High Level Overview

Purpose

The Serial Commander is a terminal user interface (TUI) for sending commands to an STM32 CAN board over a serial connection. It provides a live split-screen view of outgoing commands and incoming serial data, with editable motor and PIO parameters.

Prerequisites

The STM32 firmware must be built with serial command reception enabled. The firmware must parse newline-terminated ASCII strings from UART/USB-CDC and handle the following command tokens:

- `speed <erpm>`
- `stop`
- `profile <erpm> <accel> <time_ms>`

Without this firmware support the TUI will open and display incoming data, but sent commands will have no effect aside form rebuilding and flashing the firmware

Core Responsibilities

The tool handles three concerns in one interface:

- **Command dispatch** — sends formatted serial strings (`speed`, `stop`, `profile`) to the STM32 on keypress
- **Live serial monitoring** — streams and colour-codes all incoming serial output in real time
- **PIO integration** — triggers PlatformIO build/flash operations without leaving the terminal

Architecture

Three concurrent components run during a session:

Main TUI Loop — Draws the interface at 100 ms intervals, handles keypresses, and dispatches commands or parameter edits. Runs on the main thread via `curses.wrapper`.

Serial Reader Thread — Daemon thread that continuously calls `ser.readline()` and appends decoded lines to a shared `rx_log` deque. Terminates automatically when the serial connection drops.

PIO Runner Thread — Spawned on demand when the user triggers a build. Runs `pio run -t <target> -e <env>` as a subprocess and streams stdout into `rx_log`. Does not block the TUI.

All three components share `rx_log` and `tx_log` as `collections.deque(maxlen=200)` — thread-safe for the single-producer/single-consumer append and iteration patterns used here.

Revision #1
Created 2026-07-02 09:30:57 UTC by Nikolaos Diamantopoulos
Updated 2026-07-02 09:32:01 UTC by Nikolaos Diamantopoulos