

protocol_tool_gui.py

A point-and-click window for `protocol_tool.py` — pick a port and a file from dropdowns and dialogs instead of typing command-line flags.

It does not reimplement anything. It calls `protocol_tool.py` directly, so the same validation and the same handshake apply.

What you get

- **Serial port dropdown**, auto-populated from the ports actually present, with a *Refresh* button for when you plug the Generator in after starting the app.
- **Browse** dialog for picking the protocol file.
- **Receive Fm Generator** and **Send to Generator** buttons, each behind a confirm prompt. They are worded from the PC's point of view so they mirror the Generator's own menu: *Receive Fm Generator* here is the Generator's *Send To CPU*, and *Send to Generator* here is its *Receive Fm CPU*. Each button pairs with the opposite-sounding option on the Generator, as the two ends of one cable should. "Download" and "Upload" remain only as the command-line subcommands and the internal protocol names.
- **Activity log** showing the same TX: /RX: traffic the command-line tool prints, live as it happens.
- **Status line** at the bottom.

The transfer runs on a background thread, so the window stays responsive and the log fills in as the transfer proceeds rather than all at once at the end. Both buttons are disabled while an operation is running.

Using it

```
python3 protocol_tool_gui.py
```

Then:

1. Start the matching mode on the Generator — **USB Comm** — **Send To CPU** to download, **USB Comm** — **Receive Fm CPU** to upload.
2. Pick the serial port (usually `/dev/ttyUSB0`).
3. Name the protocol file — **Browse**, or type the name straight into the field and press **Return**. A bare `protocols - 12 . txt` is enough; the folder is filled in from the one you used last.
4. Click **Receive Fm Generator** or **Send to Generator**, and confirm.

It must be run from the folder containing `protocol_tool.py`, since it imports it.

Remembered folder

The last folder you used is saved to `~/.protocol_tool_gui_settings.json` and reused next time — including folders you type by hand, not just ones picked through Browse. It lives in your home directory rather than beside the script, so every copy of the tool across project versions shares the same “where I was last working” memory.

If no folder has been used yet, it falls back to a `Protocols - x` folder beside the script, then to the script’s own folder.

Two details worth knowing

Browse lets you name a file that does not exist yet. Receiving is the reason: the number in `protocols - N.txt` tells the Generator which SD card slot to send, so fetching slot 12 for the first time means naming a `protocols - 12.txt` you do not have a copy of. Browse is a save-style dialog with overwrite confirmation turned off, so it can pick an existing file for sending or accept a typed name for receiving.

Type the name, don’t retype the path. Changing which slot you want means editing two digits, so the field accepts a bare `protocols - 12.txt` and supplies the remembered folder itself. Press **Return** and it expands to the full path and shows the slot it resolved to in the status line.

The name is checked at all three points — when you press Return in the field, when the Browse dialog closes, and again when you click a transfer button (the field can still be edited after the first two). Anything that isn’t `protocols - N.txt` with `N = 0–99` is refused up front, with the offending text selected so it can be corrected in place, rather than after you have already started the transfer on the Generator.

The Generator itself cannot be given a bad name: the PC sends a *number*, and the device builds `/protocols - N.txt` from it, so an unreachable file cannot be created on the card even by a typo.

The confirm prompts name the matching step on the device — *USB Comm → Send To CPU* for a receive, *USB Comm → Receive Fm CPU* for a send. The two sides are worded from opposite points of view, so pairing *Send* with *Send* is the easiest mistake to make and the prompt heads it off.

A refusal from the Generator is reported immediately. If the card has no `protocols - N.txt` in the slot you asked for, the device answers `ERROR: FILE_NOT_FOUND` and the transfer stops there with an explanation. It no longer sits out the full 20-second handshake window and then blames a timeout.

Requirements

Needs `tkinter` and `pyserial`. On Debian/Mint, `tkinter` may be a separate package:

```
sudo apt install python3-tk
```

Related

- `protocol_tool.py` — the engine underneath; all errors come from there
- `protocol_editor.py` — for editing the file before uploading