CHAPTER – 4

MEASUREMENT AND ANALYSIS

26 | Page

4.1 INTRODUCTION:

For moving of it is need that both Azimuth and Elevation angles. It is need some

current to move both azimuth housing and elevation housing. Two vertical brushless

AC (BLAC) servo motors, two gearboxes, and couplings are all included in the

Azimuth housing. For installing gearboxes, limit switches, and encoder assemblies,

several cutouts are also offered. They aid in lubricating the gears as well. The slew

ring bearing is attached to the top of the azimuth housing using high tension bolts.

The housings and the slew ring bearing are precisely manufactured to fit. The

housing has an access door in the middle. The Azimuth housing is connected to the

Azimuth slew ring bearing by high tension fasteners. The inner ring of the azimuth

slew ring bearing is where these bolts are attached. For handling, there are two

lifting hooks available. There are two horizontal BLAC servo motors inside the

Elevation housing. On it are installed the encoder assembly, limit switch, and

gearboxes. Additionally, several cut outs are provided for the mounting of encoder

assemblies, limit switches, and gearboxes. They aid in lubricating the gears as well.

Elevation slew ring bearings come in pairs. The Azimuth slew ring bearing and the

bottom of the Elevation housing have been precisely machined to fit. High tensile

steel bolts are used for the fitting. Cables are given some access. High strength steel

bolts are used to secure the Elevation housing in place after the bottom was

machined to fit the moveable portion of the Azimuth slew ring bearing. The arms of

the yoke are connected to yoke fixing plates via the slew ring bearings of the

elevation to the housing of the elevation. The shock absorber assemblies are

mounted using buffer assembly brackets. For handling purposes, there are four

lifting hooks available.

4.2 Utility Work:

To simplify the process of calculating the given currents to move the antenna, this

helps avoid manual calculation by converting the DOT file into Excel and creating

graphs on your own. Here, it is developed as a GUI (graphical user interface) using

Python. Azimuth housing and elevation housing consist of two motors to move the

antenna. As the first step, we have to give the DOT file, which contains both azimuth

(M1 and M2 current) and elevation current. By taking that DOT file as the input file, it

is analyzed and read that the data present in the file is calculated as the average of each

five degrees of both M1 and M2 currents and elevation. It will create a secondary graph

(with a single x axis and two y axis) according to the average values we got from the

output Excel. We can identify the output excel by giving the name as the output,

starting with the excel name. For better visual purposes, it created two output

secondary graphs, one for M1 and another for M2.

Python Code:

import pandas as pd

27 | Page

import math

from openpyxl.chart import LineChart

from openpyxl import Workbook

import os

import tkinter as tk

from tkinter import filedialog

import shutil

# Define your existing code as a function

def process\_dot\_file(file\_path):

# Your existing code here

x = os.path.dirname(file\_path)

y = os.path.basename(file\_path)

a = &quot;out.text&quot;

shutil.copy(file\_path, a)

a1 = &quot;output1.csv&quot;

with open(a, &#39;r&#39;) as b1:

lines = [line.strip().split() for line in b1]

csv\_lines = [&#39;,&#39;.join(line) for line in lines]

with open(a1, &#39;w&#39;) as h1:

h1.write(&#39;\n&#39;.join(csv\_lines))

os.remove(a)

df = pd.read\_csv(a1)

28 | Page

os.remove(a1)

z = df.columns.tolist()

i = 0

s = 0

s1 = 0

p = 0

L = []

L1 = []

L3 = []

L4 = []

M = []

M1 = []

j = math.floor(df.loc[0, &quot;ElActual&quot;]) + 5

H = 5

def G(M1):

l = 0

a = len(M1)

while(True):

k = (l + a) // 2

if M1[k] &lt; M1[k+1]:

l = k + 1

else:

a = k

if a == l:

break

29 | Page

return k

if(&quot;Avg\_El\_iM1&quot; not in z):

k = z.index(&quot;El\_iM1&quot;)

df.insert(k + 1, &quot;Avg\_El\_iM1&quot;, None)

df.insert(k + 3, &quot;Avg\_El\_iM2&quot;, None)

kum = G(df[&quot;ElActual&quot;])

while(i &lt;= kum):

if(df.loc[i, &quot;ElActual&quot;] &lt; j):

s = s + df.loc[i, &quot;El\_iM1&quot;]

s1 = s1 + df.loc[i, &quot;El\_iM2&quot;]

else:

if i - p != 0:

df.loc[i - 1, &quot;Avg\_El\_iM1&quot;] = s / (i - p)

L.append(format(s / (i - p), &quot;.3f&quot;))

s = 0

df.loc[i - 1, &quot;Avg\_El\_iM2&quot;] = s1 / (i - p)

L3.append(format(s1 / (i - p), &quot;.3f&quot;))

s1 = 0

p = i

M.append(H)

H = H + 5

j = j + 5

i = i + 1

L.append(format(s / (i - p), &quot;.3f&quot;))

L3.append(format(s1 / (i - p), &quot;.3f&quot;))

30 | Page

M.append(H)

i = kum + 1

p = i

j = math.floor(df.loc[i, &quot;ElActual&quot;]) - 5

s = 0

s1 = 0

while(i &lt; len(df[&quot;ElActual&quot;])):

if(df.loc[i, &quot;ElActual&quot;] &gt; j):

s = s + df.loc[i, &quot;El\_iM1&quot;]

s1 = s + df.loc[i, &quot;El\_iM2&quot;]

else:

if i - 1 - p != 0:

df.loc[i - 1, &quot;Avg\_El\_iM1&quot;] = s / (i - p)

L1.append(format(s / (i - p), &quot;.3f&quot;))

s = 0

df.loc[i - 1, &quot;Avg\_El\_iM2&quot;] = s1 / (i - p)

L4.append(format(s1 / (i - p), &quot;.3f&quot;))

s1 = 0

p = i

M1.append(H)

H -= 5

j = j - 5

i = i + 1

L1.append(format(s / (i - p), &quot;.3f&quot;))

L4.append(format(s1 / (i - p), &quot;.3f&quot;))

31 | Page

M1.append(H)

p = [None] \* len(L)

d = pd.DataFrame()

d[&quot;El&quot;] = M + M1

d[&quot;UPM1&quot;] = L + p

d[&quot;DNM1&quot;] = p + L1

d[&quot;&quot;] = [None] \* (len(M) \* 2)

d[&quot;El2&quot;] = M + M1

d[&quot;UPM2&quot;] = L3 + p

d[&quot;DNM2&quot;] = p + L4

# Specify the directory where you want to save the output file

output\_directory = os.path.dirname(file\_path)

# Combine the output directory and the filename

b1 = os.path.join(output\_directory, &quot;OutPut&quot; + y[:len(y) - 3] + &quot;xlsx&quot;)

d.to\_excel(b1, index=False)

# Read data from the Excel file

df = pd.read\_excel(b1)

# Create a new Excel writer object

writer = pd.ExcelWriter(b1, engine=&#39;xlsxwriter&#39;)

# Write the DataFrame to the Excel file

32 | Page

df.to\_excel(writer, sheet\_name=&#39;Sheet1&#39;, index=False)

# Get the xlsxwriter workbook and worksheet objects

workbook = writer.book

worksheet = writer.sheets[&#39;Sheet1&#39;]

# Create a chart object

chart = workbook.add\_chart({&#39;type&#39;: &#39;line&#39;})

chart1 = workbook.add\_chart({&#39;type&#39;: &#39;line&#39;})

# Define the data ranges for all five series

categories = &#39;=Sheet1!$A$2:$A$&#39; + str(len(df) + 1)

values\_upm1 = &#39;=Sheet1!$B$2:$B$&#39; + str(len(df) + 1)

values\_dnm1 = &#39;=Sheet1!$C$2:$C$&#39; + str(len(df) + 1)

values\_dnm2 = &#39;=Sheet1!$E$2:$E$&#39; + str(len(df) + 1)

values\_upm3 = &#39;=Sheet1!$F$2:$F$&#39; + str(len(df) + 1)

values\_dnm3 = &#39;=Sheet1!$G$2:$G$&#39; + str(len(df) + 1)

# Configure the series data

chart.add\_series({

&#39;name&#39;: &#39;UPM1&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_upm1,

&#39;line&#39;: {&#39;color&#39;: &#39;red&#39;},

})

33 | Page

chart.add\_series({

&#39;name&#39;: &#39;DNM1&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm1,

&#39;line&#39;: {&#39;color&#39;: &#39;blue&#39;},

})

chart.add\_series({

&#39;name&#39;: &#39;EL&#39;,

&#39;y2\_axis&#39;: True,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm2,

&#39;line&#39;: {&#39;color&#39;: &#39;orange&#39;},

})

chart1.add\_series({

&#39;name&#39;: &#39;EL&#39;,

&#39;y2\_axis&#39;: True,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm2,

&#39;line&#39;: {&#39;color&#39;: &#39;orange&#39;},

})

chart1.add\_series({

34 | Page

&#39;name&#39;: &#39;M2UP&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_upm3,

&#39;line&#39;: {&#39;color&#39;: &#39;purple&#39;},

})

chart1.add\_series({

&#39;name&#39;: &#39;M2DN&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm3,

&#39;line&#39;: {&#39;color&#39;: &#39;yellow&#39;},

})

# Set chart title and axis labels

chart.set\_title({&#39;name&#39;: &#39;2D Line Chart&#39;})

chart.set\_x\_axis({&#39;name&#39;: &#39;EL&#39;})

chart.set\_y\_axis({&#39;name&#39;: &#39;Values&#39;})

chart.set\_y2\_axis({&#39;name&#39;: &#39;Degrees&#39;})

chart1.set\_title({&#39;name&#39;: &#39;2D Line Chart&#39;})

chart1.set\_x\_axis({&#39;name&#39;: &#39;EL&#39;})

chart1.set\_y\_axis({&#39;name&#39;: &#39;Values&#39;})

chart1.set\_y2\_axis({&#39;name&#39;: &#39;Degrees&#39;})

# Insert the chart into the worksheet

35 | Page

worksheet.insert\_chart(&#39;J5&#39;, chart)

worksheet.insert\_chart(&#39;J22&#39;, chart1)

# Close the Excel writer (this saves the file)

writer.close()

# Create a function to select any file

def select\_file():

file\_path = filedialog.askopenfilename(

title=&quot;Select File&quot;,

filetypes=[(&quot;All Files&quot;, &quot;\*.\*&quot;)], # Changed filetypes to allow all files

initialdir=os.path.expanduser(&quot;~&quot;)

)

if file\_path:

process\_dot\_file(file\_path)

else:

print(&quot;No file selected.&quot;)

# Create a tkinter root window

root = tk.Tk()

root.title(&quot;Select File&quot;)

# Function to display M1 values and graph

def display\_m1():

36 | Page

# Add code here to display M1 values and graph in the GUI

pass

# Function to display M2 values and graph

def display\_m2():

# Add code here to display M2 values and graph in the GUI

pass

# Create buttons for M1 and M2

m1\_button = tk.Button(root, text=&quot;M1&quot;, command=display\_m1)

m2\_button = tk.Button(root, text=&quot;M2&quot;, command=display\_m2)

# Create a button to select the DOT file

select\_button = tk.Button(root, text=&quot;Select DOT File&quot;, command=select\_file)

# Pack buttons

select\_button.pack(pady=10)

m1\_button.pack(pady=10)

m2\_button.pack(pady=10)

root.mainloop()

Output:

37 | Page

Here we have select and give DOT file as the input file.

Here we are selecting input DOT file.

38 | Page

Here we can observe output file is created.

The Output that we got in excel.

import pandas as pd

import math

from openpyxl.chart import LineChart

from openpyxl import Workbook

39 | Page

import os

import tkinter as tk

from tkinter import filedialog

import shutil

# Define your existing code as a function

def process\_dot\_file(file\_path):

# Your existing code here

x = os.path.dirname(file\_path)

y = os.path.basename(file\_path)

a = &quot;out.text&quot;

shutil.copy(file\_path, a)

a1 = &quot;output1.csv&quot;

with open(a, &#39;r&#39;) as b1:

lines = [line.strip().split() for line in b1]

csv\_lines = [&#39;,&#39;.join(line) for line in lines]

with open(a1, &#39;w&#39;) as h1:

h1.write(&#39;\n&#39;.join(csv\_lines))

os.remove(a)

df = pd.read\_csv(a1)

os.remove(a1)

z = df.columns.tolist()

i = 0

40 | Page

s = 0

s1 = 0

p = 0

L = []

L1 = []

L3 = []

L4 = []

M = []

M1 = []

j = math.floor(df.loc[0, &quot;ElActual&quot;]) + 5

H = 5

def G(M1):

l = 0

a = len(M1)

while(True):

k = (l + a) // 2

if M1[k] &lt; M1[k+1]:

l = k + 1

else:

a = k

if a == l:

break

return k

if(&quot;Avg\_El\_iM1&quot; not in z):

k = z.index(&quot;El\_iM1&quot;)

41 | Page

df.insert(k + 1, &quot;Avg\_El\_iM1&quot;, None)

df.insert(k + 3, &quot;Avg\_El\_iM2&quot;, None)

kum = G(df[&quot;ElActual&quot;])

while(i &lt;= kum):

if(df.loc[i, &quot;ElActual&quot;] &lt; j):

s = s + df.loc[i, &quot;El\_iM1&quot;]

s1 = s1 + df.loc[i, &quot;El\_iM2&quot;]

else:

if i - p != 0:

df.loc[i - 1, &quot;Avg\_El\_iM1&quot;] = s / (i - p)

L.append(format(s / (i - p), &quot;.3f&quot;))

s = 0

df.loc[i - 1, &quot;Avg\_El\_iM2&quot;] = s1 / (i - p)

L3.append(format(s1 / (i - p), &quot;.3f&quot;))

s1 = 0

p = i

M.append(H)

H = H + 5

j = j + 5

i = i + 1

L.append(format(s / (i - p), &quot;.3f&quot;))

L3.append(format(s1 / (i - p), &quot;.3f&quot;))

M.append(H)

i = kum + 1

p = i

42 | Page

j = math.floor(df.loc[i, &quot;ElActual&quot;]) - 5

s = 0

s1 = 0

while(i &lt; len(df[&quot;ElActual&quot;])):

if(df.loc[i, &quot;ElActual&quot;] &gt; j):

s = s + df.loc[i, &quot;El\_iM1&quot;]

s1 = s + df.loc[i, &quot;El\_iM2&quot;]

else:

if i - 1 - p != 0:

df.loc[i - 1, &quot;Avg\_El\_iM1&quot;] = s / (i - p)

L1.append(format(s / (i - p), &quot;.3f&quot;))

s = 0

df.loc[i - 1, &quot;Avg\_El\_iM2&quot;] = s1 / (i - p)

L4.append(format(s1 / (i - p), &quot;.3f&quot;))

s1 = 0

p = i

M1.append(H)

H -= 5

j = j - 5

i = i + 1

L1.append(format(s / (i - p), &quot;.3f&quot;))

L4.append(format(s1 / (i - p), &quot;.3f&quot;))

M1.append(H)

p = [None] \* len(L)

d = pd.DataFrame()

43 | Page

d[&quot;El&quot;] = M + M1

d[&quot;UPM1&quot;] = L + p

d[&quot;DNM1&quot;] = p + L1

d[&quot;&quot;] = [None] \* (len(M) \* 2)

d[&quot;El2&quot;] = M + M1

d[&quot;UPM2&quot;] = L3 + p

d[&quot;DNM2&quot;] = p + L4

# Specify the directory where you want to save the output file

output\_directory = os.path.dirname(file\_path)

# Combine the output directory and the filename

b1 = os.path.join(output\_directory, &quot;OutPut&quot; + y[:len(y) - 3] + &quot;xlsx&quot;)

d.to\_excel(b1, index=False)

# Read data from the Excel file

df = pd.read\_excel(b1)

# Create a new Excel writer object

writer = pd.ExcelWriter(b1, engine=&#39;xlsxwriter&#39;)

# Write the DataFrame to the Excel file

df.to\_excel(writer, sheet\_name=&#39;Sheet1&#39;, index=False)

# Get the xlsxwriter workbook and worksheet objects

44 | Page

workbook = writer.book

worksheet = writer.sheets[&#39;Sheet1&#39;]

# Create a chart object

chart = workbook.add\_chart({&#39;type&#39;: &#39;line&#39;})

chart1 = workbook.add\_chart({&#39;type&#39;: &#39;line&#39;})

# Define the data ranges for all five series

categories = &#39;=Sheet1!$A$2:$A$&#39; + str(len(df) + 1)

values\_upm1 = &#39;=Sheet1!$B$2:$B$&#39; + str(len(df) + 1)

values\_dnm1 = &#39;=Sheet1!$C$2:$C$&#39; + str(len(df) + 1)

values\_dnm2 = &#39;=Sheet1!$E$2:$E$&#39; + str(len(df) + 1)

values\_upm3 = &#39;=Sheet1!$F$2:$F$&#39; + str(len(df) + 1)

values\_dnm3 = &#39;=Sheet1!$G$2:$G$&#39; + str(len(df) + 1)

# Configure the series data

chart.add\_series({

&#39;name&#39;: &#39;UPM1&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_upm1,

&#39;line&#39;: {&#39;color&#39;: &#39;red&#39;},

})

chart.add\_series({

&#39;name&#39;: &#39;DNM1&#39;,

45 | Page

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm1,

&#39;line&#39;: {&#39;color&#39;: &#39;blue&#39;},

})

chart.add\_series({

&#39;name&#39;: &#39;EL&#39;,

&#39;y2\_axis&#39;: True,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm2,

&#39;line&#39;: {&#39;color&#39;: &#39;orange&#39;},

})

chart1.add\_series({

&#39;name&#39;: &#39;EL&#39;,

&#39;y2\_axis&#39;: True,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm2,

&#39;line&#39;: {&#39;color&#39;: &#39;orange&#39;},

})

chart1.add\_series({

&#39;name&#39;: &#39;M2UP&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_upm3,

46 | Page

&#39;line&#39;: {&#39;color&#39;: &#39;purple&#39;},

})

chart1.add\_series({

&#39;name&#39;: &#39;M2DN&#39;,

&#39;categories&#39;: categories,

&#39;values&#39;: values\_dnm3,

&#39;line&#39;: {&#39;color&#39;: &#39;yellow&#39;},

})

# Set chart title and axis labels

chart.set\_title({&#39;name&#39;: &#39;2D Line Chart&#39;})

chart.set\_x\_axis({&#39;name&#39;: &#39;EL&#39;})

chart.set\_y\_axis({&#39;name&#39;: &#39;Values&#39;})

chart.set\_y2\_axis({&#39;name&#39;: &#39;Degrees&#39;})

chart1.set\_title({&#39;name&#39;: &#39;2D Line Chart&#39;})

chart1.set\_x\_axis({&#39;name&#39;: &#39;EL&#39;})

chart1.set\_y\_axis({&#39;name&#39;: &#39;Values&#39;})

chart1.set\_y2\_axis({&#39;name&#39;: &#39;Degrees&#39;})

# Insert the chart into the worksheet

worksheet.insert\_chart(&#39;J5&#39;, chart)

worksheet.insert\_chart(&#39;J22&#39;, chart1)

47 | Page

# Close the Excel writer (this saves the file)

writer.close()

# Create a function to select any file

def select\_file():

file\_path = filedialog.askopenfilename(

title=&quot;Select File&quot;,

filetypes=[(&quot;All Files&quot;, &quot;\*.\*&quot;)], # Changed filetypes to allow all files

initialdir=os.path.expanduser(&quot;~&quot;)

)

if file\_path:

process\_dot\_file(file\_path)

else:

print(&quot;No file selected.&quot;)

# Create a tkinter root window

root = tk.Tk()

root.title(&quot;Select File&quot;)

# Function to display M1 values and graph

def display\_m1():

# Add code here to display M1 values and graph in the GUI

pass

48 | Page

# Function to display M2 values and graph

def display\_m2():

# Add code here to display M2 values and graph in the GUI

pass

# Create buttons for M1 and M2

m1\_button = tk.Button(root, text=&quot;M1&quot;, command=display\_m1)

m2\_button = tk.Button(root, text=&quot;M2&quot;, command=display\_m2)

# Create a button to select the DOT file

select\_button = tk.Button(root, text=&quot;Select DOT File&quot;, command=select\_file)

# Pack buttons

select\_button.pack(pady=10)

m1\_button.pack(pady=10)

m2\_button.pack(pady=10)

root.mainloop()