```
# -*- coding: utf-8 -*-
##########################################################################################
# File name:        server.py                                            #
# File description: Socket server for EH-Drone control                   #
# Author name:      Giacomo Dollevedo, Gustavo Fernandes                 #
# Creation date:    18nov2020                                            #
# Revision date:    17dec2020                                            #
##########################################################################################

from threading import Thread
import socket
from queue import PriorityQueue
import curses

global svDisp
global svInput
global blank
blank = "                "

cmd_q = PriorityQueue(maxsize = 5)

global droneOnline
droneOnline = 0

global recv_counter
recv_counter = 0

##########################################################################################
#                                                                        #
#   Method Name: receive                                                 #
#                                                                        #
#   Description: This method is used to receive text data from a socket connection, without decoding from bytes to text #
#                                                                        #
#   Input parameters:   conn  -> Socket object that data will be received from #
#                                                                        #
#   Output parameters:  Returns the message in byte format               #
#                                                                        #
##########################################################################################
def receive(conn):
    try:
```

```
41
42          message_len = int(conn.recv(5).decode())   #Receiving first 5 bytes
(incoming message length)
43          return conn.recv(message_len)              #Returning the incoming message
44
45      except socket.error as e:
46          print(e)
47          pass
48
49
50 ##########################################################################################
###################################
51 #                                    #
52 #   Method Name: sendD                #
53 #                                    #
54 #   Description: This method is used to send text data over a socket connection,
enconding it into bytes and            #
55 #                   marking the first 5 bytes with the message length.
                                       #
56 #                                    #
57 #   Input parameters:   conn  -> Socket object that data will be sent to
                                       #
58 #                       data  -> String that will be sent over the connection
                                       #
59 #                                    #
60 #   Output parameters:  n/a          #
61 #                                    #
62 ##########################################################################################
###################################
63 def sendD(conn, data):
64     try:
65         data_b = str.encode(data)
66         data_len = str(len(data_b))
67
68         code = str.encode(data_len.rjust(5, '0'))   #Making a 5 byte sized string
with '0's
69         message = code + data_b                     #Concatenating length and message
70         conn.sendall(message)                       #Sending resulting message
71
72     except socket.error as e:
73         print(e)
74         pass
75
76
77 ##########################################################################################
###################################
78 #                                    #
79 #   Method Name: authESP              #
80 #                                    #
```

```
#   Description: Authentication routine to ensure the client that was connected is the Drone client                    #
#                                                                                                                      #
#   Input parameters:   conn  -> Socket object that data will be sent to                                               #
#                                                                                                                      #
#   Output parameters:  n/a                                                                                            #
#                                                                                                                      #
########################################################################################################################
def authESP(conn):

    global svDisp

    sendD(conn, "#AEHDRONE")

    print("Autenticando...")
    svDisp.addstr(4,0,'Autenticando...')
    svDisp.refresh()
    key = receive(conn).decode()

    if(key == "OK"):
        return 1

    else:
        return 0



########################################################################################################################
#                                                                                                                      #
#   Method Name: threadedHandleDrone                                                                                   #
#                                                                                                                      #
#   Description: Threaded method that will handle all Drone socket communication                                       #
#                                                                                                                      #
#   Input parameters:   conn  -> Socket object that data will be sent to                                               #
#                                                                                                                      #
#   Output parameters:  n/a                                                                                            #
#                                                                                                                      #
########################################################################################################################
def threadedHandleDrone(conn):

    global droneOnline
    global recv_counter
```

```
122     global svDisp
123     global svInput
124     global blank
125 
126     command = "NULL"
127 
128 
129     with conn:
130         while(command != "#G"):
131             try:
132                 command = cmd_q.get()[1]
133 
134             except:
135                 command = "NULL"
136 
137         print("ENVIANDO --->\t" + command)
138         svDisp.addstr(7,0,'ENVIANDO COMANDO --->')
139         svDisp.addstr(7,23, command)
140         svDisp.refresh()
141         svInput.refresh()
142 
143         sendD(conn, command)
144 
145         msg = receive(conn).decode(encoding="utf-8")
146         print("RECEBIDO --->\t" + msg)
147         svDisp.addstr(8,0,'MENSAGEM RECEBIDA --->')
148 
149         svDisp.addstr(8,24, str(msg))
150 
151         svDisp.refresh()
152 
153         svInput.addstr(3,0,"JOYSTICK DESABILITADO",
curses.color_pair(3)|curses.A_BOLD)
154         svInput.addstr(4,0,"\t´#J´ = HABILITAR JOYSTICK", curses.color_pair(1))
155         svInput.addstr(5,0,"\t´#SGaxis;kp;ki;kd´ = SETAR GANHO CONTROLADOR POR EIXO",
curses.color_pair(2))
156         svInput.addstr(6,0,"\t´SVxxxx;xxxx;xxxx;xxxx´ = SETAR THROTTLE",
curses.color_pair(1))
157         svInput.addstr(7,0,"\t´#R´ = RESETAR MOTORES PARA VELOCIDADE BASE",
curses.color_pair(2))
158         svInput.clrtobot()
159         svInput.move(1,15)
160         svInput.refresh()
161 
162 
163         while(droneOnline == 1):
164 
165             if(cmd_q.qsize() == 0):
166                 try:
167                     cmd_q.put((3, "#K"))
168                 except:
169                     pass
170             else:
171                 pass
172 
173             print("AGUARDANDO...")
174             msg = receive(conn).decode(encoding="utf-8")
175             print("RECEBIDO --->\t" + msg)
176             svDisp.addstr(8,0,'MENSAGEM RECEBIDA --->')
```

```
177             splitado = str(msg).split("\n")
178 
179             try:
180                 svDisp.addstr(8,24, splitado[0])
181                 svDisp.addstr(9,24, splitado[1])
182 
183             except:
184                 pass
185 
186             #svDisp.addstr(8,24, str(msg) + blank)
187             svDisp.refresh()
188             svInput.refresh()
189             recv_counter += 1
190 
191             if(recv_counter == 5):
192                 command = cmd_q.get()[1]
193 
194                 # if(command == '#J'):
195                 #     curses.cbreak()
196 
197                 # if(command == 's'):
198                 #     curses.nocbreak()
199 
200                 print("ENVIANDO --->\t" + command)
201                 svDisp.addstr(7,0,'ENVIANDO COMANDO --->')
202                 #svDisp.move(7,23)
203                 #svDisp.clrtoel()
204                 svDisp.addstr(7,23, command + blank)
205                 svDisp.refresh()
206                 svInput.refresh()
207                 sendD(conn, command)
208                 recv_counter = 0
209 
210 
211 
212 
213 
214 ##########################################################################################
    ####################################
215 #                                                                                        #
    #
216 #   Method Name: main                                                                    #
    #
217 #                                                                                        #
    #
218 #   Description: Main method that the script runs. Start up the server and handle
    user input.                                                                            #
219 #                                                                                        #
    #
220 #   Input parameters:   stdscr  -> Curses library object to better handle console
    display                                                                                #
221 #                                                                                        #
    #
222 #   Output parameters:  n/a                                                              #
    #
223 #                                                                                        #
    #
224 ##########################################################################################
    ####################################
```

```
225 def main(stdscr):
226
227     global droneOnline
228     global svDisp
229     global svInput
230     global blank
231
232     #HOST = '192.168.43...'
233     HOST = '192.168.43.182'
234
235
236     PORT = 25565        # Port to listen on (non-privileged ports are > 1023)
237
238     joystick_enabled = 0
239
240 ##%% SERVER CREATION AND CONNECTION SECTION
241
242     #   Statement to ensure this will run only once and will not create infinite
243     # instances recursively (Python error)
244     if __name__ == '__main__':
245                 #0000000000000000000000000000000000000000000000000000
246         curses.nocbreak()
247         svDisp = curses.newwin(20,120,0,0)
248
249
250
251         svInput = curses.newwin(8,120,21,0)
252         curses.start_color()
253         curses.init_pair(1, curses.COLOR_GREEN, curses.COLOR_BLACK)
254         curses.init_pair(2, curses.COLOR_YELLOW, curses.COLOR_BLACK)
255         curses.init_pair(3, curses.COLOR_RED, curses.COLOR_BLACK)
256         curses.init_pair(4, curses.COLOR_GREEN, curses.COLOR_WHITE)
257
258         curses.echo()
259         svInput.keypad(True)
260
261         svInput.addstr(0,0, "------------------------------------------------------------
------------------------------------------------------------")
262         svInput.addstr(1,0, "User Input >>")
263         svInput.addstr(2,0, "------------------------------------------------------------
------------------------------------------------------------")
264         svInput.move(1,15)
265         svInput.refresh()
266
267         # Creating a socket object
268         # AF_INET     -> IPv4 address
269         # SOCK_STREAM -> TCP Connection
270         s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
271
272
273         # Binding socket object to a specific host (IP) and network port
274         try:
275             s.bind((HOST, PORT))
276         except socket.error as e:
277             print(e)
278
279
280         # Starting the server by listening to incoming connections
281         s.listen()
```

```
        print('------SERVER STARTED------')
        svDisp.addstr(0,0,'------SERVER STARTED------')
        svDisp.refresh()
        
        
        
        
        
        while True:
            # When connection is found, store its socket object (conn) and address
            print('Listening to connections...\n')
            svDisp.addstr(2,0,'Listening to connections...')
            svDisp.refresh()
            try:
                conn, addr = s.accept()
                print(conn)
                print(addr)
                svDisp.addstr(3,0,"Cliente Conectado: " + str(addr))
                svDisp.refresh()

                if(authESP(conn)):
                    print('Autenticado!')
                    svDisp.addstr(5,0,'Autenticado!')
                    svDisp.refresh()
                    svInput.refresh()
                    client_thread = Thread(target=threadedHandleDrone, args=(conn,))
                    client_thread.daemon = True
                    client_thread.start()
                    droneOnline = 1

                    while(droneOnline == 1):

                        if(joystick_enabled == 0):
                            command = svInput.getstr().decode(encoding="utf-8")

                        else:
                            command = svInput.getkey()

                            if(command == 'PADPLUS'):
                                command = '+'

                            elif(command == 'PADMINUS'):
                                command = '-'



                        cmd_q.put((1, command))

                        if(command == '#J'):
                            joystick_enabled = 1
                            curses.cbreak()
                            svInput.move(3,0)
                            svInput.clrtobot()
                            svInput.addstr(3,0,"JOYSTICK HABILITADO",
curses.color_pair(1)|curses.A_BOLD)
                            svInput.addstr(4,0,"\t´8´ = -PITCH RATE\t ´4´ = -ROLL
RATE\t ´-´ = -THROTTLE", curses.color_pair(1))
                            svInput.addstr(5,0,"\t´2´ = +PITCH RATE\t ´6´ = +ROLL
RATE\t ´+´ = +THROTTLE", curses.color_pair(1))
```

```
338                             svInput.addstr(6,0,"\t´5´ = SETAR ROLL E PITCH PARA 0",
curses.color_pair(2))
339                             svInput.addstr(7,0,"\t´s´ = DESABILITAR JOYSTICK",
curses.color_pair(3))
340                             svInput.addstr(3,99, "8",
curses.color_pair(1)|curses.A_BOLD)
341                             svInput.addstr(4,99, "|",
curses.color_pair(1)|curses.A_BOLD)
342                             svInput.addstr(5,95,  "4---5---6",
curses.color_pair(1)|curses.A_BOLD)
343                             svInput.addstr(6,99, "|",
curses.color_pair(1)|curses.A_BOLD)
344                             svInput.addstr(7,99, "2",
curses.color_pair(1)|curses.A_BOLD)
345
346
347
348
349
350                         if(command == 's'):
351                             joystick_enabled = 0
352                             svInput.move(3,0)
353                             svInput.clrtobot()
354                             svInput.addstr(3,0,"JOYSTICK DESABILITADO",
curses.color_pair(3)|curses.A_BOLD)
355                             svInput.addstr(4,0,"\t´#J´ = HABILITAR JOYSTICK",
curses.color_pair(1))
356                             svInput.addstr(5,0,"\t´#SGaxis;kp;ki;kd´ = SETAR GANHO
CONTROLADOR POR EIXO", curses.color_pair(2))
357                             svInput.addstr(6,0,"\t´STxxxx;xxxx;xxxx;xxxx´ = SETAR
THROTTLE", curses.color_pair(1))
358                             svInput.addstr(7,0,"\t´#R´ = RESETAR MOTORES PARA
VELOCIDADE BASE", curses.color_pair(2))
359                             svInput.refresh()
360                             curses.nocbreak()
361
362
363                         svDisp.addstr(10,0, "ULTIMO COMANDO INSERIDO --->" + command
+ blank)
364                         svDisp.refresh()
365
366                         svInput.move(1,15)
367                         svInput.clrtoeol()
368                         svInput.refresh()
369
370
371
372
373                 else:
374                     print('ERRO na Autenticacao!\nEncerrando o programa!')
375                     break
376
377             except (KeyboardInterrupt, SystemExit, socket.error):
378                 s.close()
379                 print('Server Closed')
380
381
382         s.close()
383         print('Server closed')
```

```
384
385         return
386
387
388 curses.wrapper(main)
389
390 main()
391
392
```