```
from tkinter import *
from tkinter import ttk
from tkinter import filedialog
from tkinter import messagebox
from  ttkthemes import ThemedStyle
import serial
import serial.tools.list_ports
import datetime
import threading
import multiprocessing
import os
import csv

#for printing debugging messages in console
dbg = 0

gRoot = Tk()
gRoot.config(bg="white")
gRoot.geometry("1080x640")
gRoot.title("Black Box Connect")

# sty = ttk.Style()
# sty.theme_use("clam")
sty = ThemedStyle(gRoot)
sty.set_theme('radiance')


gRoot.columnconfigure(0,weight=1)
gRoot.rowconfigure(0,weight=1)
#sty.configure("gframe.TFrame",background="white")
gFrame = ttk.LabelFrame(gRoot,text="Connection Setting",padding=10, style='TFrame')
gFrame.grid(column=1,row=1, sticky=(W,E))

# Frame for commands

gFrameCmd = ttk.LabelFrame(gRoot,text="List of commands",padding=10, width=130 , style
='TFrame')
gFrameCmd.grid(column=1,row=3, sticky=(N, S, E, W))


#Frame for COM messages

gFrame21 = ttk.Frame(gRoot,padding=10, style='TFrame')
gFrame21.grid(column=1,row=2, sticky=(W, E, N))
gRoot.resizable(0,0)


for x in range(10):
    gFrame.columnconfigure(x,weight = x)
    gFrame.rowconfigure(x,weight = x)

label1=ttk.Label(gFrame, text = "Serial Console")
label1.grid(column=2,row=0)
gFrame.rowconfigure(0,weight=2)

sty.configure("label2.TLabel",borderwidth=4,relief="ridge",foreground="red",ipadx=10)
label2=ttk.Label(gFrame,sty="label2.TLabel", text = "Select Com Port")
label2.grid(column=1,row=1, sticky = (N,E,W,S))

"""
Com Port List
"""
#Start
ports = serial.tools.list_ports.comports()
com_port_list = [com[0] for com in ports]
com_port_list.insert(0,"Select an Option")
if dbg == 1:
    print(com_port_list)
#END
com_value_inside = StringVar()
baud_value_inside = StringVar()
baud_menu = ttk.OptionMenu(gFrame,baud_value_inside,"select baud rate","9600",
```

```
                            '19200','28800','38400','57600','76800')
baud_menu.grid(column=3, row=1, sticky = (E))
def com_port_list_update():
    global ports
    global com_port_list
    ports = serial.tools.list_ports.comports()
    com_port_list = [com[0] for com in ports]
    com_port_list.insert(0,"Select an Option")
    if dbg == 1:
        print(com_port_list)
    com_menu = ttk.OptionMenu(gFrame,com_value_inside,*com_port_list)
    com_menu.grid(column=2, row=1, sticky = (E))
    #Frame for the COM LIST
    gRoot_com_list = Toplevel(gRoot)
    x = gRoot.winfo_x()
    y = gRoot.winfo_y()
    gRoot_com_list.geometry("+%d+%d" %(x+200,y+200))
    gFrame01 = ttk.Frame(gRoot_com_list,padding=10)
    gFrame01.grid(column=0,row=1, sticky=(W))
    #Create a horizontal scrollbar
    scrollbar = ttk.Scrollbar(gFrame01, orient= 'horizontal')
    scrollbar.grid(column=1,row=2, sticky=W+E)

    Lb1 = Listbox(gFrame01, xscrollcommand = 1, width = 50, font= ('Helvetica 8 bold'
    ))
    counter = 0;
    for x in ports:
        Lb1.insert(counter, str(x))
    #print (counter)
    counter += 1
    Lb1.grid(column=1,row=1, sticky=W+E)
    Lb1.config(xscrollcommand= scrollbar.set)

    #Configure the scrollbar
    scrollbar.config(command= Lb1.xview)


def serial_print():
    global serFlag
    global ser
    global counter1
    x =""
    #print("Task 1 assigned to thread: {}".format(threading.current_thread().name))
    #print("ID of process running task 1: {}".format(os.getpid()))
    if(serFlag):
        if(ser.in_waiting>0):
            #
            try:
                #x = ser.read(ser.in_waiting)
                x = ser.readline(ser.in_waiting)
                #x = ser.read_until(expected='\n', size=ser.in_waiting)
                #print(x)
                y = str(x.decode())
                Lb2.insert(counter1, str(y))
                Lb2.see("end")
                #print (counter1)
                counter1 += 1
                #gFrame.after(100,serial_print)
            except:
                pass
        ser.flush()
        gFrame.after(100,serial_print)

def CONFIG():
    ser.write("CONFIG".encode())

def write_intg(intg : str):
    CINTG = ("INTG:" + intg).replace("\n","\r")
    ser.write(CINTG.encode())

def write_inti(inti : str):
    CINTI = ("INTI:" + inti).replace("\n","\r")
```

```
143         ser.write(CINTI.encode())
144 
145     def write_toff(toff : str):
146         CTOFF = ("TOFF:" + toff).replace("\n","\r")
147         ser.write(CTOFF.encode())
148 
149     def write_leds(leds : str):
150         if (leds == "ON"):
151             CLEDS = "LEDV:1\r"
152         elif (leds == "OFF"):
153             CLEDS = "LEDV:0\r"
154         else:
155             CLEDS = "LEDV:1\r"
156 
157         ser.write(CLEDS.encode())
158 
159     def write_configs(tgnss : str, timu : str, toff: str, ledst : str):
160 
161         CINTG = ("INTG:" + tgnss).replace("\n","\r")
162         ser.write(CINTG.encode())
163         CINTI = ("INTI:" + timu).replace("\n","\r")
164         ser.write(CINTI.encode())
165         CTOFF = ("TOFF:" + toff).replace("\n","\r")
166         ser.write(CTOFF.encode())
167 
168     def EXIT():
169         ser.write("EXIT".encode())
170 
171     def config_mode():
172         filewin2 = Toplevel(gRoot)
173         filewin2.geometry("450x210")
174 
175         Label(filewin2, text = "GNSS measure interval : ", anchor='w').grid(column=1, row
            = 1)
176         txt_tgnss = Text(filewin2, height=1, width=10)
177         txt_tgnss.grid(column=2, row = 1)
178         txt_tgnss.insert(END, "5000")
179         button_intg = ttk.Button(filewin2, text="Send", command=lambda:[write_intg(
            txt_tgnss.get(1.0, END))]).grid(column=3, row = 1)
180 
181         Label(filewin2, text = "IMU measure interval : ", anchor='w').grid(column=1, row =
             2)
182         txt_timu = Text(filewin2, height=1, width=10)
183         txt_timu.grid(column=2, row = 2)
184         txt_timu.insert(END, "500")
185         button_inti = ttk.Button(filewin2, text="Send", command=lambda:[write_inti(
            txt_timu.get(1.0, END))]).grid(column=3, row = 2)
186 
187         Label(filewin2, text = "Inactive delay : ",anchor='w').grid(column=1, row = 3)
188         txt_toff = Text(filewin2, height=1, width=10)
189         txt_toff.grid(column=2, row = 3)
190         txt_toff.insert(END, "60")
191         button_toff = ttk.Button(filewin2, text="Send", command=lambda:[write_toff(
            txt_toff.get(1.0, END))]).grid(column=3, row = 3)
192 
193         ledList = StringVar(filewin2)
194         ledList.set("txt")
195         Label(filewin2, text = "Etat LED de Vie : ",anchor='w').grid(column=1, row = 4)
196         led_menu = ttk.OptionMenu(filewin2,  ledList, "ON", "ON", "OFF")
197         led_menu.config(width=5)
198         led_menu.grid(column=2, row = 4)
199         button_toff = ttk.Button(filewin2, text="Send", command=lambda:[write_leds(ledList
            .get())]).grid(column=3, row = 4)
200 
201 
202         Label(filewin2, text = "").grid(column=1, row = 5)
203         button2 = ttk.Button(filewin2, text="Exit", command=lambda:[EXIT(), filewin2.
            destroy()], underline=TRUE).grid(column=1, row = 7, columnspan=2)
204         #filewin2.protocol("WM_DELETE_WINDOW", EXIT())
205 
206     ser = serial.Serial()
207     serFlag = 0
```

```
208  def serial_connect(com_port,baud_rate):
209      global ser
210      ser.baudrate = baud_rate
211      ser.port = com_port
212      ser.timeout = 1
213      ser._xonxoff=1
214      ser.bytesize=serial.EIGHTBITS
215      ser.parity=serial.PARITY_NONE
216      ser.stopbits=serial.STOPBITS_ONE
217      ser.open()
218      global serFlag
219      serFlag = 1
220
221      t1 = threading.Thread(target = serial_print, args = (), daemon=1)
222      t1.start()
223      #t1.join()
224      """
225      P1 = multiprocessing.Process(target = serial_print, args=())
226      P1.start()
227      P1.join()
228      """
229      #serial_print()
230  counter1 = 0;
231
232  def SHUTDOWN():
233      ser.write("SHUTDOWN".encode())
234  def GCLR():
235      ser.write("GCLR".encode())
236  def ICLR():
237      ser.write("ICLR".encode())
238  def GLIVE():
239      ser.write("GLIVE".encode())
240  def ILIVE():
241      ser.write("ILIVE".encode())
242  def GLOG():
243      ser.write("GLOG".encode())
244  def ILOG():
245      ser.write("ILOG".encode())
246
247  def serial_close():
248      global ser
249      global serFlag
250      serFlag = 0
251      ser.close()
252
253  def power_off():
254      global ser
255      global serFlag
256      serFlag = 0
257      ser.close()
258      if messagebox.askokcancel("Quit", "Do you want to quit?"):
259          gRoot.destroy()
260
261
262  def submit_value():
263      if dbg == 1:
264          print("selected option: {}".format(com_value_inside.get()))
265          print(" Baud Rate {}".format(baud_value_inside.get()))
266      serial_connect(com_value_inside.get(),baud_value_inside.get())
267
268
269  Lb2 = Listbox(gFrame21, width = 130, height=20, xscrollcommand = 1)
270  Lb2.grid(column=1, row = 1, sticky = W+E)
271  Sb2 = ttk.Scrollbar(gFrame21,orient = 'vertical')
272  Sb2.config(command=Lb2.yview)
273  Sb2.grid(column = 2,row =1, sticky=N+S)
274  Sb2v = ttk.Scrollbar(gFrame21,orient = 'horizontal')
275  Sb2v.grid(column = 1,row =2, sticky=W+E)
276  Sb2v.config(command = Lb2.xview)
277  Lb2.configure(xscrollcommand = Sb2v.set, yscrollcommand = Sb2.set)
278
279  def clear_listbox():
```

```
280         Lb2.delete(0,END)
281
282
283
284
285     subBtn = ttk.Button(gFrameCmd,text="GNSS live data",command = GLIVE, width=15)
286     subBtn.grid(column=1,row=1, sticky = (E))
287     subBtn = ttk.Button(gFrameCmd,text="IMU live data",command = ILIVE, width=15)
288     subBtn.grid(column=1,row=2, sticky = (E))
289
290     subBtn = ttk.Button(gFrameCmd,text="Get GNSS logs",command = GLOG, width=20)
291     subBtn.grid(column=2,row=1, sticky = (E))
292     subBtn = ttk.Button(gFrameCmd,text="Get IMU logs",command = ILOG, width=20)
293     subBtn.grid(column=2,row=2, sticky = (E))
294
295     subBtn = ttk.Button(gFrameCmd,text="Delete GNSS logs",command = GCLR, width=20)
296     subBtn.grid(column=3,row=1, sticky = (E))
297     subBtn = ttk.Button(gFrameCmd,text="Delete IMU logs",command = ICLR, width=20)
298     subBtn.grid(column=3,row=2, rowspan=2, sticky = (E))
299
300     subBtn = ttk.Button(gFrameCmd,text="Configurate BlackBox",command = lambda:[CONFIG(),
        config_mode()], width=24)
301     subBtn.grid(column=0,row=1, rowspan=2, sticky = (E))
302
303
304     subBtn = ttk.Button(gFrameCmd,text="Exit",command = EXIT, style = "W.TButton",
        underline=TRUE, width=10)
305     subBtn.grid(column=4,row=1, sticky = (E))
306
307     subBtn = ttk.Button(gFrameCmd,text="Shutdown",command = SHUTDOWN, style = "W.TButton",
         underline=TRUE, width=10)
308     subBtn.grid(column=4,row=2, sticky = (E))
309
310
311
312     subBtn = ttk.Button(gFrame,text="submit",command = submit_value)
313     subBtn.grid(column=4,row=1, sticky = (E))
314
315     RefreshBtn = ttk.Button(gFrame,text="Get List",command = com_port_list_update)
316     RefreshBtn.grid(column=2,row=2, sticky = (E))
317
318
319
320     closeBtn = ttk.Button(gFrame,text="Disconnect",command = serial_close)
321     closeBtn.grid(column=4,row=2, sticky = (E))
322
323     clearBtn = ttk.Button(gFrame,text="Clear Messages",command = clear_listbox)
324     clearBtn.grid(column=3,row=2, sticky = (E))
325
326
327
328     """
329     #Add a Listbox Widget
330     listbox = Listbox(win, width= 350, font= ('Helvetica 15 bold'))
331     listbox.pack(side= LEFT, fill= BOTH)
332
333     #Add values to the Listbox
334     for values in range(1,101):
335        listbox.insert(END, values)
336     """
337     def donothing():
338        filewin = Toplevel(gRoot)
339        button = Button(filewin, text="Do nothing button")
340        button.pack()
341
342     def writeFile(fileName : str, extension : str):
343
344         extension = extension.replace("\n", "")
345         fileName = fileName.replace("\n", "")
346         filePath = str("./" + fileName + "." + extension)
347         txtToSave = "".join(str(el) for el in Lb2.get(0, END))
348
```

```
349         if(extension == "csv"):
350             with open(filePath, 'w', newline='', encoding='utf-8') as f:
351                 writer = csv.writer(f)
352                 writer.writerow(Lb2.get(0, END))
353 
354         else:
355             file = open(filePath,"w")
356             file.write(txtToSave)
357 
358 
359 
360     def save():
361         filewin = Toplevel(gRoot)
362         filewin.geometry("200x170")
363         Label(filewin, text = "File name : ").pack()
364 
365         today = datetime.datetime.now()
366 
367         display_text = StringVar()
368         txt_save = Text(filewin, height=1, width=20)
369         txt_save.pack()
370         txt_save.insert(END, "log_"+today.strftime("%m%d%y_%H%M%S"))
371 
372         Label(filewin, text = "Format : ").pack()
373 
374         varList = StringVar(filewin)
375         varList.set("txt")
376         format_menu = ttk.OptionMenu(filewin,  varList, "txt", "txt", "csv", "xls", "docx"
            , "odf")
377         format_menu.config(width=5)
378         format_menu.pack()
379 
380         Label(filewin, text = "").pack()
381 
382         button = ttk.Button(filewin, text="Sauvegarder", command=lambda:[writeFile(
            txt_save.get(1.0, END), varList.get()), filewin.destroy() ], underline=TRUE).pack
            ()
383 
384         #Lb2.text
385 
386     def About_me():
387        filewin = Toplevel(gRoot)
388        Label1 = Label(filewin, text = "https://github.com/Ali-Z0/1924B_MiniBoiteNoire").
           pack()
389        button = Button(filewin, text="Quit", command = filewin.destroy)
390        button.pack()
391 
392     menubar = Menu(gRoot)
393     filemenu = Menu(menubar, tearoff=0)
394     #filemenu.add_command(label="New", command=donothing)
395     #filemenu.add_command(label="Open", command=donothing)
396     filemenu.add_command(label="Save", command=save)
397     #filemenu.add_command(label="Save as...", command=donothing)
398     filemenu.add_command(label="Close", command=power_off)
399 
400     #filemenu.add_separator()
401 
402     #filemenu.add_command(label="Exit", command=gRoot.quit)
403     menubar.add_cascade(label="File", menu=filemenu)
404     editmenu = Menu(menubar, tearoff=0)
405     editmenu.add_command(label="Undo", command=donothing)
406 
407     editmenu.add_separator()
408 
409     """
410     editmenu.add_command(label="Cut", command=donothing)
411     editmenu.add_command(label="Copy", command=donothing)
412     editmenu.add_command(label="Paste", command=donothing)
413     editmenu.add_command(label="Delete", command=donothing)
414     editmenu.add_command(label="Select All", command=donothing)
415     """
416 
```

```
417   #menubar.add_cascade(label="Edit", menu=editmenu)
418   helpmenu = Menu(menubar, tearoff=0)
419   #helpmenu.add_command(label="Help Index", command=donothing)
420   #helpmenu.add_command(label="About...", command=donothing)
421   #menubar.add_cascade(label="Help", menu=helpmenu)
422   menubar.add_command(label = "Black Box", command = About_me)
423   menubar.add_separator()
424   menubar.add_command(label = "Quit", command = power_off)
425
426   gRoot.protocol("WM_DELETE_WINDOW", power_off)
427   gRoot.config(menu=menubar)
428   gRoot.mainloop()
```