MONITORX

1. Overview:

The program is a Serial Monitor that communicates with a device (like Arduino) connected to a serial port. It provides an interface to configure and monitor the serial communication. It supports features such as changing text color, logging received data to a file, and executing functions based on received data.

2. Key Features:

2.1 Configuration:

- **Port Configuration:** User can configure the serial port, baud rate, and other settings.
- **Start Screen:** Offers predefined configurations and an option to customize.

2.2 Monitoring:

- **Serial Header:** Displays key information like port, baud rate, log file, timestamp, and batch number.
- **Textbox Function:** Displays received data with an optional timestamp.
- **Functions Handling:** Executes specific functions based on received data (e.g., clearing the screen, changing color).

2.3 Logging:

- **File Logging:** Logs received data to a file, with an option to enable/disable.
- **Timestamp:** Optionally includes a timestamp in the log file.

2.4 User Interaction:

- **Load Screen:** Allows customization of serial port, baud rate, logging, and display lines.
- **Serial Write:** Enables the user to send data to the connected device.

2.5 Dynamic Behavior:

- **Continuous Reading:** Uses a loop to continuously read data from the serial port.
- **Batch Management:** Tracks and displays batches of received data.
- **Color Change:** Allows changing the text color based on received commands.

3. Working Stages:

3.1 Initialization:

- The program initializes necessary variables and structures.
- The console window is cleared and color is set.

3.2 Start Screen:

- User is presented with options to start with predefined configurations or customize.
- User input is processed to configure the serial monitor.

3.3 Load Screen:

- If the user chooses to customize, additional information is gathered (port, baud rate, etc.).
- Loading bar is displayed.

3.4 Serial Header Display:

- Key information is displayed at the top of the console.

3.5 Serial Reading Loop:

- The program continuously reads data from the serial port.
- Received data is displayed in the textbox, functions are executed, and the batch is managed.

3.6 User Interaction:

- The user can send data to the connected device. (By Tab + S)

4. Enhancements:

4.1 Function Expansion:

- Additional functions could be added based on specific commands received.

4.2 Error Handling:

- Improved error handling for file operations, serial communication, and user inputs.

4.3 Modularity:

- Code could be organized into functions and classes for better modularity.

5. Usage with Arduino:

- The program can be used with an Arduino or similar device sending serial data.
- Arduino can send commands (e.g., change color) that the program interprets.

6. File Logging Usage:

- Logging is controlled by user choice (enable/disable).
- Received data, along with timestamps, is logged to a file.

7. Color Changing Usage:

- Serial commands like "Serialcolora" change the text color in the console.

8. Improvements:

- The program could benefit from more structured code, error handling, and modularity.
- Consideration for potential naming conflicts or unintended side effects.

Credits:

This project was developed by Ayush Soni, a student at IIIT-Trichy. The entire codebase and design were crafted by Ayush Soni with the aim of creating a versatile Serial Monitor application.

In summary, the program is a versatile Serial Monitor with various features catering to configuration, monitoring, and interaction with a connected device. It demonstrates the flexibility of using serial data for dynamic interactions and control.