```python
# SM.py
import serial
from typing import Optional


class SerialManager:
    def __init__(self):
        self.serial_port: Optional[serial.Serial] = None
        self.ser_data = bytearray(2048)
        self.user_data = bytearray(2048)

    # ---------------- INIT ----------------
    def initSerialPort(self, portName: str) -> int:
        try:
            self.serial_port = serial.Serial(
                port=portName,
                baudrate=9600,
                bytesize=serial.EIGHTBITS,
                parity=serial.PARITY_NONE,
                stopbits=serial.STOPBITS_ONE,
                timeout=0.1
            )
            return 0
        except:
            return 1

    # ---------------- CLOSE ----------------
    def serclose(self):
        if self.serial_port and self.serial_port.is_open:
            self.serial_port.close()

    # ---------------- TX ----------------
    def serTx(self, data: bytes, data_len: int) -> int:
        if not self.serial_port or not self.serial_port.is_open:
            return 0
        try:
            return self.serial_port.write(data[:data_len])
        except:
            return 0

    # ---------------- RX ----------------
    def serRx(self) -> int:
        if not self.serial_port or not self.serial_port.is_open:
            return 0
        try:
            rx = self.serial_port.read(self.serial_port.in_waiting or 1)
            if rx:
                length = len(rx)
                self.user_data[:length] = rx
                self.ser_data[:length] = rx
                return length
        except:
            pass
        return 0


# GUI.py
import tkinter as tk
import threading
import time
from SM import SerialManager

sm = SerialManager()
PORT = "COM3"  # Change if needed

if sm.initSerialPort(PORT) != 0:
    print("Port not available")

running = True

# ---------------- READ THREAD ----------------
def read_serial():
    while running:
```

```
        length = sm.serRx()
        if length > 0:
            data = sm.ser_data[:length].hex().upper()
            log_text.insert(tk.END, f"RX: {data}\n")
            log_text.see(tk.END)
        time.sleep(0.05)

threading.Thread(target=read_serial, daemon=True).start()

# ---------------- SEND ----------------
def send_bytes():
    data = entry.get().strip()
    if data:
        try:
            b = bytes.fromhex(data)
            sm.serTx(b, len(b))
            log_text.insert(tk.END, f"TX: {data.upper()}\n")
            log_text.see(tk.END)
            entry.delete(0, tk.END)
        except Exception as e:
            log_text.insert(tk.END, f"Error: {str(e)}\n")

# ---------------- CLOSE ----------------
def close_app():
    global running
    running = False
    sm.serclose()
    root.destroy()

# ---------------- GUI ----------------
root = tk.Tk()
root.title("Serial Send & Read")
root.geometry("700x500")
root.protocol("WM_DELETE_WINDOW", close_app)

tk.Label(root, text="Send Hex (space separated):").pack(pady=5)

entry = tk.Entry(root, width=60)
entry.pack(pady=5)

tk.Button(root, text="Send", command=send_bytes).pack(pady=5)

log_text = tk.Text(root, height=20)
log_text.pack(fill=tk.BOTH, expand=True, padx=10, pady=10)

root.mainloop()
```