PARALLELY CONTROLLING TWO USB PORTS USING PYSERIAL MODULE .

Main motive of programme :- (run1.py and run2.py)

Reading data from usb port which is either sequence of 2 or 3 at any instant of time . And then detecting the change in level i.e either from 2 to 3 or from 3 to2 . At that change in level we have to send stop sequence "**xxxxxxxxxxx**" and after that there is a delay of 8.7 sec (according to user requirement) and then restart "**xxxxxxxxxx**" and then again user required delay of 8.7 sec and then start sequence "**xxxxxxxxxxxxx**" . (these all are done in run1.py and run2.py) .

**** One perticular change in level (say either from 2 to 3 or from 3 to 2) can be also detected . Just by removing (or just comment) a perticular loop .

**** Record_run1.txt and record_run2.txt are just for record purposes , programme is not using it for receive data or send command sequence .

**** Timeout in run1.py and run2.py is different then timeout argument used in serial.Serial() function in serial_port.py . At this timeout if programm won't find any change in level (i.e. 2 to 3 or 3 to 2) then it will exit the loop and then connection will be terminated (so , no command sequence is sent) .

CODE OF RUN1.PY (BOTH RUN1 AND RUN2 ARE ALIKE)

```
import time
import os
import serial_port as sp

if sp.is_connection_established_run1 == False :
```

```
        print(   'serial connetion for run1 is not established ' ) ;


else  :

        print(   'serial connetion for run1 is established with port :' , sp.run1_port_addr ) ;

        f = open("record_run1.txt" , "w") ;



        #message = raw_input("Give Message to be sent : ")
        message1 = "xxxxxxxxxx" ;
        message2 = "xxxxxxxxxx" ;
        message3 = "xxxxxxxxxx" ;



        print("hi i am run1") ;

        timeout = time.time() + 10 ;
        temp = sp.pulse_1.read() ;

        #print( temp ) ;
        f.write(temp ) ;
        while (True) :
                        if time.time() > timeout :
                                break ;


                        a = sp.pulse_1.read() ;


                        if( a =='2' and temp == '3' ):
                                print("is in loop 3 to 2 ");
                                f.write(message1);
                        sp.pulse_1.write(message1) ;
                                f.write(a);
                                print("stop") ;
                                time.sleep(8.7) ;
                                 #print("message");
                                f.write(message2);
                        sp.pulse_1.write(message2) ;
                                #print(a) ;
                                f.write(a);
                                print("restart") ;
                                time.sleep(8.7) ;
                                 #print("message");
                                f.write(message3);
                        sp.pulse_1.write(message3) ;
                                #print(a) ;
```

```
                    f.write(a);
                    print("run") ;
                    break ;
            elif ( a == '3' and temp == '2' ):
                    print("is in loop 2 to 3 ");
                    f.write(message1);
                    sp.pulse_1.write(message1) ;
            #print(a) ;
                    f.write(a);
                    print("stop") ;
                    time.sleep(8.7) ;
                     #print("message");
                    f.write(message2);
            sp.pulse_1.write(message2) ;
                    #print(a) ;
                    f.write(a);
                    print("restart");
                    time.sleep(8.7) ;
                    #print("message");
                    f.write(message3);
            sp.pulse_1.write(message3) ;
                    #print(a) ;
                    f.write(a);
                    print("start") ;
                    break ;

        else :
                    f.write(a);
                    #print(a) ;
            temp = a ;



shut_down = sp.pulse_1.close() ;

if shut_down < 0 :
        print(' connection of run1 is terminated  ')
else :
        print('connection of run1 is still alive ')
```

Pyserial :- (code for establishing (hand shaking) connection is in serial_port.py)

The timeout argument is for preventing the permanent blocking of usb port . Because when we give read command , it won't come back untill it will get 8 bit data (in our case default is 8 bit we can change it to other bit also) . So, if it not getting any data then after timeout (= 11 sec) value the read command will return back . For establishing connection we require usbport address wich is stored in /dev/ directory in linux . And observerd that name of file is ttyUSB1 or ttyUSB2 or ttyUSB3 etc , in order to save users time from searching file name and then manualy putting it in serial.Serial( ) argument , we first read that /dev/ directory and then searching for all ttyUSB* file (total number of ttyUSB* file will be alway same as number of USB connections but that * can be 1 , 2 , 3 etc) . After finding the port , we are reading it into an array and then using for loop giving each element of that array to serial.Serial() function .

***** Totaly random selection for port is happening , so we can't say which port is going to be alloted to which usb file name(ttyUSB1 or ttyUSB2 etc) by the system .

*****So , we can't say that the serial_port.py will assign which port to run1.py and run2.py . So , we can't say that among two usb connected to hardware ports on laptop which one is assign to run1.py and which one is assign to run2.py . (Amit Sir's requirement is that both port is symmetric) .

***** One efficiency problem is ..... while importing serial_port.py in run1.py and run2.py . The connection established two times , which can be avoided by converting serial_port.py as a function and then using that function whenever required .

CODE FOR ESTABLISHING CONNECTION TO USB PORT :

```
import serial
import os


store_port = os.popen( " ls /dev/ttyUSB* " ).read() ;        # in /dev/  directory reading all files name start with ttyUSB
store_port = store_port.split() ;                    # space seprated string elements ( usb file name )  to array of usb name


is_connection_established_run1 = False ;
is_connection_established_run2 = False ;

for itr in range( len(store_port) ) :

        try:
                pulse_1 = serial.Serial(store_port[itr], 115200 ,timeout = 11) ;
```

```
                if pulse_1.isOpen() :
                        is_connection_established_run1 = True ;
                        run1_port_addr = store_port[itr];
                        del store_port[itr] ;       # deleting the usb file name to which connection is
established
                        break ;                     # break from the loop and estab. connection for other
usb in nxt loop

        except serial.SerialException as ex :
                print( store_port[itr],"port is unavailable ") ;


for itr in range( len(store_port) ) :
        try:
                pulse_2 = serial.Serial(store_port[itr], 115200 ,timeout = 11) ;
                if pulse_2.isOpen() :
                        is_connection_established_run2 = True ;
                        run2_port_addr = store_port[itr];
                        del store_port[itr] ;
                        break ;
        except serial.SerialException as ex :
                print( store_port[itr],"port is unavailable ") ;
```

Running both script parallely :- (master.py)

multiprocessing library of python is used . Logic wise for computing two arguments from a function we pass two arguments using mupltiprossesing pool function .

```
def compute( argument ) :
        return compute**compute ;

pool = pool ( 3 )            # number of arguments we want to send say 3
```

pool.map(compute , [1,2,3]) # sending 3 arguments i.e 1 , 2 , 3

output : 1 , 4 , 27

So, if we want to compute (run) two files simultaneously we should pass two file name to the function instead of some constants , but that function won't compute anything but instructs the operating system to run its argument (python file) .

```
def compute( argument.py ) :
        os.system( ' python argument.py' ) ;

pool = pool ( 2 )        # number of argument files we want to send say 2

pool.map( compute , [ file1.py , file2.py ] )
```

MASTER.PY

```
import os
from multiprocessing import Pool


#number of usb a user want to connect having address /dev/ttyUSB* format only
num_usb = 2 ;

processes = ['run1.py' , 'run2.py']


def run_processes(process):
   os.system('python {}'.format(process));


# aloting num_usb labours for running num_usb py scripts
pool = Pool(processes = num_usb);
pool.map(run_processes , processes);
```