DATE	**18 NOVEMBER 2022**
TEAM ID	**PNT2022TMID17268**
PROJECT NAME	**PROJECT-Gas Leakage Monitoring and Alerting System**

Source Coding:

```
import time
import sys
import ibmiotf.application # IBM IoT Watson Platform Module
import ibmiotf.device
import tkinter as tk # Python GUI Package
from tkinter import ttk # Python GUI
import time
from threading import Thread


#Provide your IBM Watson Device Credentials
organization = "23g01e"
deviceType = "madhu"
deviceId = "py33"
authMethod = "token"
authToken = "madhudhava"


# Tkinter root window
root = tk.Tk()
root.geometry('350x300')  # Set size of root window
root.resizable(False, False) # root window non-resizable
root.title('Gas Leakage Monitoring And Alerting System for Industries (PNT2022TMID49514)')


# Layout Configurations
```

```
root.columnconfigure(0, weight=1)
root.columnconfigure(1, weight=3)


# Temperature and Humidity sliders initialization
current_gas = tk.DoubleVar()
current_temp = tk.DoubleVar()



# slider - temperature and humidity functions
def get_current_gas(): # function returns current gas level value
    return '{: .2f}'.format(current_gas.get())

def get_current_temp(): # function returns current temperature value
    return '{: .2f}'.format(current_temp.get())




def slider_changed(event): # Event Handler for changes in sliders
    print('-------')
    print('Gas Level: {: .2f} , Temperature: {: .2f} '.format(current_gas.get(),current_temp.get()))
    print('-------')
    gas_label.configure(text=str(get_current_gas()) +"  ppm")    # Displays current gas level as label
content
    temp_label.configure(text=str(get_current_temp()) +" °C") # Displays current temperature as label
content



# Tkinter Labels

# label for the gas level slider
slider_gas_label = ttk.Label(root,text='Set Gas Level:')
slider_gas_label.grid(column=0,row=0,sticky='w')
```

```
#  Gas Level slider
slider_gas = ttk.Scale(root,from_=200,to=2000,orient='horizontal',
command=slider_changed,variable=current_gas)
slider_gas.grid(column=1,row=0,sticky='we')


# current gas level label
current_gas_label = ttk.Label(root,text='Current Gas Level:')
current_gas_label.grid(row=1,columnspan=2,sticky='n',ipadx=10,ipady=10)


# Gas level label (value gets displayed here)
gas_label = ttk.Label(root,text=str(get_current_gas()) +" ppm")
gas_label.grid(row=2,columnspan=2,sticky='n')


# label for the temperature slider
slider_temp_label = ttk.Label(root,text='Set Temperature:')
slider_temp_label.grid(column=0,row=12,sticky='w')


#  temperature slider
slider_temp = ttk.Scale(root,from_=0,to=100,orient='horizontal',
command=slider_changed,variable=current_temp)
slider_temp.grid(column=1,row=12,sticky='we')


# current temperature label
current_temp_label = ttk.Label(root,text='Current Temperature:')
current_temp_label.grid(row=16,columnspan=2,sticky='n',ipadx=10,ipady=10)


# temperature label (value gets displayed here)
temp_label = ttk.Label(root,text=str(get_current_temp()) +" °C")
temp_label.grid(row=17,columnspan=2,sticky='n')
```

```
def publisher_thread():
    thread = Thread(target=publish_data)
    thread.start()

def publish_data():
    # Exception Handling
    try:
        deviceOptions = {"org": organization, "type": deviceType, "id": deviceId, "auth-method":
authMethod,
                         "auth-token": authToken}
        deviceCli = ibmiotf.device.Client(deviceOptions)
    # .....................................

    except Exception as e:
        print("Caught exception connecting device: %s" % str(e))
        sys.exit()

    deviceCli.connect()  # Connect to IBM Watson IoT Platform

    while True:
        temp = int(current_temp.get())
        gas_level = int(current_gas.get())

        # Send Temperature & Humidity to IBM Watson IoT Platform
        data = {'gas_level' : gas_level, 'temperature': temp, }

        def myOnPublishCallback():
            print("Published Gas Level = %s ppm" % gas_level, "Temperature = %s C" % temp, "to IBM
Watson")

        success = deviceCli.publishEvent("event", "json", data, qos=0,
on_publish=myOnPublishCallback)
        if not success:
```

```
            print("Not connected to IoTF")
        time.sleep(1)

publisher_thread()

root.mainloop() # startup Tkinter GUI

# Disconnect the device and application from the cloud
deviceCli.disconnect()
```