In [48]:
```
from statistical_utilities import math_utilities as utils
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.signal import find_peaks
from scipy.ndimage import gaussian_filter1d
import statsmodels.api as sm
```

ALMACENAMIENTO DE DATOS: Todos los datos tomados en esta práctica están almacenados en OneDrive: https://uniandes-my.sharepoint.com/:f:/g/personal/k_murcia_uniandes_edu_co/EpqZUJG_V_pPrmdpxrRXXbgBaegSgA48bKjsUeEoe=L8zMbV

INCERTIDUMBRES:

- $\sigma_{U_1} = 0.01\ \mathrm{V}$.
- $\sigma_{I_A} = 0.01\ \mathrm{nA}$.
- $\sigma_T = 1\ ^\circ\mathrm{C}$.

SESIÓN 01

FECHA: 2024-08-15

HORA: 16:00

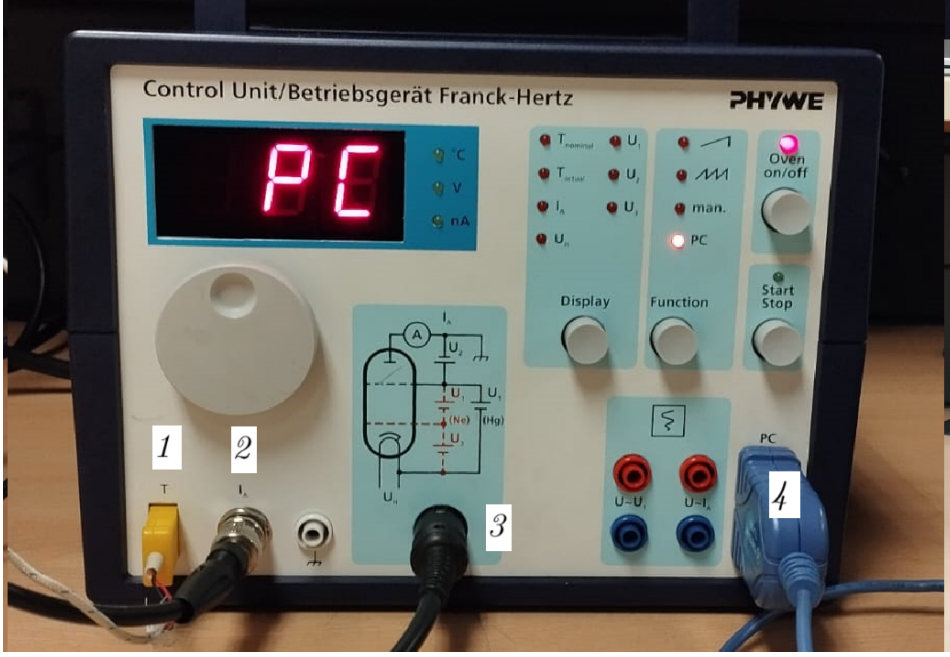
EXPERIMENTALISTAS: Juan Carlos Rojas Velásquez (jc.rojasv1@uniandes.edu.co) & Katherin A. Murcia S. (k.murcia@uniandes.edu.co)

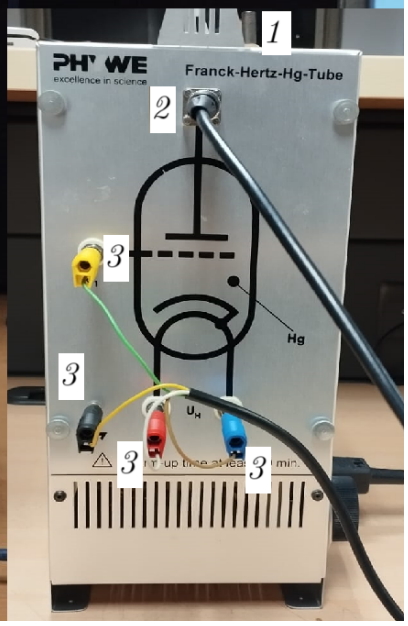
LABORATORIO: B-301.

OBJETIVOS DE LA SESIÓN:

- Tomar series de datos con T, U_2 y U_H constantes (al menos 3).
- Tomar series de datos a diferentes temperaturas con U_2 y U_H constantes (al menos 3).
- Tomar series de datos con variaciones de U_2 y con T y U_H fijos (al menos 3).
- Tomar series de datos con variaciones de U_H a T y U_2 constantes(al menos 3).

CONEXIONES: Se conectaron los dispositivos siguiendo los esquemas de los manuales operativos. Los cuales resultaron en las siguientes conexiones de la estufa (derecha) y el módulo de control (izquierda):





Las conexiones van así (aquí se ilustran las conexiones por la correspondencia de índices): **1.** Sensor de temperatura se introduce en la estufa. **2.** Cable de medición de corriente I_A. **3.** Conexión de alimentación de voltajes que se divide en cuatro cables: U_H (cables rojo y azul), U_1 (cable amarillo) y tierra (cable negro) desde el controlador a la estufa y **4.** Conexión del módulo de control con el computador.

CONDICIONES ADICIONALES: Hay una ventana grande abierta junto a algunos metros del horno. Esta provoca brisa, creemos que podría alterar de manera sutil los datos tomados.

ACTIVIDAD 02. EJERCICIO 01.

PARÁMETROS: Se ajustaron los siguentes parámetros en el software para hacer las mediciones: $U_1 = 60.00\text{V}$, $U_2 = 1.5\text{V}$, $U_H = 6.3\text{V}$, variando las temperaturas para cada una de las tomas de datos alrededor de $T = 205\ ^\circ\text{C}$.

- Los primeros intentos de hacer mediciones fallaron. En adelante se realizaron graduaciones de temperaturas de forma que los cambios en temperatura no fueran tan drásticos e inestables.

- Incrementamos la temperatura muy lentamente y nos retrasamos.

TOMA DE DATOS: Las series de datos se llaman: ACTIVIDAD 02. TOMA N. T=X°C.csv

In [49]:
```
dat1 = pd.read_csv("ACTIVIDAD 02. TOMA 02. T=203°C.csv",sep="   ")
dat2 = pd.read_csv("ACTIVIDAD 02. TOMA 04. T=204°C.csv",sep="   ")
dat3 = pd.read_csv("ACTIVIDAD 02. TOMA 05. T=206°C.csv",sep="   ")
```

In [50]:
```
dat1.T
```

Out[50]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5

Corriente IA	0.43	0.46	0.46	0.48	0.46	0.49	0.47	0.48	0.42	0.45	...	14.16	14.24	14.40	14.48	14.65	14.68	1

2 rows × 2457 columns

In [51]:
```
dat2.T
```

Out[51]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.40	0.36	0.44	0.41	0.47	0.38	0.48	0.39	0.51	0.41	...	7.31	7.39	7.38	7.44	7.47	7.58	

2 rows × 2457 columns

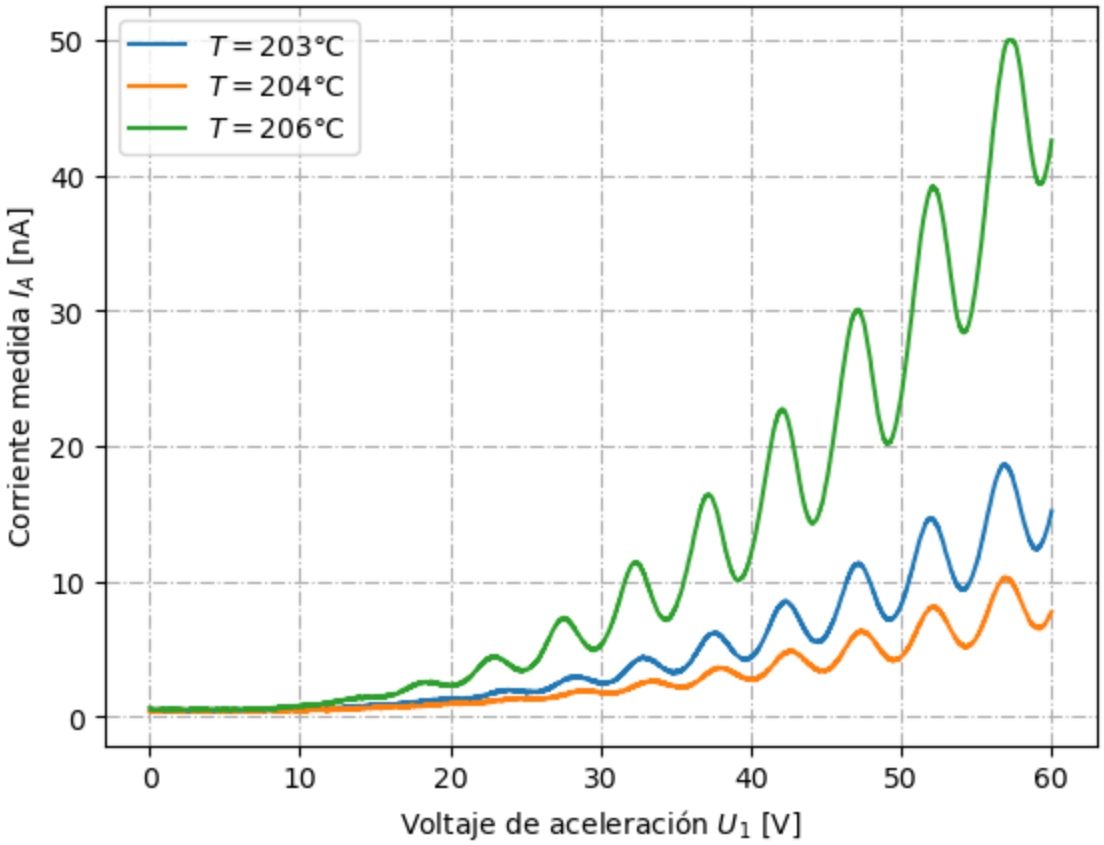
In [52]:
```
dat3.T
```

Out[52]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.67	0.65	0.58	0.60	0.54	0.56	0.50	0.55	0.51	0.57	...	41.05	41.21	41.36	41.54	41.74	41.96	4

2 rows × 2457 columns

In [53]:
```
plt.plot(dat1["Voltage U1"].values,dat1["Corriente IA"].values, label=r"$T = 203$°C")
plt.plot(dat2["Voltage U1"].values,dat2["Corriente IA"].values, label=r"$T = 204$°C")
plt.plot(dat3["Voltage U1"].values,dat3["Corriente IA"].values, label=r"$T = 206$°C")
plt.xlabel(r"Voltaje de aceleración $U_1$ [V]")
plt.ylabel(r"Corriente medida $I_A$ [nA]")
plt.grid(linestyle = "-.")
plt.legend()
plt.savefig("Gráfica Actividad 02 Punto 1.png")
```



Cada gráfica presenta un comportamiento creciente y oscila; cuyo valles y picos se amplifican a medida que se incrementa el voltaje. Si bien las posiciones de los valles y picos de cada una de las series de datos no se ven afectadas al variar la temperatura, se puede evidenciar un cambio en la magnitud de la corriente medida. Consideramos, dado que las posiciones de los valles y picos están alineadas bastante bien, que no sería necesario hacer un análisis estadísitico de los datos.

Aprendimos a tomar mediciones ágilmente

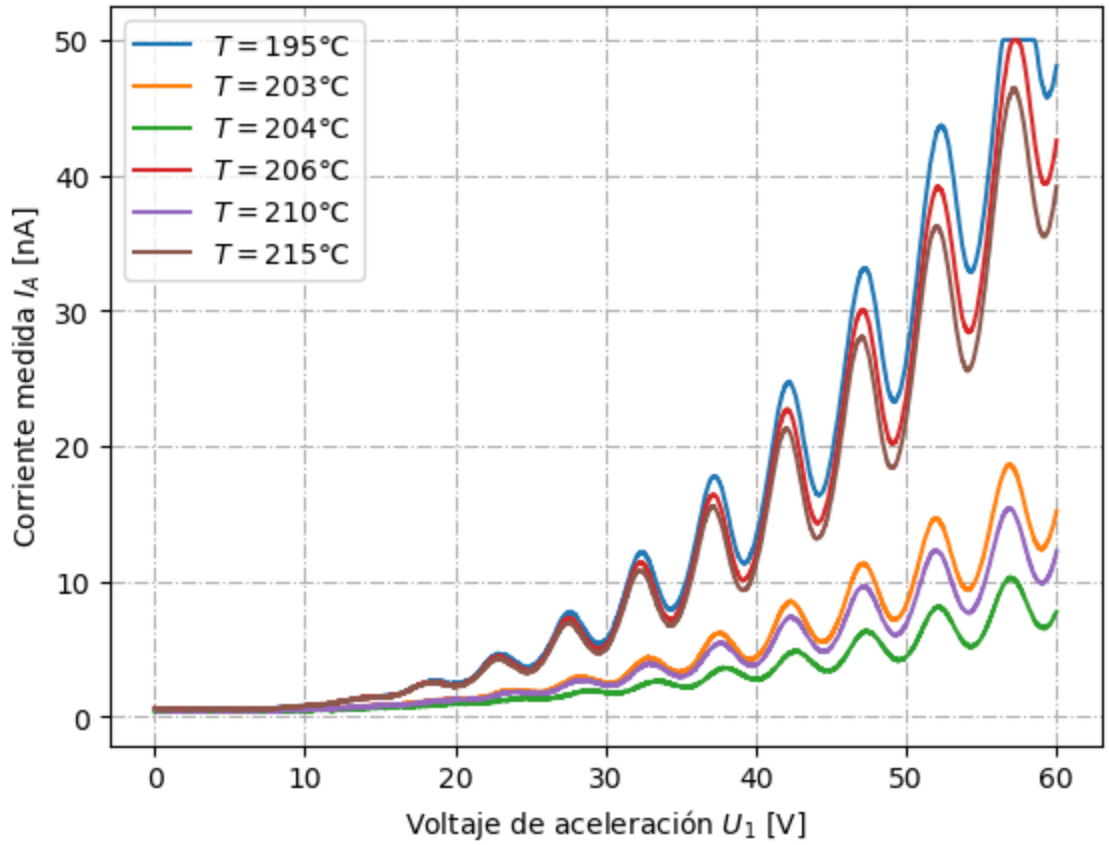
ACTIVIDAD 02. EJERCICIO 02.

PARÁMETROS: Se ajustaron los siguentes parámetros en el software para hacer las mediciones: $U_1 = 60.00\text{V}$, $U_2 = 1.5\text{V}$ y $U_H = 6.3\text{V}$ variando las temperaturas para cada una de las tomas de datos.

TOMA DE DATOS: Las series de datos se llaman: ACTIVIDAD 02. TOMA N. T=X°C.csv

In [54]:
```
dat1 = pd.read_csv("ACTIVIDAD 02. TOMA 02. T=203°C.csv",sep="   ")
dat2 = pd.read_csv("ACTIVIDAD 02. TOMA 04. T=204°C.csv",sep="   ")
dat3 = pd.read_csv("ACTIVIDAD 02. TOMA 05. T=206°C.csv",sep="   ")
dat4 = pd.read_csv("ACTIVIDAD 02. TOMA 06. T=195°C.csv",sep="   ")
#dat5 = pd.read_csv("ACTIVIDAD 02. TOMA 04. T=204°C.csv",sep="  ")
dat6 = pd.read_csv("ACTIVIDAD 02. TOMA 07. T=215°C.csv",sep="   ")
#dat7 = pd.read_csv("ACTIVIDAD 02. TOMA 01. T=210°C.csv",sep="  ")
dat8 = pd.read_csv("ACTIVIDAD 02. TOMA 03. T=210°C.csv",sep="   ")

plt.plot(dat4["Voltage U1"].values,dat4["Corriente IA"].values, label=r"$T = 195$°C")
plt.plot(dat1["Voltage U1"].values,dat1["Corriente IA"].values, label=r"$T = 203$°C")
plt.plot(dat2["Voltage U1"].values,dat2["Corriente IA"].values, label=r"$T = 204$°C")
plt.plot(dat3["Voltage U1"].values,dat3["Corriente IA"].values, label=r"$T = 206$°C")
plt.plot(dat8["Voltage U1"].values,dat8["Corriente IA"].values, label=r"$T = 210$°C")
```

```
plt.plot(dat6["Voltage U1"].values,dat6["Corriente IA"].values, label=r"$T = 215$°C")

#plt.plot(dat5["Voltage U1"].values,dat5["Corriente IA"].values, label=r"$T = 204$°C")
#plt.plot(dat7["Voltage U1"].values,dat7["Corriente IA"].values, label=r"$T = 210$°C")


plt.xlabel(r"Voltaje de aceleración $U_1$ [V]")
plt.ylabel(r"Corriente medida $I_A$ [nA]")
plt.grid(linestyle = "-.")
plt.legend()
plt.savefig("Gráficas Actvidad 2 Punto 2.png")
```



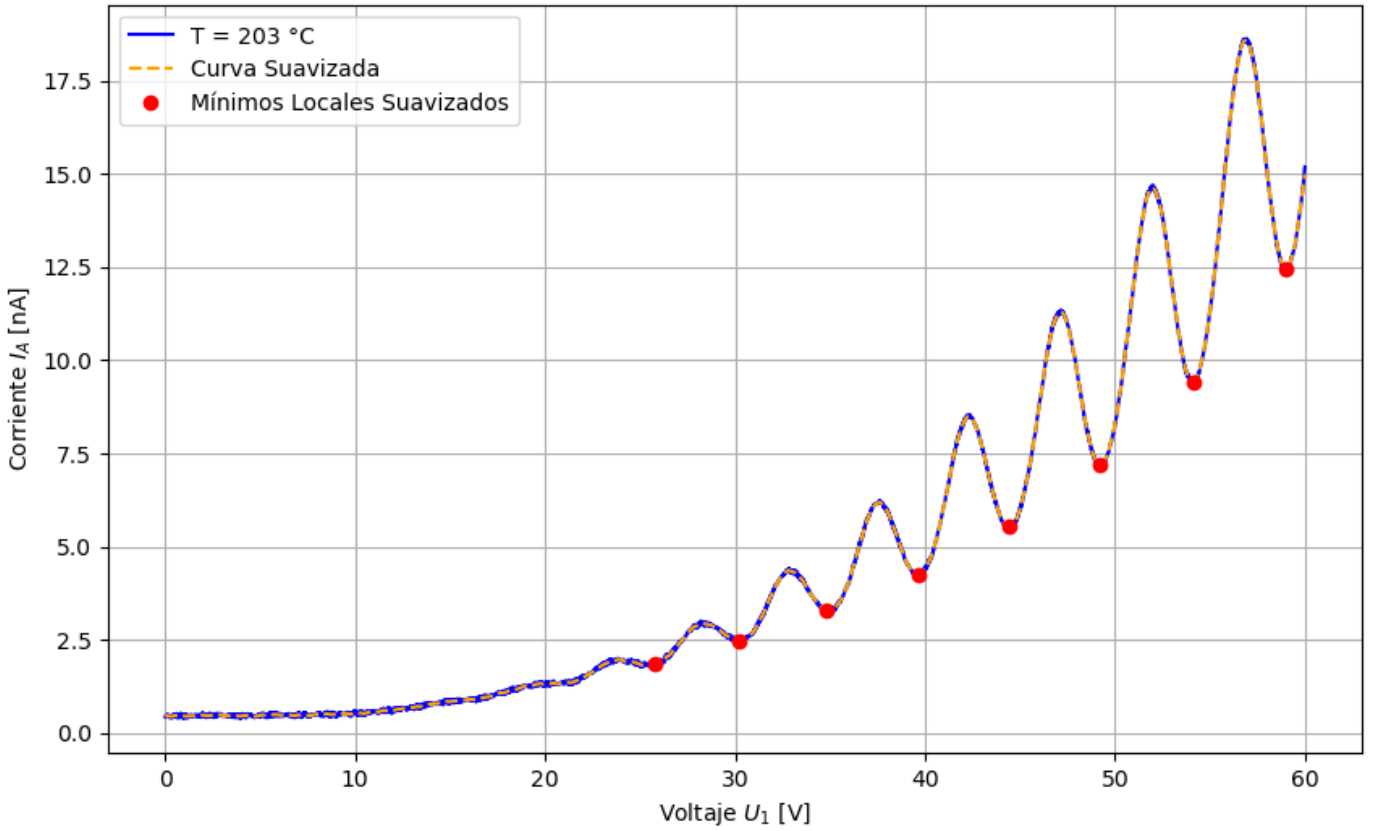
El comportamiento de los datos no es el esperado dado que se espera que a medida que la temperatura aumente las magnitudes de corriente disminuyan, sin embargo se encuentran puntos de acumulación den corriente

---

ACTIVIDAD 02. EJERCICIO 03.

In [55]:
```
file_path = 'ACTIVIDAD 02. TOMA 02. T=203°C.csv'
data = pd.read_csv(file_path, sep='     ')
N = 8
# Convertir los datos a arrays numpy
voltaje = data['Voltage U1'].values
corriente = data['Corriente IA'].values

# Suavizar la curva usando un filtro gaussiano
smoothed_corriente = gaussian_filter1d(corriente, sigma=3)

# Encontrar el mínimo de la curva suavizada
min_envolvente_index = np.argmin(smoothed_corriente)

# Encontrar los mínimos locales en la curva suavizada
```

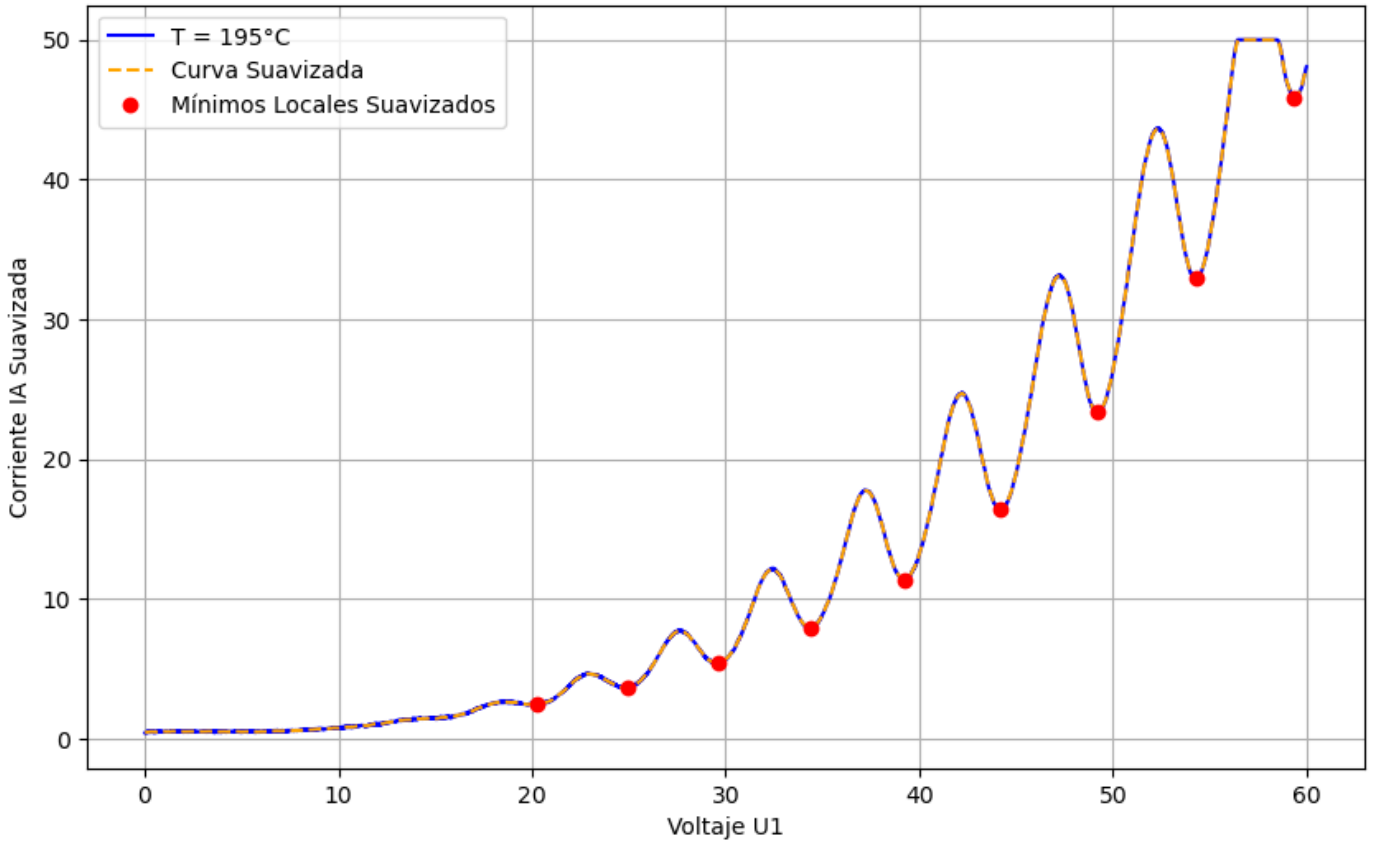
```
minima_suavizada_indices, _ = find_peaks(-smoothed_corriente)  # Invertir la corriente s

# Crear un DataFrame con los resultados de los mínimos locales
minima_suavizada_voltajes = voltaje[minima_suavizada_indices]
minima_suavizada_corrientes = smoothed_corriente[minima_suavizada_indices]

minima_suavizada_df = pd.DataFrame({
    'Voltaje U1': minima_suavizada_voltajes[:-7],
    'Corriente IA Suavizada': minima_suavizada_corrientes[:-7]
})

# Mostrar los resultados de los mínimos locales
# print(minima_suavizada_df)

# Plotear la curva original, la curva suavizada, y los mínimos locales
plt.figure(figsize=(10, 6))
plt.plot(voltaje, corriente, label=r"T = 203 °C", color='blue')
plt.plot(voltaje, smoothed_corriente, label="Curva Suavizada", color='orange', linestyle
plt.plot(voltaje[minima_suavizada_indices][-1*N:], smoothed_corriente[minima_suavizada_i
plt.xlabel('Voltaje $U_1$ [V]')
plt.ylabel('Corriente $I_A$ [nA]')
plt.legend()
plt.grid(True)
plt.show()
print(minima_suavizada_voltajes[-8:])
min_import = minima_suavizada_voltajes[-8:]

delta1 = np.zeros(len(min_import)-1)

for j in range(len(delta1)):
    delta1[j] = min_import[j+1] - min_import[j]

print(np.mean(delta1))
```




```
[25.78 30.2  34.84 39.65 44.41 49.23 54.09 58.99]
4.744285714285715
```

In [56]:
```
file_path = 'ACTIVIDAD 02. TOMA 06. T=195°C.csv'
data = pd.read_csv(file_path, sep='     ')
```

```
N = 9
# Convertir los datos a arrays numpy
voltaje = data['Voltage U1'].values
corriente = data['Corriente IA'].values

# Suavizar la curva usando un filtro gaussiano
smoothed_corriente = gaussian_filter1d(corriente, sigma=3)

# Encontrar el mínimo de la curva suavizada
min_envolvente_index = np.argmin(smoothed_corriente)

# Encontrar los mínimos locales en la curva suavizada
minima_suavizada_indices, _ = find_peaks(-smoothed_corriente)  # Invertir la corriente s

# Crear un DataFrame con los resultados de los mínimos locales
minima_suavizada_voltajes = voltaje[minima_suavizada_indices]
minima_suavizada_corrientes = smoothed_corriente[minima_suavizada_indices]

minima_suavizada_df = pd.DataFrame({
    'Voltaje U1': minima_suavizada_voltajes[:-7],
    'Corriente IA Suavizada': minima_suavizada_corrientes[:-7]
})

# Mostrar los resultados de los mínimos locales
# print(minima_suavizada_df)

# Plotear la curva original, la curva suavizada, y los mínimos locales
plt.figure(figsize=(10, 6))
plt.plot(voltaje, corriente, label="T = 195°C", color='blue')
plt.plot(voltaje, smoothed_corriente, label="Curva Suavizada", color='orange', linestyle
plt.plot(voltaje[minima_suavizada_indices][-1*N:], smoothed_corriente[minima_suavizada_i
plt.xlabel('Voltaje U1')
plt.ylabel('Corriente IA Suavizada')
plt.legend()
plt.grid(True)
plt.show()
print(minima_suavizada_voltajes[-N:])
min_import = minima_suavizada_voltajes[-N:]

delta2 = np.zeros(len(min_import)-1)

for j in range(len(delta2)):
    delta2[j] = min_import[j+1] - min_import[j]

print(np.mean(delta2))
```

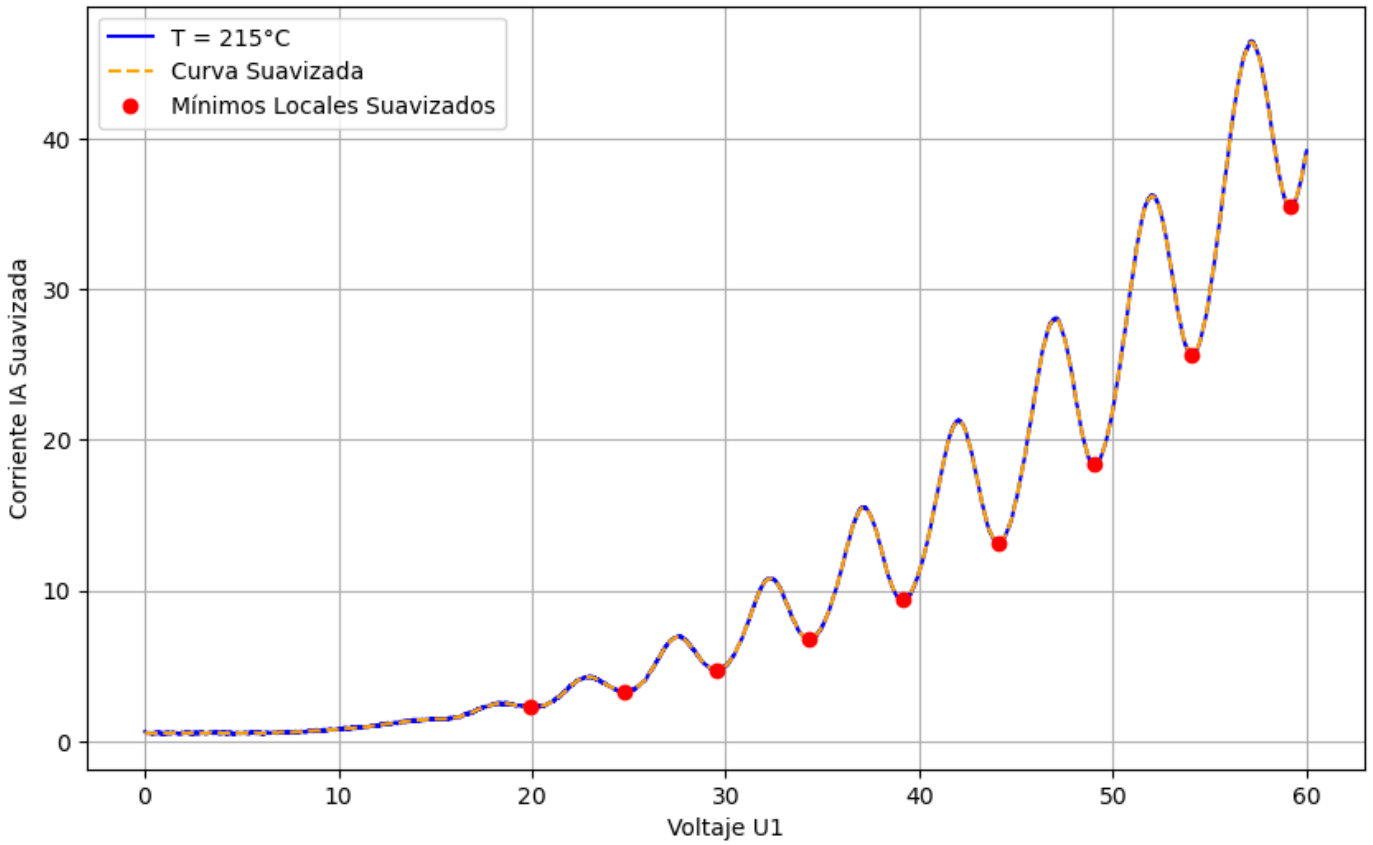



```
[20.24 24.9  29.62 34.35 39.21 44.2  49.2  54.28 59.36]
4.890000000000001
```

In [57]:

```
file_path = 'ACTIVIDAD 02. TOMA 07. T=215°C.csv'
data = pd.read_csv(file_path, sep='     ')
N = 9
# Convertir los datos a arrays numpy
voltaje = data['Voltage U1'].values
corriente = data['Corriente IA'].values

# Suavizar la curva usando un filtro gaussiano
smoothed_corriente = gaussian_filter1d(corriente, sigma=3)

# Encontrar el mínimo de la curva suavizada
min_envolvente_index = np.argmin(smoothed_corriente)

# Encontrar los mínimos locales en la curva suavizada
minima_suavizada_indices, _ = find_peaks(-smoothed_corriente)  # Invertir la corriente s

# Crear un DataFrame con los resultados de los mínimos locales
minima_suavizada_voltajes = voltaje[minima_suavizada_indices]
minima_suavizada_corrientes = smoothed_corriente[minima_suavizada_indices]

minima_suavizada_df = pd.DataFrame({
    'Voltaje U1': minima_suavizada_voltajes[:-7],
    'Corriente IA Suavizada': minima_suavizada_corrientes[:-7]
})

# Mostrar los resultados de los mínimos locales
# print(minima_suavizada_df)

# Plotear la curva original, la curva suavizada, y los mínimos locales
plt.figure(figsize=(10, 6))
plt.plot(voltaje, corriente, label="T = 215°C", color='blue')
plt.plot(voltaje, smoothed_corriente, label="Curva Suavizada", color='orange', linestyle
plt.plot(voltaje[minima_suavizada_indices][-N:], smoothed_corriente[minima_suavizada_ind
plt.xlabel('Voltaje U1')
plt.ylabel('Corriente IA Suavizada')
```

```
plt.legend()
plt.grid(True)
plt.show()
print(minima_suavizada_voltajes[-N:])
min_import = minima_suavizada_voltajes[-N:]

delta3 = np.zeros(len(min_import)-1)

for j in range(len(delta3)):
    delta3[j] = min_import[j+1] - min_import[j]

delta3
print(np.mean(delta3))
```




```
[19.95 24.73 29.52 34.28 39.14 44.05 49.05 54.09 59.16]
4.901249999999999
```

T (°C)	σ_T (°C)	<ΔE> (eV)	σ_<ΔE> (eV)
203	1	4,74	0,01
195	1	4,89	0,01
215	1	4,9	0,01

ACTIVIDAD 02. EJERCICIO 04.

Las discrepancias entre la distancia calculada entre mínimos y el valor esperado de $4.67e\mathrm{V}$ se puede explicar teniendo en cuenta la función trabajo del amperímetro, puede este se debe llevar un poco de energía.

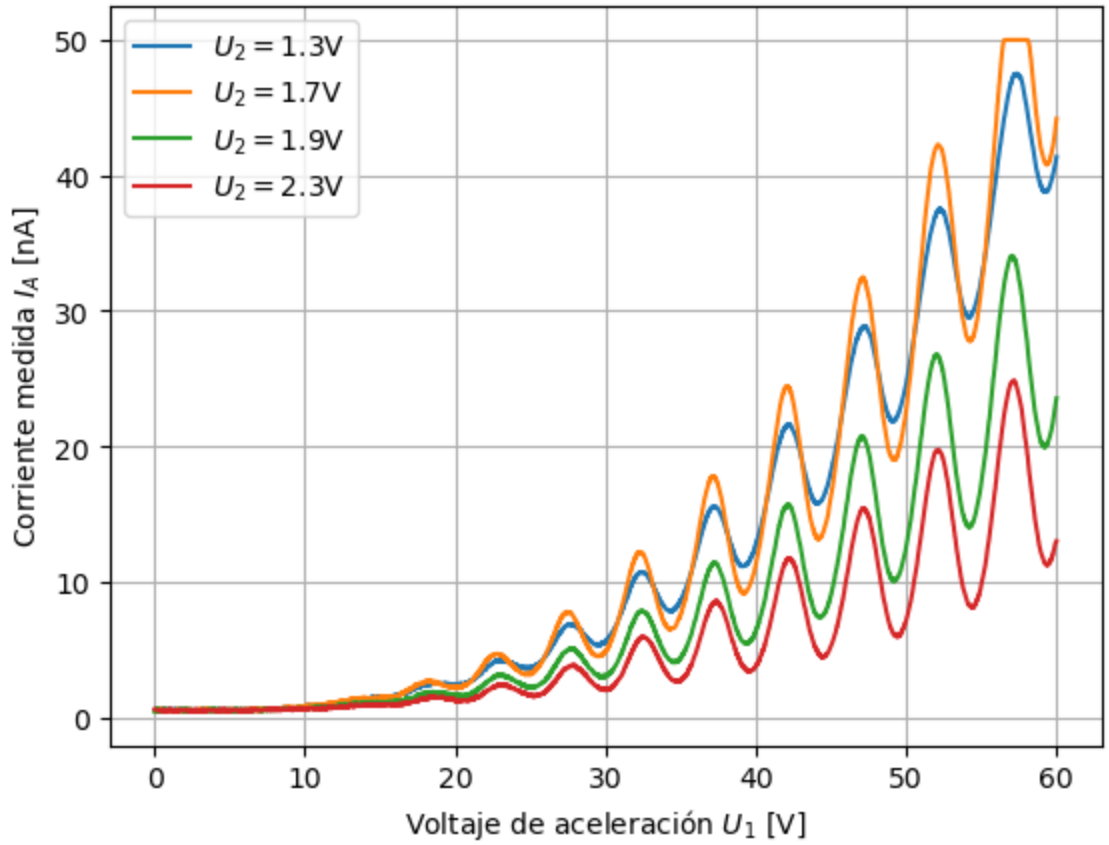
ACTIVIDAD 02. EJERCICIO 05.

con $T = 210\ ^\circ\mathrm{C}$ el sensor de corriente se saturó muchísimo para $U_2 = 1\mathrm{V} \implies$ tomamos datos para valores de U_2 entre $1.3\mathrm{V}$ y $2.3\mathrm{V}$.

PARÁMETROS: Se ajustaron los siguentes parámetros en el software para hacer las mediciones: $U_1 = 60.00\mathrm{V}$, $T = 210^\circ\mathrm{C}$ y $U_H = 6.3\mathrm{V}$ variando las U_2 para cada una de las tomas de datos en el rango de $(1.3 - 2.3)\mathrm{V}$.

TOMA DE DATOS: Las series de datos se llaman: ACTIVIDAD 02.05. TOMA N. T=210. U_2=X.csv

In [58]:
```
datos1 = pd.read_csv("ACTIVIDAD 02.05. TOMA 01. T=210. U_2=1.7.csv",sep="       ")
datos2 = pd.read_csv("ACTIVIDAD 02.05. TOMA 02. T=210. U_2=1.9.csv",sep="       ")
datos3 = pd.read_csv("ACTIVIDAD 02.05. TOMA 03. T=210. U_2=1.3.csv",sep="       ")
datos4 = pd.read_csv("ACTIVIDAD 02.05. TOMA 04. T=210. U_2=2.13.csv",sep="      ")

plt.plot(datos3["Voltage U1"].values,datos3["Corriente IA"].values, label=r"$U_2 = 1.3$V
plt.plot(datos1["Voltage U1"].values,datos1["Corriente IA"].values, label=r"$U_2 = 1.7$V
plt.plot(datos2["Voltage U1"].values,datos2["Corriente IA"].values, label=r"$U_2 = 1.9$V
plt.plot(datos4["Voltage U1"].values,datos4["Corriente IA"].values, label=r"$U_2 = 2.3$V
plt.legend()
plt.xlabel(r"Voltaje de aceleración $U_1$ [V]")
plt.ylabel(r"Corriente medida $I_A$ [nA]")
plt.grid()
```



La gráfica anterior muestran las series de datos a temperatura constante ($T = 210^\circ\mathrm{C}$) a diferentes valores de voltaje de desaceleración U_2. Los datos se comportan de forma esperada a excepción de la medida para $U_2 = 1.7\mathrm{V}$ pues se deberaba que su gráfica fuera menor a la dada por los datos $U_2 = 1.3\mathrm{V}$. Creemos que esto puede deberse a la limitación del instrumento de medición de coriente.

ACTIVIDAD 02. EJERCICIO 06.

PARÁMETROS: $U_1 = 60.00\text{V}$. $T = 210^\circ\text{C}$. $U_2 = 1.5\text{V}$. Tomamos datos en el rango para $U_H = 5.9\text{V}$ a 6.7V.

TOMA DE DATOS: Las series de datos se llaman: ACTIVIDAD 02.06. TOMA N. T=210. U_2=2.3. U_H=X.csv

La primera medida para el ejercicio 6 no se debe tener en cuenta dado que intentamos tomar datos con $T = 205^\circ\text{C}$ y $U_2 = 1.5\text{V}$ y la corriente se saturó para el voltaje $U_H = 5.9\text{V} \implies$ cambiamos los paámetros a los que están registrados bajo el subtítulo de la sección.

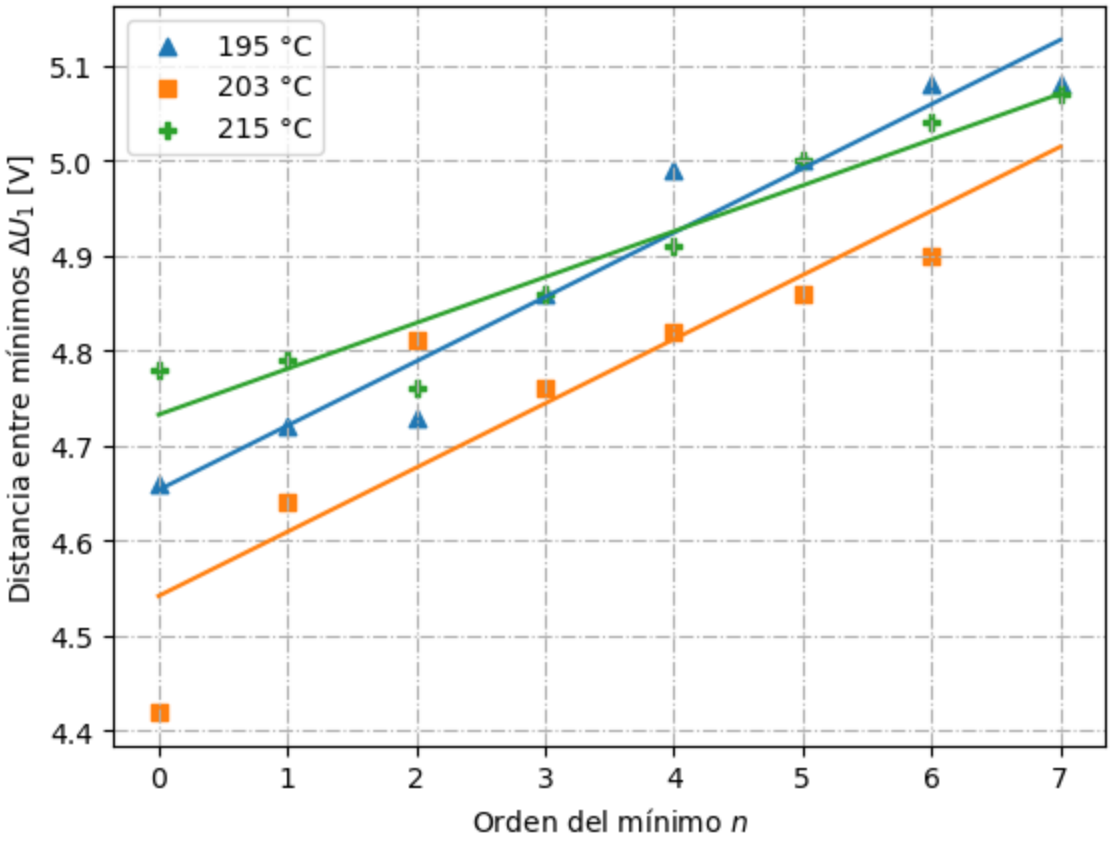
Sólo alcanzamos a hacer dos tomas, una de las cuales no sirve.

ACTIVIDAD 02. EJERCICIO 07.

In [59]:
```
deltas = [delta2,delta1,delta3]
mark = ["^","s","P"]
temps = ["195 °C","203 °C","215 °C"]
nlist = [0,0,0]
for i in range(len(deltas)):
    n = np.array(list(range(len(deltas[i]))))
    nlist[i] = n
    plt.scatter(n,deltas[i],marker=mark[i],label=temps[i])

plt.xlabel("Orden del mínimo $n$")
plt.ylabel(r"Distancia entre mínimos $\Delta U_1$ [V]" )
plt.grid(linestyle="-.")

x_space = np.linspace(0,7)
for i in range(len(deltas)):
    params = utils.linear_w_regression(nlist[i],deltas[i],1*np.ones_like(deltas[i]),0.1*
    plt.plot(x_space,params[0]*x_space + params[2])
plt.legend()
```

```
m: 0.06761904761904762 uncertanty m: 0.01543033499620919
b: 4.653333333333333 uncertanty b: 0.06454972243679027
m: 0.06750000000000048 uncertanty m: 0.01889822365046136
b: 4.541785714285712 uncertanty b: 0.06813851438692468
m: 0.04821428571428516 uncertanty m: 0.01543033499620919
b: 4.7325 uncertanty b: 0.06454972243679027
```

Out[59]:
```
<matplotlib.legend.Legend at 0x1eb3b1c0760>
```



SESIÓN 02.

FECHA: 2024-08-22

HORA: 16:00

EXPERIMENTALISTAS: Juan Carlos Rojas Velásquez (jc.rojasv1@uniandes.edu.co) & Katherin A. Murcia S. (k.murcia@uniandes.edu.co)

LABORATORIO: B-301.

OBJETIVOS DE LA SESIÓN:

- Repetir algunas tomas de datos, pues no se observa el comportamiento esperado con varios sets de datos tomados durante la sesión anterior por errores con el sensor de temperatura o con el uso del horno:
- Repetir tomas de datos con T, U_2 y U_H constantes (tener al menos 5 útiles).
- Repetir tomas de datos a temperaturas diferentes (entre 195°C y 215°C) con U_2 y U_H constantes (tener al menos 5 útiles).
- Repetir tomas de datos a T y U_H constantes con variaciones de U_2 (la vez pasada tomamos entre 1,3V y 2,3V)(tener al menos 5 útiles).
- Repetir tomas de datos a T y U_2 constantes con variaciones de U_H (entre 5,8V y 6,8V)(tener al menos 5 útiles).
- Tomar datos de la actividad 1 y realizar el análisis cualitativo correspondiente.
- Desarrollar el anásis preliminar de los datos tomados.

Esta vez pusimos la punta de la termocupla en contacto directo con el tubo para tener una mejor lectura de su temperatura. Esto resultará en que las temperaturas medidas durante la sesión de hoy difieran de las temperaturas medidas durante la sesión anterior.

Al intentar tomar datos con {U1=60,00V. U2=1,5V. UH=6.3V. T=205°C} el sensor de corriente se satura. Esto no es típico y tampoco es lo esperado. Se sospecha que la termocupla está descalibrada o malfuncionando.

Probamos medir la temperatura con dos termocuplas distintas al mismo tiempo y obtuvimos más o menos la misma temperatura. Con esto, descartamos un error en la termocupla.

ACTIVIDAD 02. EJERCICIO 01. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (AL MENOS 5 SERIES ÚTILES)

TOMA DE DATOS: Las series de datos se llaman: ACTIVIDAD 02. EJERCICIO 01. U1=60,00V, U2=2V. UH=6.3V. T=215°C (n).csv

- La toma ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (1) empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $214\ ^\circ\mathrm{C}$.
- La toma ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=214°C. (2) empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $213\ ^\circ\mathrm{C}$. Presenta la siguiente inconsistencia: aunque la toma tiene una temperatura menor, la magnitud de la corriente medida dismuniyó con respecto a la toma (1). (comportamiento contrario al esperado).
- La toma ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (3) empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $213\ ^\circ\mathrm{C}$.
- La toma ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (4) empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $211\ ^\circ\mathrm{C}$.
- La toma ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (5) empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $212\ ^\circ\mathrm{C}$.

La toma (5) fue realizada después de una pausa de una hora tras la toma (4), durante la cual el horno se enfrió.

In [60]:
```
"""Importación de datos automática la lista 'data' contiene las series de datos mostrado
correspondería a la medida cuyo paréntesis termina en (1), 'data[1]' correspondería a la
en (2) y así sucesivamente.
"""
data = []
for i in range(1,6):
    data.append(pd.read_csv("sesion2/ACTIVIDAD 02. EJERCICIO 01. U1 = 60,00 V, U2 = 2V.
```

Se tabulan los datos crudos

In [61]:
```
data[0].T
```

Out[61]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.36	0.37	0.38	0.38	0.36	0.39	0.35	0.39	0.35	0.37	...	39.28	39.42	39.69	39.87	40.12	40.32	4

2 rows × 2457 columns

In [62]:
```
data[1].T
```

Out[62]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.8	59.82	59.85	59.87	59.90	5
Corriente IA	0.33	0.32	0.38	0.38	0.43	0.39	0.43	0.39	0.44	0.38	...	20.82	21.0	21.22	21.39	21.61	21.84	2

2 rows × 2457 columns

In [63]:
```
data[2].T
```

Out[63]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.49	0.48	0.43	0.42	0.41	0.42	0.39	0.44	0.42	0.47	...	22.18	22.36	22.58	22.76	22.98	23.19	2

2 rows × 2457 columns

In [64]:
```
data[3].T
```

Out[64]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.54	0.52	0.46	0.44	0.38	0.45	0.39	0.44	0.42	0.47	...	15.29	15.47	15.58	15.80	15.91	16.14	1

2 rows × 2457 columns

In [65]:
```
data[4].T
```

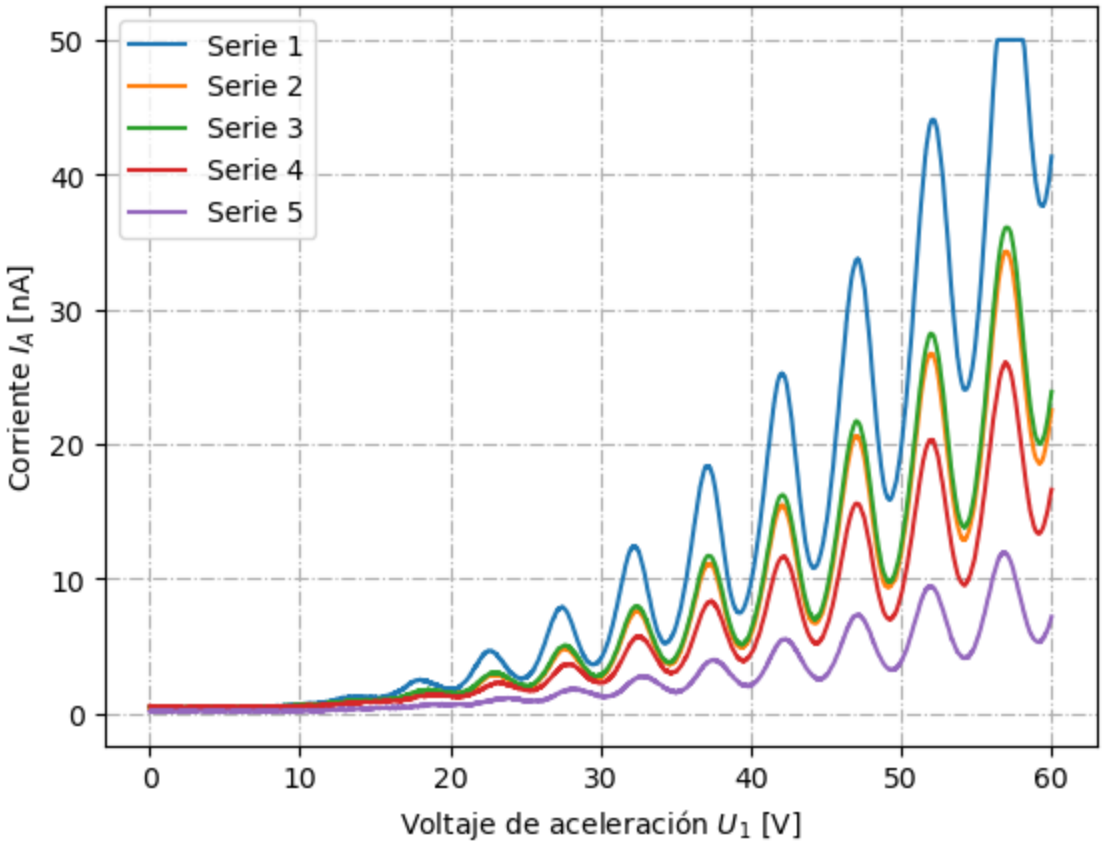
Out[65]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.8	59.82	59.85	59.87	59.90	5
Corriente IA	0.13	0.11	0.15	0.15	0.17	0.13	0.15	0.13	0.17	0.12	...	6.38	6.4	6.55	6.57	6.71	6.77	

2 rows × 2457 columns

In [66]:
```
for i in range(len(data)):
    plt.plot(data[i]["Voltage U1"].values,data[i]["Corriente IA"].values,label="Serie %s
plt.grid(linestyle="-.")
plt.ylabel(r"Corriente $I_A$ [nA]")
plt.xlabel(r"Voltaje de aceleración $U_1$ [V]" )
plt.legend()
```

Out[66]:
```
<matplotlib.legend.Legend at 0x1eb3d4eee80>
```



Las gráficas anteriores corresponden a las series de datos tomados para el experimento de Franck-Hertz para una temperatura constante de 215°C con ciertas fluctuaciones de ± 2°C que no pudieron ser corregidas con la estufa. Si bien estas también presentan fluctuaciones en la magnitud de la corriente que no es consistente con lo esperado, hay 3 medidas (serie 2, serie 3 y serie 4) que no parecen variar mucho, a diferencia de las medidas "serie 1" y "serie 2". Creemos que este efecto es debido a problemas con la termocupla o, incluso, con el posicionamiento en la misma dentro de la estufa pues no se puede poner exactamente en la misma posición que en la sesión 1.

ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=X°C. (AL MENOS 5 SERIES ÚTILES)

TOMA DE DATOS: Las tomas de datos se llaman

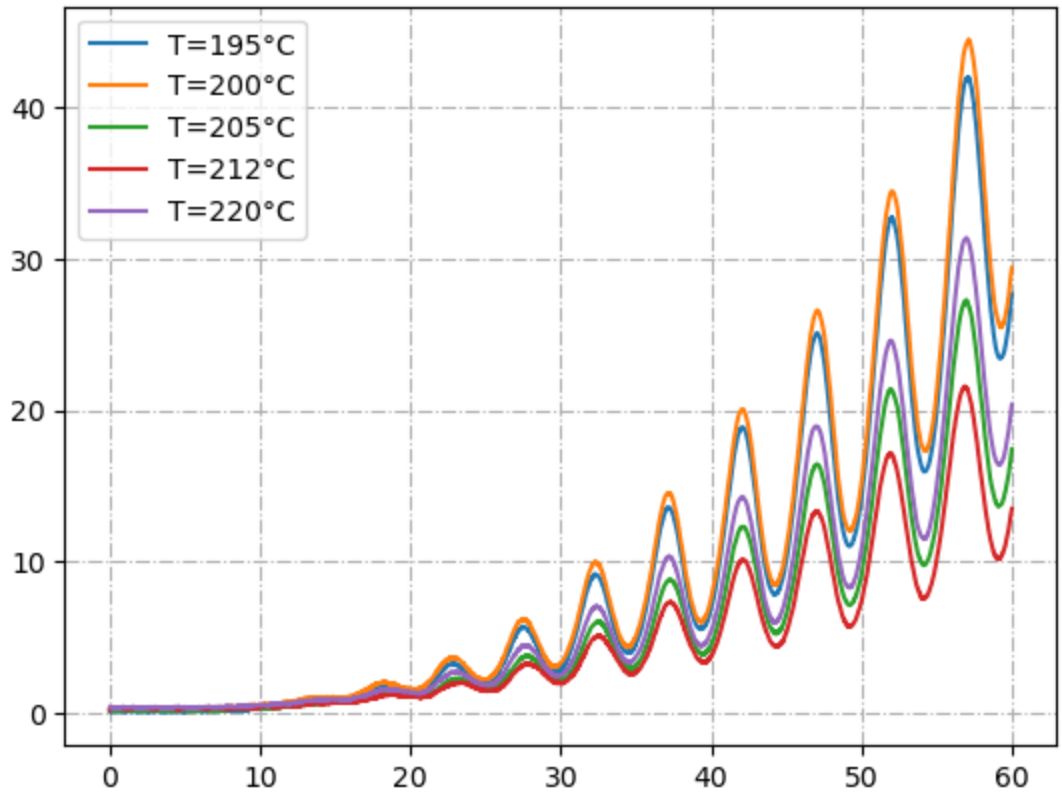
- ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=215°C. (1-5).
- el número entre paréntesis indica la toma de datos.

- ACTIVIDAD 02. EJERCICIO 02. U1=60,00V. U2=2V. UH=6.3V. T=°C empezó con $T = 215\ ^\circ\mathrm{C}$ y bajó hasta $214\ ^\circ\mathrm{C}$.

In [67]:
```
#Importación de series de datos a diferentes temperaturas
#Las temperaturas que se tomaron fueron 195°C, 200°C, 205°C, 212°C y 220°C
Temp = ["195°C", "200°C", "205°C", "212°C" , "220°C"]
nombre = "sesion2/ACTIVIDAD 02. EJERCICIO 02. U1 = 60,00 V, U2 = 2V. UH = 6,3V. T="

datos_temp = []

for i  in range(len(Temp)):
    datos_temp.append(pd.read_csv(nombre + Temp[i]+".csv",sep=" ",decimal=','))
```

```
for i in range(len(datos_temp)):
    plt.plot(datos_temp[i]["Voltage U1"].values,datos_temp[i]["Corriente IA"].values,lab

plt.grid(linestyle="-.")
plt.legend()
```

Out[67]: <matplotlib.legend.Legend at 0x1eb3d4fa4f0>



Las series de datos en crudo se muestran a continuación:

In [68]:
```
datos_temp[0].T
```

Out[68]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.10	0.13	0.11	0.12	0.10	0.14	0.13	0.16	0.12	0.15	...	25.67	25.82	26.09	26.24	26.55	26.73	2

2 rows × 2457 columns

In [69]:
```
datos_temp[1].T
```

Out[69]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.40	0.39	0.35	0.35	0.32	0.34	0.34	0.37	0.36	0.39	...	27.51	27.76	27.97	28.22	28.44	28.69	2

2 rows × 2457 columns

In [70]:
```
datos_temp[2].T
```

Out[70]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.22	0.19	0.23	0.20	0.24	0.19	0.25	0.19	0.26	0.20	...	15.78	15.89	16.12	16.28	16.51	16.63	1

2 rows × 2457 columns

In [71]:
```
datos_temp[3].T
```

Out[71]:

	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.8	59.82	59.85	59.87	59.9	5
Corriente IA	0.20	0.23	0.28	0.29	0.34	0.30	0.35	0.31	0.32	0.28	...	12.16	12.3	12.42	12.56	12.73	12.9	1

2 rows × 2457 columns

In [72]:
```
datos_temp[4].T
```

Out[72]:

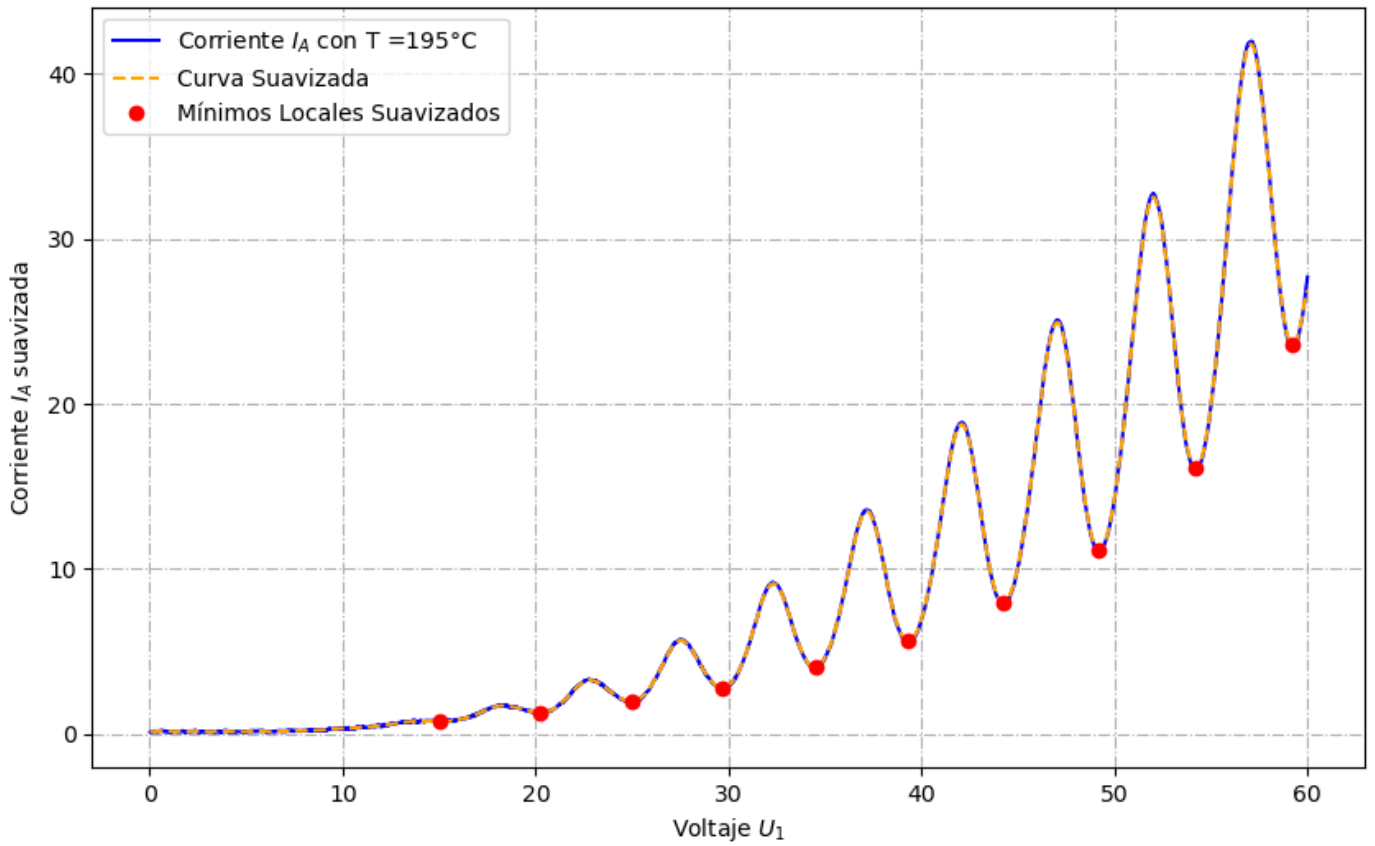
	0	1	2	3	4	5	6	7	8	9	...	2447	2448	2449	2450	2451	2452	2
Voltage U1	0.02	0.04	0.07	0.09	0.12	0.14	0.17	0.19	0.21	0.24	...	59.78	59.80	59.82	59.85	59.87	59.90	5
Corriente IA	0.37	0.36	0.38	0.39	0.41	0.36	0.40	0.35	0.40	0.36	...	18.65	18.79	19.03	19.17	19.43	19.59	1

2 rows × 2457 columns

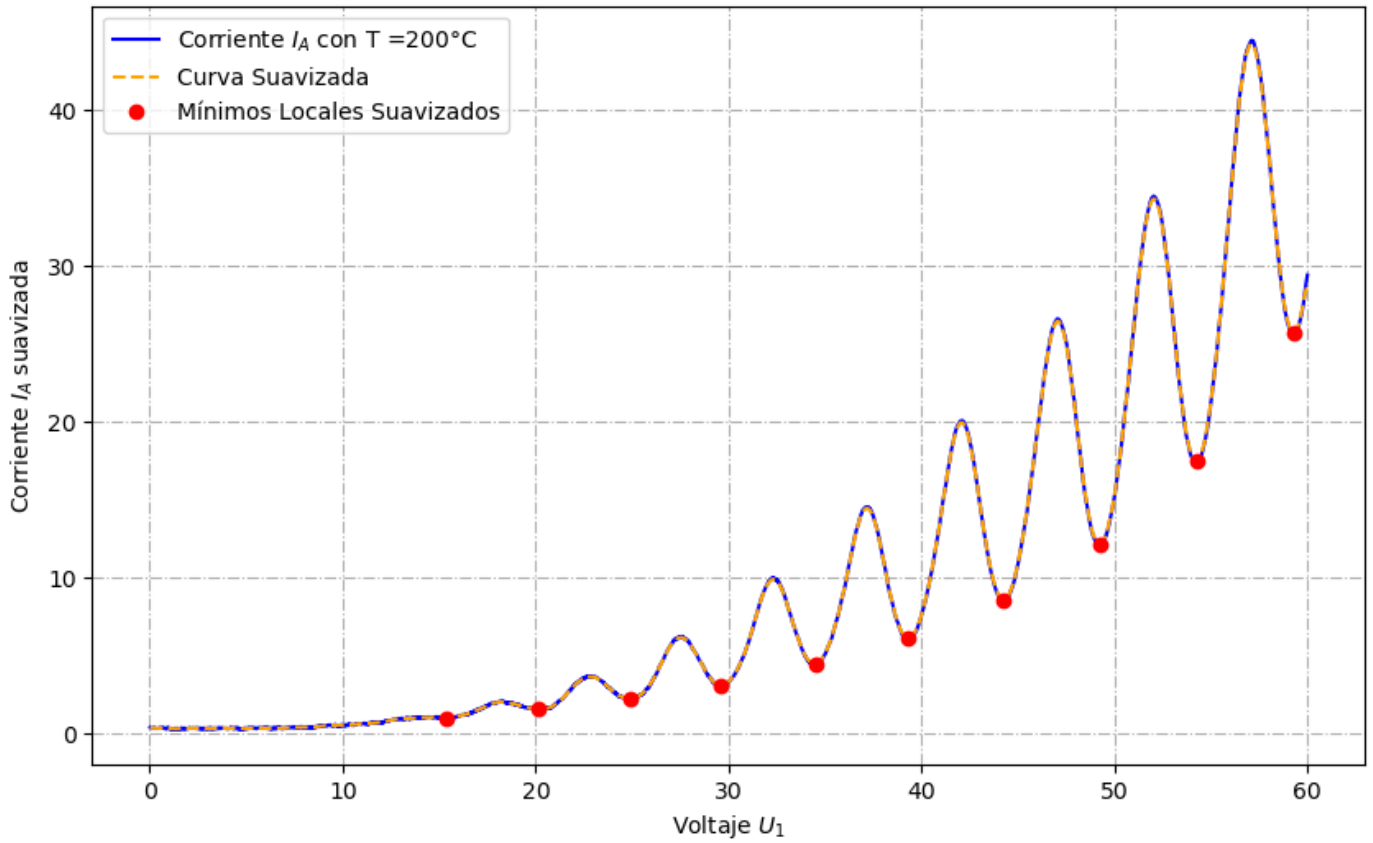
ACTIVIDAD 02. EJERCICIO 03. CON LOS DATOS DE DE TEMPERATURAS DIFERENTES

In [73]:
```
N = [10,10,10,10,10]
deltas_temp = []
uncer_deltas_temp = []
i = 0
for serie in datos_temp:
    voltaje = serie['Voltage U1'].values
    corriente = serie['Corriente IA'].values
    # Suavizar la curva usando un filtro gaussiano
    smoothed_corriente = gaussian_filter1d(corriente, sigma=5.2)
    # Encontrar el mínimo de la curva suavizada
    min_envolvente_index = np.argmin(smoothed_corriente)
    # Encontrar los mínimos locales en la curva suavizada
    minima_suavizada_indices, _ = find_peaks(-smoothed_corriente)  # Invertir la corrien
    # Crear un DataFrame con los resultados de los mínimos locales
    minima_suavizada_voltajes = voltaje[minima_suavizada_indices]
    minima_suavizada_corrientes = smoothed_corriente[minima_suavizada_indices]

    plt.figure(figsize=(10, 6))
    plt.plot(voltaje, corriente, label=r"Corriente $I_A$ con T =%s"%(Temp[i]), color='bl
    plt.plot(voltaje, smoothed_corriente, label="Curva Suavizada", color='orange', lines
    plt.plot(voltaje[minima_suavizada_indices][-N[i]:], smoothed_corriente[minima_suaviz
    plt.xlabel(r'Voltaje $U_1$')
```

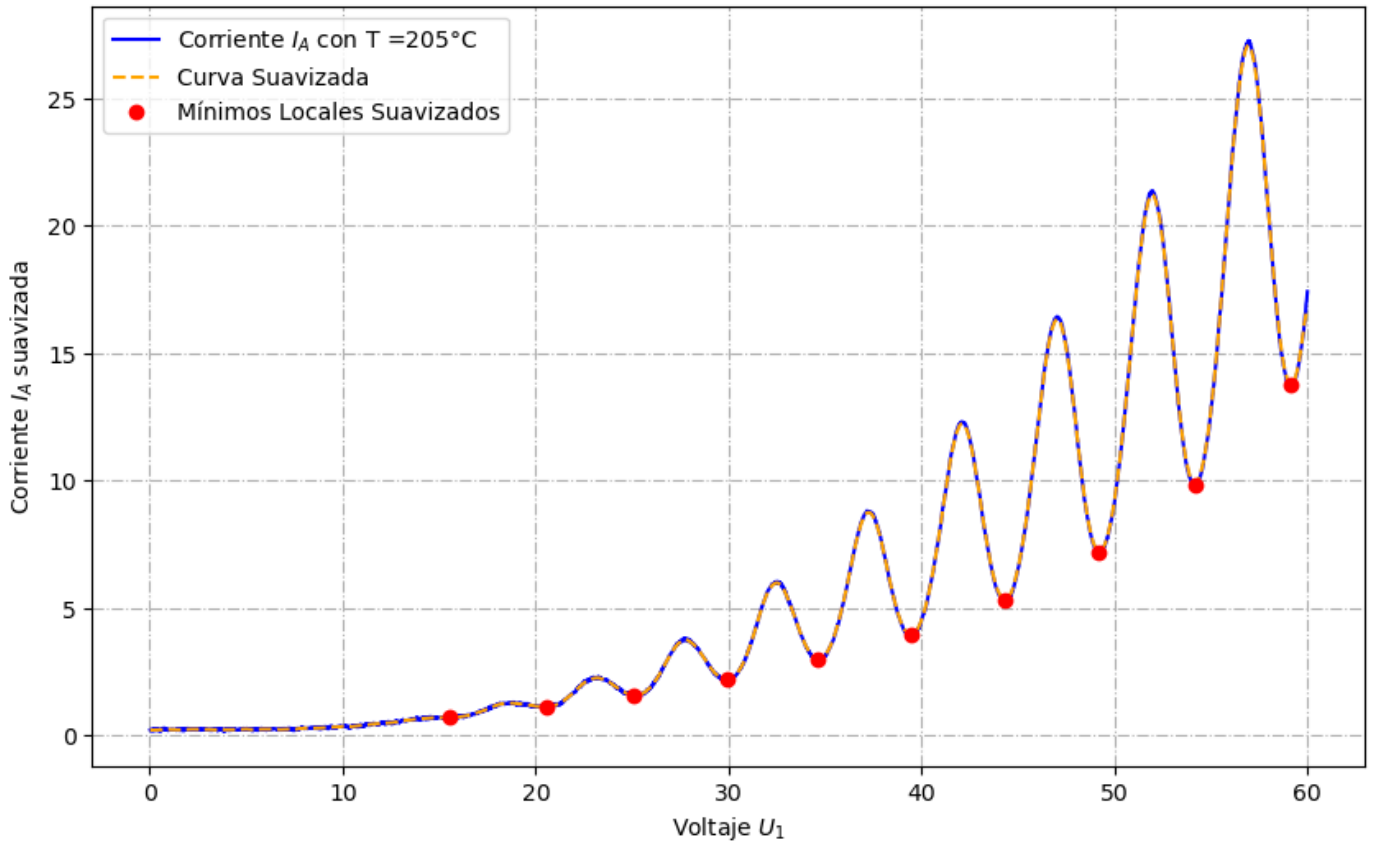
```
    plt.ylabel(r'Corriente $I_A$ suavizada')
    plt.legend()
    plt.grid(linestyle='-.')
    plt.show()

    min_import = minima_suavizada_voltajes[-N[i]:]

    delta2 = np.zeros(len(min_import)-1)

    for j in range(len(delta2)):
        delta2[j] = min_import[j+1] - min_import[j]

    deltas_temp.append(delta2.copy())
    uncer_deltas_temp.append(np.sqrt(2)*np.ones_like(delta2[-N[i]:])*0.01)


    print("Diferencia entre mínimos: ",delta2)
    print("Incertidumbres: ",np.sqrt(2)*np.ones_like(delta2[-N[i]:])*0.01)
    i+=1
```



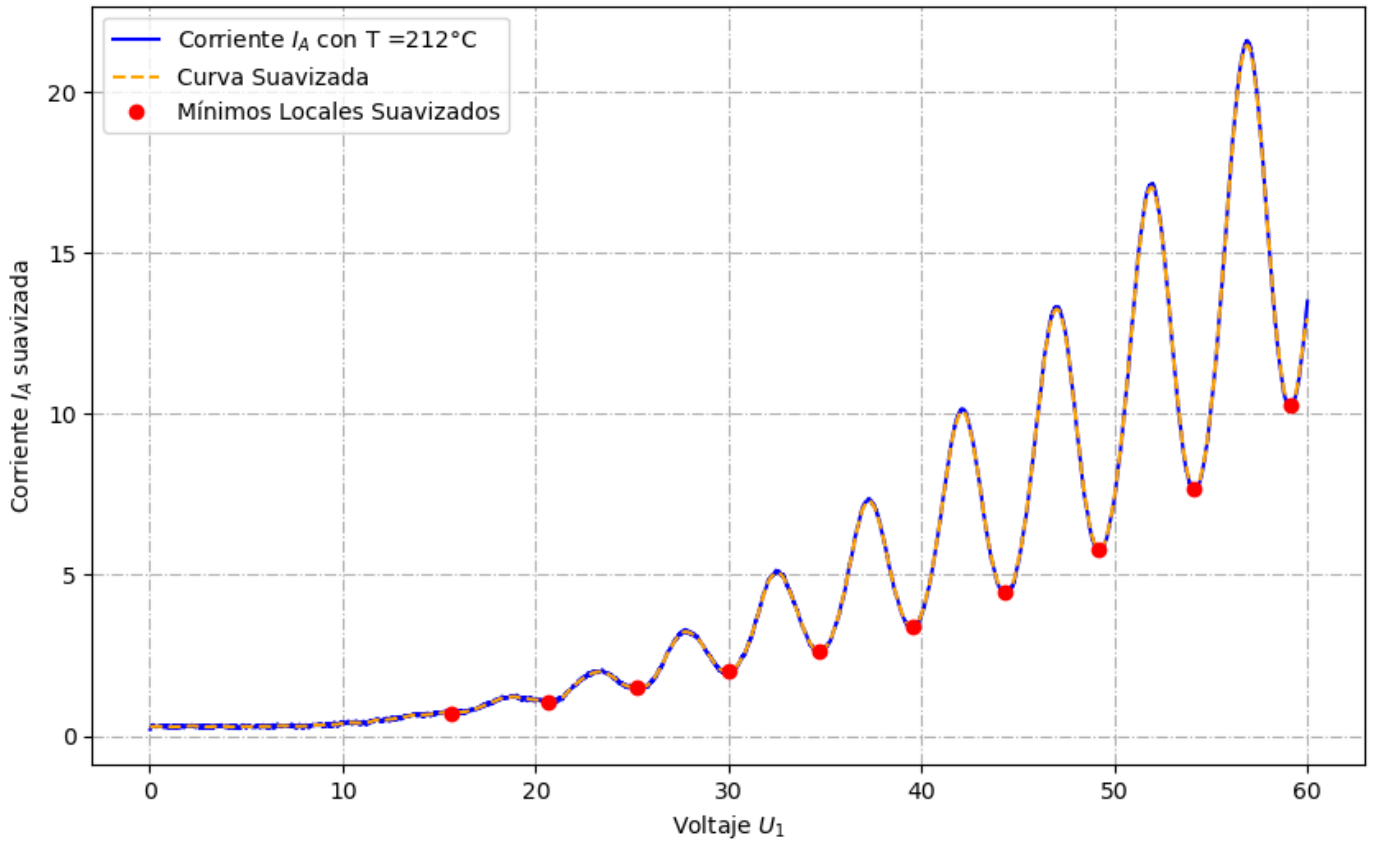

```
Diferencia entre mínimos:  [5.18 4.84 4.61 4.84 4.83 4.93 4.96 5.01 5.03]
Incertidumbres:  [0.01414214 0.01414214 0.01414214 0.01414214 0.01414214 0.01414214
 0.01414214 0.01414214 0.01414214]
```



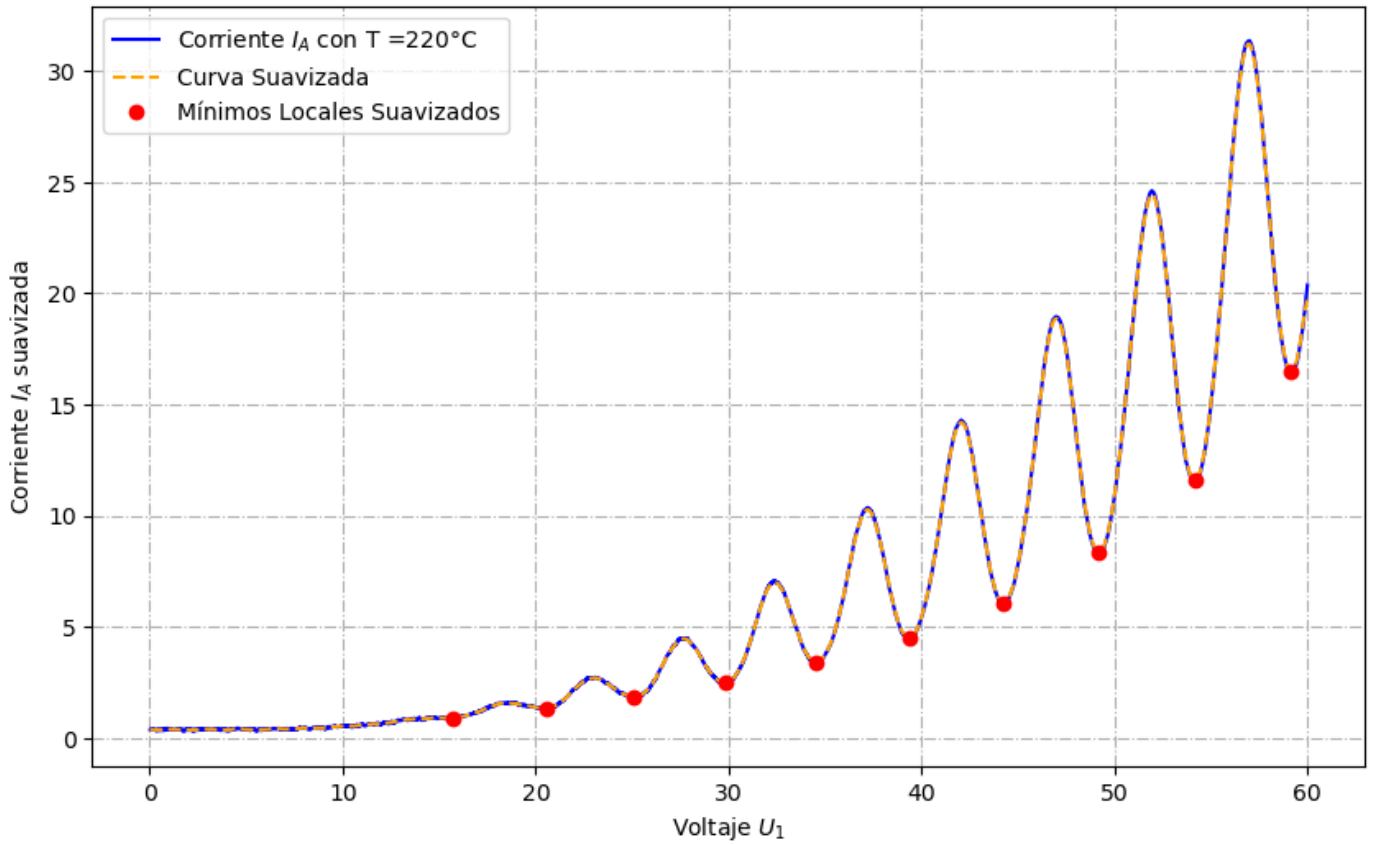

```
Diferencia entre mínimos:  [4.84 4.76 4.66 4.89 4.83 4.93 4.99 5.   5.06]
Incertidumbres:  [0.01414214 0.01414214 0.01414214 0.01414214 0.01414214 0.01414214
 0.01414214 0.01414214 0.01414214]
```




```
Diferencia entre mínimos:  [4.98 4.54 4.86 4.69 4.84 4.86 4.88 4.96 4.98]
Incertidumbres:  [0.01414214 0.01414214 0.01414214 0.01414214 0.01414214 0.01414214
 0.01414214 0.01414214 0.01414214]
```




```
Diferencia entre mínimos:  [5.   4.62 4.71 4.69 4.86 4.81 4.86 4.93 4.96]
Incertidumbres:  [0.01414214 0.01414214 0.01414214 0.01414214 0.01414214 0.01414214
 0.01414214 0.01414214 0.01414214]
```




```
Diferencia entre mínimos:  [4.86 4.54 4.74 4.71 4.84 4.88 4.96 4.96 5.  ]
Incertidumbres:  [0.01414214 0.01414214 0.01414214 0.01414214 0.01414214 0.01414214
 0.01414214 0.01414214 0.01414214]
```

T (°C)	σ_T (°C)	<ΔE> (eV)	σ_<ΔE> (eV)
195	1	4,91	0,01
200	1	4,88	0,01
205	1	4,84	0,01
212	1	4,82	0,01
220	1	4,83	0,01

Como se puede observar en las listas correspondientes a cada serie con temperatura diferentes, se puede ver que estas son diferentes al ΔU_1 esperada de $4.67\ e\mathrm{V}$ y aumentan con forme se aumenta la temperatura. Esto es debido a la energía cinética adicional del electrón antes de colisionar con un átomo de mercurio.

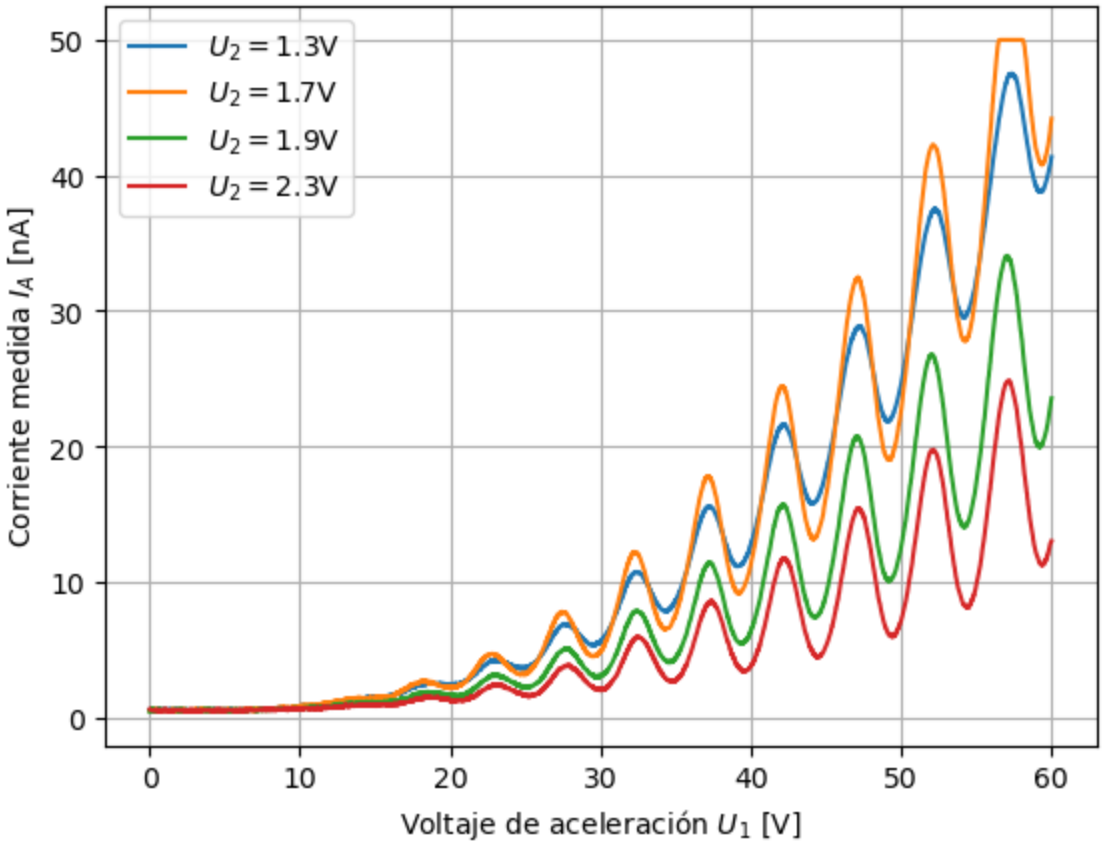
ACTIVIDAD 02. EJERCICIO 04.

Las discrepancias entre la distancia calculada entre mínimos y el valor esperado de 4.67eV se puede explicar teniendo en cuenta la función trabajo del amperímetro, puede este se debe llevar un poco de energía.

ACTIVIDAD 02. EJERCICIO 05.

Consideramos que las medidas tomadas en la sesión anterior no presentaban discrepancias tan significativas como para volver a tomar datos. Estos se muestran a continuación:

In [74]:
```
datos1 = pd.read_csv("ACTIVIDAD 02.05. TOMA 01. T=210. U_2=1.7.csv",sep="       ")
datos2 = pd.read_csv("ACTIVIDAD 02.05. TOMA 02. T=210. U_2=1.9.csv",sep="       ")
datos3 = pd.read_csv("ACTIVIDAD 02.05. TOMA 03. T=210. U_2=1.3.csv",sep="       ")
datos4 = pd.read_csv("ACTIVIDAD 02.05. TOMA 04. T=210. U_2=2.13.csv",sep="      ")

plt.plot(datos3["Voltage U1"].values,datos3["Corriente IA"].values, label=r"$U_2 = 1.3$V
plt.plot(datos1["Voltage U1"].values,datos1["Corriente IA"].values, label=r"$U_2 = 1.7$V
plt.plot(datos2["Voltage U1"].values,datos2["Corriente IA"].values, label=r"$U_2 = 1.9$V
plt.plot(datos4["Voltage U1"].values,datos4["Corriente IA"].values, label=r"$U_2 = 2.3$V
plt.legend()
plt.xlabel(r"Voltaje de aceleración $U_1$ [V]")
plt.ylabel(r"Corriente medida $I_A$ [nA]")
plt.grid()
```



ACTIVIDAD 02. EJERCICIO 06. U1=60,00V. U2=2V. UH=XV. T=212°C. (AL MENOS 5 SERIES ÚTILES)

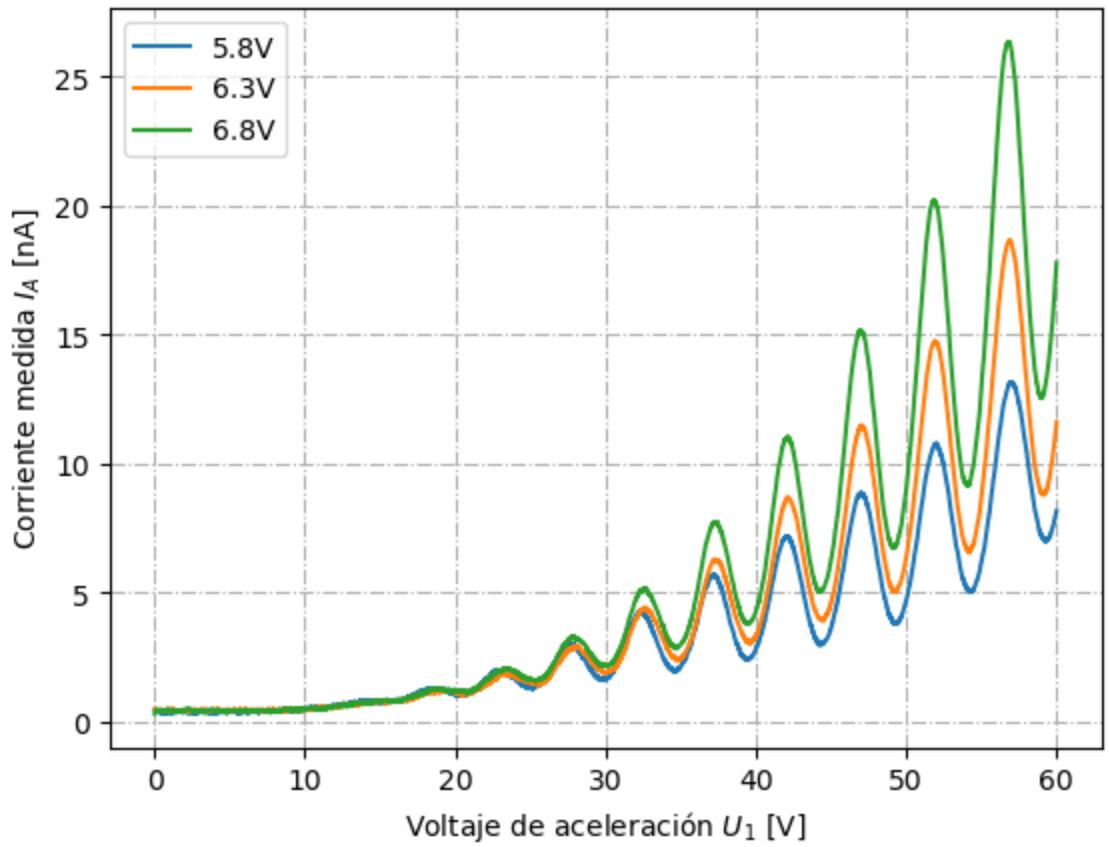
LAS SERIES DE DATOS SE LLAMAN:

- ACTIVIDAD 02. EJERCICIO 06. U1=60,00V. U2=2V. UH=XV. T=212°C. (1-5).
- el número entre paréntesis indica la toma de datos.

- ACTIVIDAD 02. EJERCICIO 06. U1=60,00V. U2=2V. UH=6.8V. T=212°C empezó con $T = 212\ ^\circ\mathrm{C}$ y bajó hasta $209\ ^\circ\mathrm{C}$.
- ACTIVIDAD 02. EJERCICIO 06. U1=60,00V. U2=2V. UH=6.8V. T=212°C empezó con $T = 212\ ^\circ\mathrm{C}$ y bajó hasta $210\ ^\circ\mathrm{C}$.

In [75]:
```
U_H = ["5,8","6,3","6,8"]
datos_U_H = []

for i in range(len(U_H)):
    datos_U_H.append(pd.read_csv("sesion2/ACTIVIDAD 02. EJERCICIO 06. U1 = 60,00 V, U2 =

U_H = ["5.8","6.3","6.8"]

for i in range(len(U_H)):
    plt.plot(datos_U_H[i]["Voltage U1"].values,datos_U_H[i]["Corriente IA"].values, labe

plt.grid(linestyle="-.")
plt.xlabel(r"Voltaje de aceleración $U_1$ [V]")
plt.ylabel(r"Corriente medida $I_A$ [nA]")
plt.legend()
```

Out[75]:
```
<matplotlib.legend.Legend at 0x1eb3c12e610>
```



La gráfica muestra cómo al aumentar el voltaje U_H se aumenta la magnitud de la corriente que mide el amperímetro dentro de la estufa. Este es el comportamiento que se esperaba. Esto se puede explicar teniendo en cuenta que al aumentar U_H aumenta la temperatura del electrodo que libera a los electrones, los cuales siguen una distribución de Maxwell-Boltzmann en cuando a velocidades y, por tanto, en energía cinética. Esto es, entre más voltaje U_H introducido, más temperatura puede obtener el electrodo y, por ende, más electrones que tienen una velocidad que les permitiría pasar por el voltaje U_2 sin dificultad.

ACTIVIDAD 02. EJERCICIO 07. Camino medio libre

In [76]:
```
#Tomando los datos de los deltas de voltaje calculados anteriormente
deltas_temp, uncer_deltas_temp
deltas_table = []
for i in range(len(deltas_temp)):
    deltas_table.append(pd.DataFrame({r"$\Delta U_1$ (T = %s)"%(Temp[i]):deltas_temp[i],
```

In [77]:
```
deltas_table[0]
```

Out[77]:

	ΔU_1 (T = 195°C)	Incertidumbre $\sigma_{\Delta U_1}$:
0	5.18	0.014142
1	4.84	0.014142
2	4.61	0.014142
3	4.84	0.014142
4	4.83	0.014142
5	4.93	0.014142

6	4.96	0.014142
7	5.01	0.014142
8	5.03	0.014142

In [78]:
```
deltas_table[1]
```

Out[78]:

	ΔU_1 (T = 200°C)	Incertidumbre $\sigma_{\Delta U_1}$:
0	4.84	0.014142
1	4.76	0.014142
2	4.66	0.014142
3	4.89	0.014142
4	4.83	0.014142
5	4.93	0.014142
6	4.99	0.014142
7	5.00	0.014142
8	5.06	0.014142

In [79]:
```
deltas_table[2]
```

Out[79]:

	ΔU_1 (T = 205°C)	Incertidumbre $\sigma_{\Delta U_1}$:
0	4.98	0.014142
1	4.54	0.014142
2	4.86	0.014142
3	4.69	0.014142
4	4.84	0.014142
5	4.86	0.014142
6	4.88	0.014142
7	4.96	0.014142
8	4.98	0.014142

In [80]:
```
deltas_table[3]
```

Out[80]:

	ΔU_1 (T = 212°C)	Incertidumbre $\sigma_{\Delta U_1}$:
0	5.00	0.014142
1	4.62	0.014142
2	4.71	0.014142
3	4.69	0.014142
4	4.86	0.014142
5	4.81	0.014142
6	4.86	0.014142

7	4.93	0.014142
8	4.96	0.014142

In [81]:
```
deltas_table[4]
```

Out[81]:

	ΔU_1 (T = 220°C)	Incertidumbre $\sigma_{\Delta U_1}$:
0	4.86	0.014142
1	4.54	0.014142
2	4.74	0.014142
3	4.71	0.014142
4	4.84	0.014142
5	4.88	0.014142
6	4.96	0.014142
7	4.96	0.014142
8	5.00	0.014142

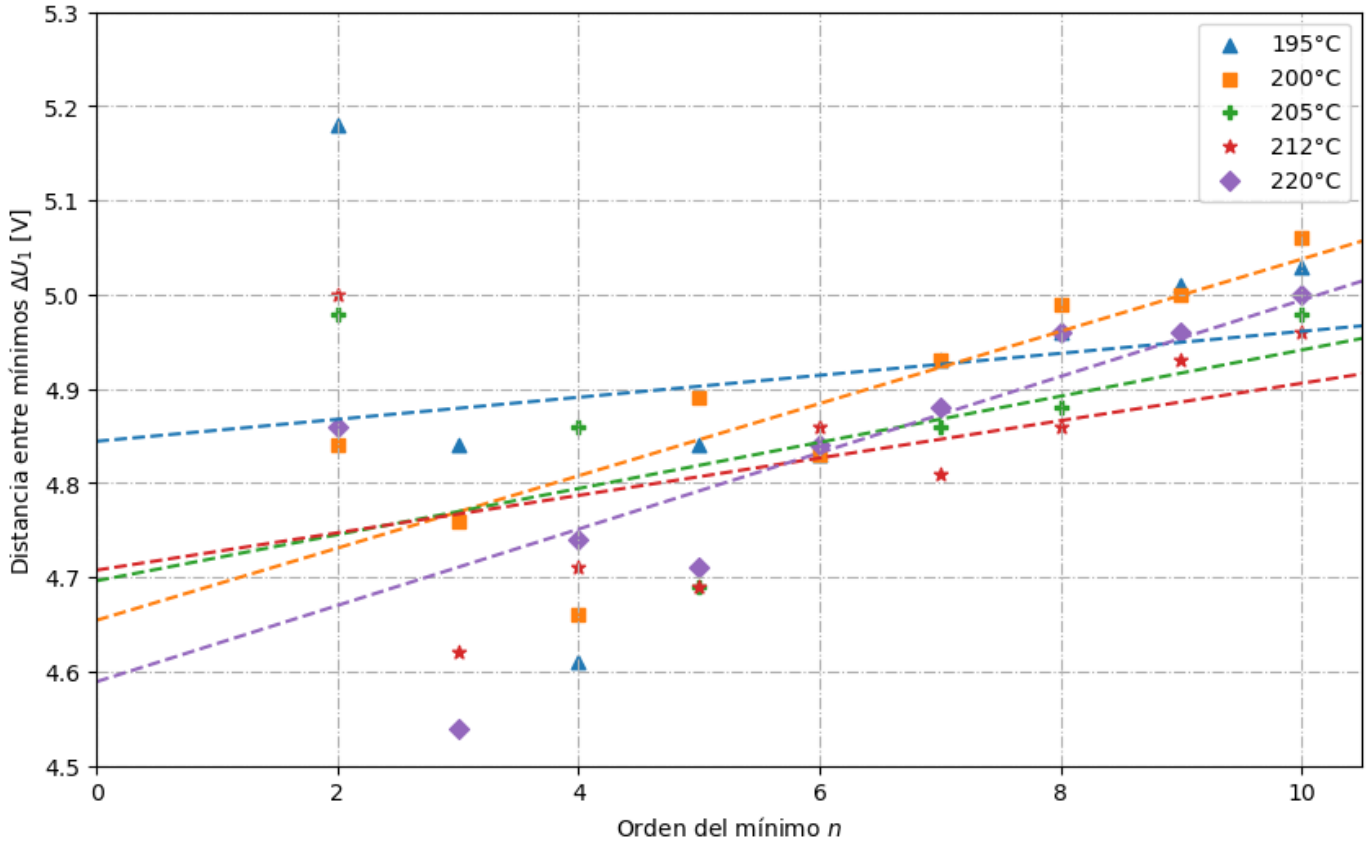
Las tablas anteriores corresponden a las diferencias de energías entre mínimos para cada temperatura junto con la incertidumbre asociada a las mediciones.

In [82]:
```
mark = ["^","s","P","*","D"]
nlist = []
pendientes = []
interceptos = []
ax = plt.figure(figsize=(10, 6))
ax.set
for i in range(len(deltas_temp)):
    n = np.array(list(range(1,len(deltas_temp[i])+1))) + np.ones_like(list(range(1,len(d
    nlist.append(n)
    plt.scatter(n,deltas_temp[i],marker=mark[i],label=Temp[i])

plt.xlabel("Orden del mínimo $n$")
plt.ylabel(r"Distancia entre mínimos $\Delta U_1$ [V]" )
plt.grid(linestyle="-.")


x_space = np.linspace(0,11)
for i in range(len(deltas_temp)):
    print("Para la regresión de temperatura T="+Temp[i])
    params = utils.linear_w_regression(nlist[i],deltas_temp[i],1*np.ones_like(deltas_tem
    pendientes.append(params[0])
    interceptos.append(params[2])
    print(30*"#")
    plt.plot(x_space,params[0]*x_space + params[2],linestyle="--")

plt.axis([0, 10.5, 4.5, 5.3])
plt.legend()
```

```
Para la regresión de temperatura T=195°C
m: 0.011666666666666667 uncertanty m: 0.0018257418583505533
b: 4.844444444444485 uncertanty b: 0.011925695879998876
##############################
Para la regresión de temperatura T=200°C
m: 0.03833333333333333 uncertanty m: 0.0018257418583505533
b: 4.654444444444449 uncertanty b: 0.011925695879998876
##############################
```

```
Para la regresión de temperatura T=205°C
m: 0.0245 uncertanty m: 0.0018257418583505533
b: 4.696333333333338 uncertanty b: 0.011925695879998876
##############################
Para la regresión de temperatura T=212°C
m: 0.019833333333333335 uncertanty m: 0.0018257418583505533
b: 4.707666666666671 uncertanty b: 0.011925695879998876
##############################
Para la regresión de temperatura T=220°C
m: 0.0405 uncertanty m: 0.0018257418583505533
b: 4.589222222222227 uncertanty b: 0.011925695879998876
##############################
```

Out[82]: `<matplotlib.legend.Legend at 0x1eb3bf17610>`



Las regresiones lineales, en efecto, presentan un aumento de distancia a medida que se aumente en el orden del mínimo. Sin embargo, no siguen el comportamiento esperado: bajar su pendiente a medida que la temperatura aumenta.

In [83]:
```
L = 12e-2 #metros
Ea = 4.67 #eV

camino_medio_libre = np.array(pendientes)*L/(2*Ea)
print(camino_medio_libre)
```

```
[0.00014989 0.00049251 0.00031478 0.00025482 0.00052034]
```

T (°C)	σ_T (°C)	L (m)	E_a (eV)	m (eV)	σ_m (eV)	<l_exp> (m)	σ_<l_exp> (m)
195	1			0,012		0,00015	
200	1			0,038		0,00049	
205	1	0,12	4,67	0,024	0,002	0,00031	2E-05
212	1			0,020		0,00025	
220	1			0,040		0,00052	

Estos valores de camino libre medio están por encima de lo esperado en un orden de magnitud, esto es,

diez veces más de los esperado.

Ahora, en cuanto a la energía E_a calculada por medio de las regresiones lineales, se obtuvieron los siguientes valores correspondientes a las temperaturas $195\ ^\circ\mathrm{C}$, $200\ ^\circ\mathrm{C}$, $205\ ^\circ\mathrm{C}$, $212\ ^\circ\mathrm{C}$ y $220\ ^\circ\mathrm{C}$, respectivamente

In [84]:
```
for i in range(len(Temp)):
    print("Para la temperatura T = %s la energía E_a es:"%(Temp[i]),pendientes[i]*0.5 +i
```

```
Para la temperatura T = 195°C la energía E_a es: 4.850277777777782
Para la temperatura T = 200°C la energía E_a es: 4.673611111111116
Para la temperatura T = 205°C la energía E_a es: 4.708583333333338
Para la temperatura T = 212°C la energía E_a es: 4.717583333333337
Para la temperatura T = 220°C la energía E_a es: 4.609472222222227
```

Los valores obtenidos para esta energía están muy cerca del valor esperado de $4.67 e\mathrm{V}$, especialmente aquel correspondiente a la temperatura $T = 200^\circ\mathrm{C}$.

T (°C)	σ_T (°C)	b (eV)	σ_b (eV)	m (eV)	σ_m (eV)	E_a (eV)	σ_E_a (eV)
195	1	4,84		0,012		4,85	
200	1	4,65		0,038		4,67	
205	1	4,70	0,01	0,024	0,002	4,71	1E-02
212	1	4,70		0,020		4,72	
220	1	4,59		0,040		4,61	