Experiment n - NAME

Date: Data taken by:

Equipment used:

1. Instrument 1 (LC: 0.1mm)
2. Instrument 2 (LC: 0.2V)
3. ...

In [ ]:
```
######****************************** INPUT YOUR DATA HERE ***************************######

data_file = "../data/exp_n_data_1.csv"                # Add the name of the data file (csv or txt).

x_label = r'x-axis (unit - e.g. $\mu$ A)'             # X axis labels
y_label = r'y-axis (unit)'                            # Y axis labels

round_slope = 3                                       # Number of digits to round-off slope
round_intercept = 3                                   # Number of digits to round-off intercept

#Once this is done, run the rest of the code: it should print a well formatted graph.

#*******************************************************************************#
```

In [ ]:
```
import numpy as np                                    # Importing the NumPy package

import matplotlib.pyplot as plt                       # Importing the Matplotlib package for plotting
                                                      # "Magic" to display images inline
%matplotlib inline

import scipy as scp                                   # Importing the SciPy package
from scipy.optimize import curve_fit                  # Importing the curve fitting module from SciPy
```

In [ ]:
```
x,y= np.loadtxt(data_file, delimiter=",",unpack=True) # Imports the data located in `data_file`,
                                                      # unpacking it such that the first column is
                                                      # stored in the variable `xpoints`, and the
                                                      # second in the variable `ypoints`.
```

In [ ]:
```
def f(x, a, b):                                       # Define a function `f` which `returns` a value o
    return a*x + b                                    # This line makes sure that the function returns

par, covariance = curve_fit(f, x, y)

m = np.round(par[0],round_slope)                      # For simplicity, we assign the values of par[] t
c = np.round(par[1],round_intercept)                  # and c, rounded off approrpiately using the np.r
```

In [ ]:
```
ytrend = m*x+c                                        # Create an array of y values corresponding to th
                                                      # which satisfy the trendline with the given slop


#####*****************  Displaying a trendline on the graph area  ******************#####

eqn = 'y(x) = '+f'{m:.3f}'+'x'+f'{c:+}'               # Equation of trendline as a string; we'll print

xmin = np.min(x)*1.1                                  # Coordinates to place the trendline: I have chos
ymax = np.max(y)*0.9                                  # minimum x-coordinate and the maximum y-coordina
                                                      # points like (14,2), for example.


plt.text(xmin, ymax,eqn,fontsize=12)                  # Adding the equation string to the graph area, a

#*******************************************************************************#

plt.scatter(x,y)                                      # Plotting the data-points
plt.plot(x, ytrend, '--',color="darkorange")          # Plotting an orange trend-line

plt.xlabel(x_label)                                   # Formatting the axes
plt.ylabel(y_label)
plt.show();
```