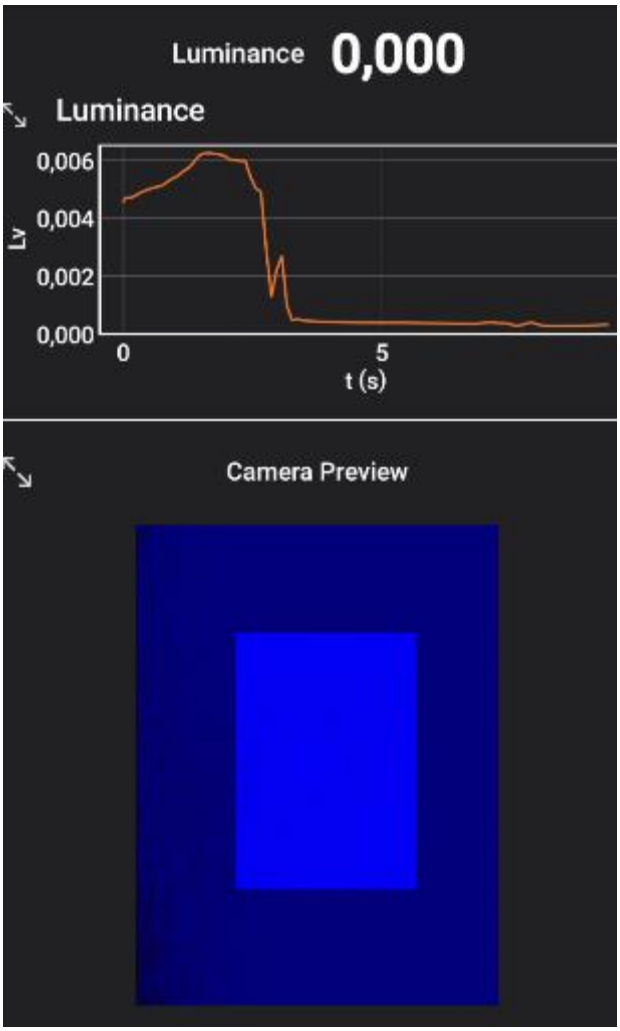

Figura 14. Ejemplo en Phyphox cuando no hay iluminación

Mientras que cuando había iluminación directa o un aumento del brillo, el evento se registraba en color rojo como se muestra en la siguiente figura.

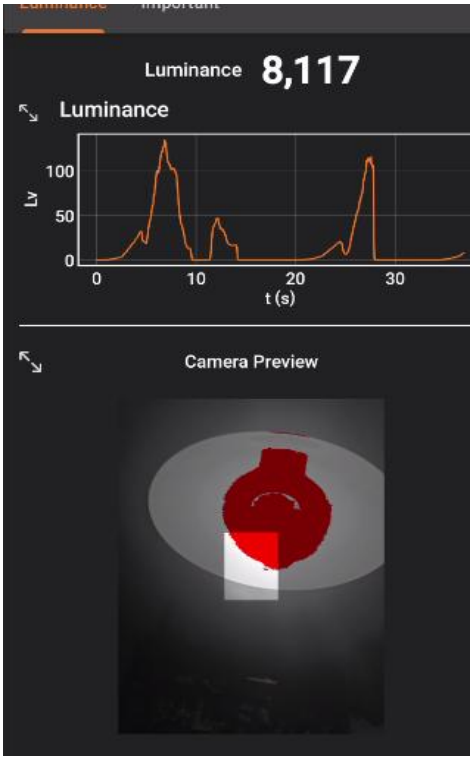

Figura 15. Ejemplo en Phyphox cuando hay iluminación

Esto permitió identificar los momentos de cambio de iluminación de forma visual, facilitando la medición precisa de intervalos temporales entre dichos eventos luminosos.

Para poder medir correctamente los intervalos de tiempo en el experimento Brightness, se realizaron pausas de aproximadamente 5 segundos entre cada cambio de luz. Esta pausa permitió que el cronómetro de Phyphox registrara de forma clara los momentos en que la cámara estaba completamente tapada, simulando un mínimo de iluminación, y cuando se exponía a la luz directa de un bombillo, representando un máximo de luminosidad. Estos cambios controlados y espaciados aseguraron que cada evento de iluminación fuera detectado con claridad por la aplicación, facilitando así la medición precisa de los intervalos temporales entre sombras y luz.

Con estas condiciones establecidas, se procedió a realizar la medición de los datos utilizando el experimento y se generó la gráfica correspondiente, la cual mostró los intervalos de tiempo entre cada transición de luz y oscuridad, permitiendo analizar visualmente los tiempos de respuesta del sistema ante variaciones en la intensidad luminosa.

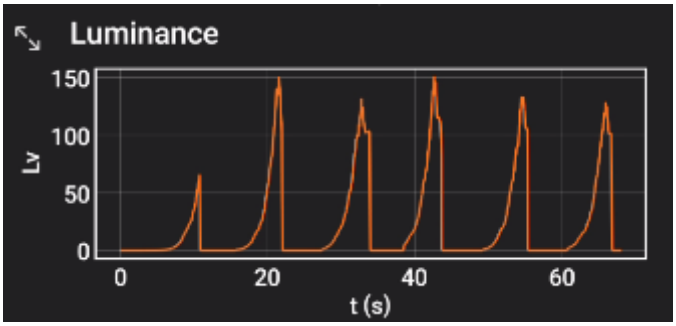

Figura 16. Valores medidos en intervalos de aproximadamente 5 segundos

Ya teniendo la gráfica del experimento Brightness se procedió a exportar los datos en un archivo CSV, siguiendo el mismo procedimiento utilizado en la primera parte del laboratorio. Este archivo se guardó con los valores separados por tabulaciones, para poder luego graficarlo correctamente en Python.

Se configuró adecuadamente el archivo CSV para evitar errores de lectura, asegurando que los datos estuvieran separados por tabulaciones y que los encabezados coincidieran con los campos requeridos. Luego, se creó el código correspondiente en Python para importar el archivo, procesar los datos y graficar los cambios de luminancia registrados durante el experimento.

```python
import csv
import matplotlib.pyplot as plt

t_values = []
l_values = []

with open('Data.csv', mode='r') as csv_file:
    csv_reader = csv.DictReader(csv_file,
delimiter='\t')
    for row in csv_reader:
        t_values.append(float(row['t']))

l_values.append(float(row['Luminance']))



# Graficar
plt.plot(t_values, l_values, label='x',
color='orange')
plt.xlabel('Tiempo (s)')
plt.ylabel('luminancia')
plt.title('luminancia en x vs. Tiempo')
plt.grid(True)
plt.tight_layout()
plt.show()
```

Figura 17. Código de Python para luminancia

Este código en Python fue diseñado para leer datos de un archivo CSV exportado desde la aplicación Phyphox, específicamente del experimento relacionado con luminancia medida por la cámara del teléfono. En primer lugar, se importaron las librerías necesarias: `csv`, que permite trabajar con archivos delimitados, y `matplotlib.pyplot`, utilizada para la generación de gráficos.

Durante la lectura del archivo, el código recorrió cada fila y fue extrayendo los valores de las columnas `t` (tiempo) y `Luminance` (luminancia). Estos valores fueron convertidos en números de punto flotante (`float`) para asegurar que pudieran ser graficados correctamente. Finalmente, se utilizó `matplotlib.pyplot` para trazar la curva de luminancia en función del tiempo, añadiendo etiquetas a los ejes, título y una cuadrícula para facilitar la lectura de la gráfica. Esta visualización permite analizar cómo varió el brillo a lo largo del tiempo durante el experimento.

Su comportamiento se puede observar en la siguiente figura.

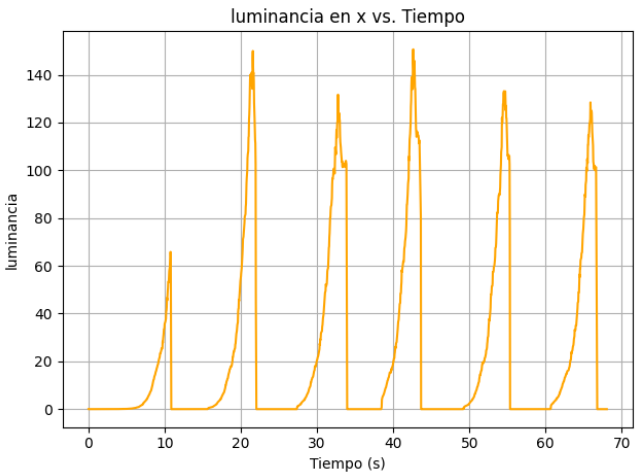

Figura 18. Grafica de luminancia obtenida en Python

Después de obtener la gráfica correspondiente utilizando Python, se procedió a realizar el mismo análisis en **MATLAB**. Para ello, se importó el archivo Data.csv al **Workspace**, y fue necesario ajustar su configuración de lectura. Por defecto, MATLAB interpreta los datos separados por comas, por lo que se modificó el delimitador a **tabulación** para que pudiera leer correctamente el archivo generado por Phyphox.

Una vez configurado el archivo, se utilizó el siguiente código para leer los datos y generar la gráfica correspondiente:

```matlab
T = readtable('Data.csv', 'Delimiter', '\t');
% Lee el archivo con separador tabulación
% Asignación correcta de columnas
col1 = T.t;     % tiempo
col2 = T.Luminance;     % x

% Gráfica 1: luminance vs t
figure;
plot(col1, col2);
xlabel('Tiempo (s)');
ylabel('Luminancia ');
title('Luminancia  vs tiempo');
grid on;
```

Figura 19. Código de MATLAB para luminancia

Este código en MATLAB tiene como objetivo graficar la luminancia en función del tiempo a partir de un archivo CSV previamente exportado desde la aplicación Phyphox. En la primera línea, se utiliza la función `readtable` para leer el archivo llamado `'Data.csv'`, especificando que el delimitador entre columnas es una tabulación (`'\t'`), lo cual es importante para interpretar correctamente los datos.

Luego, se asignan las columnas del archivo a variables individuales: `col1` almacena los valores de tiempo (`T.t`) y `col2` contiene los valores de luminancia (`T.Luminance`). Esta asignación permite trabajar más fácilmente con los datos al momento de graficar.

Finalmente, se genera una gráfica con `plot`, donde se representa la luminancia (`col2`) en el eje vertical y el tiempo (`col1`) en el eje horizontal. Se añaden etiquetas a los ejes y un título descriptivo para interpretar adecuadamente el gráfico. Además, se activa la cuadrícula

(`grid on`) para mejorar la visualización y facilitar la lectura de los datos. La siguiente gráfica permite analizar cómo varió la intensidad de luz captada por la cámara del teléfono durante el experimento.

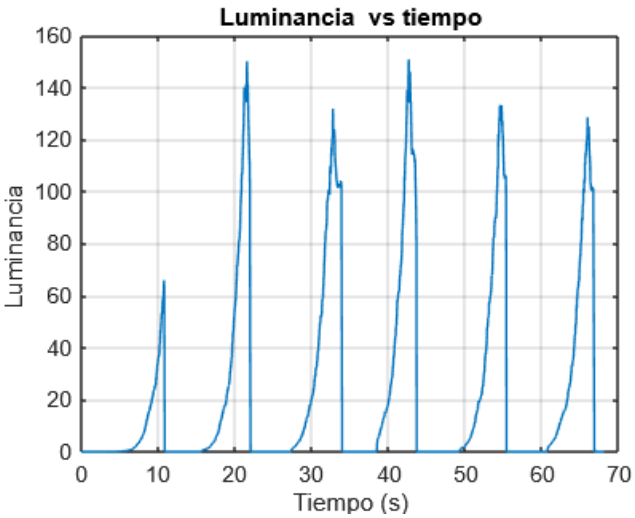

Figura 20. Grafica de luminancia MATLAB

Ya teniendo en cuenta las tres gráficas obtenidas —una desde la aplicación Phyphox, otra generada en Python y una más en MATLAB— se pudo realizar una comparación visual directa entre ellas. Al observar sus formas, se evidenció que todas presentan la misma distribución de picos de luminancia con intervalos regulares, correspondientes a los momentos en los que se encendió la fuente de luz durante el experimento. Esta coincidencia entre las tres gráficas indica que **los datos fueron correctamente capturados, exportados y procesados**, sin pérdida ni alteración de la información. Además, confirma que las herramientas utilizadas (Phyphox, Python y MATLAB) interpretaron de forma consistente los mismos valores, lo que valida tanto la metodología experimental como la confiabilidad del análisis de datos.

En la gráfica se pueden identificar **seis picos bien definidos**, los cuales corresponden a momentos de iluminación intensa, intercalados con valles donde la cámara estuvo cubierta o en condiciones de baja luz. Este patrón se repite de forma regular, lo que indica que el experimento fue realizado con pausas similares entre cada cambio de luz.

III. REFERENCIAS

[1] S. Staacks, «Your smartphone is a mobile lab.», *Phyphox*. https://phyphox-org.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[2]«¿Qué es Python? | Oracle Colombia». https://www.oracle.com/co/developer/what-is-python-for-developers/

[3]«MATLAB - El lenguaje del cálculo técnico». https://la.mathworks.com/products/matlab.html

[4] S. Staacks, «Acceleration (without g)», *Phyphox*. https://phyphox-org.translate.goog/experiment/acceleration-without-g/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc

[5]«Raw data», *Phyphox*. https://phyphox-org.translate.goog/topic/raw/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc