Robust Chat v2.20 — Complete Instruction Manual

1. Overview

Robust Chat v2.20 is an integrated amateur-radio communication suite designed for reliable digital chat under weak-signal or narrow-bandwidth conditions. It unites text messaging, beaconing, position sharing, automatic relaying, CAT transceiver control, and AI-assisted operations within a dark, compact interface.

Supported hardware includes the LiNK500 TNC, Teensy RPR TNC, and SCS Tracker(untested). The LiNK500 is ideally suited, offering both an internal robust modem and a separate CAT control interface. Bluetooth connectivity is also available RF data exchange (RF testing is required in BT).

All operational data—settings, messages, link graphs, and positions—are stored in JSON files beside the executable, in the same folder, ensuring portability between computers.

2. Installation & Startup

System Requirements:
- Windows 10 or 11
- Python 3.9+ (for .py version)
- Packages: pyserial, cryptography
- Optional hardware: TNC (LiNK500, Teensy RPR, SCS Tracker), GPS receiver

Installation:
1. Create a folder named Robust Chat and place Robust_Chat.exe and JSON files in that same folder.
2. Run via Python (install dependencies if using python) or simply double-click the EXE Windows version.

Startup:
On launch, the program loads fonts, settings, COM ports, and saved data. The window applies dark mode automatically. Logs appear in the console as [LOG] entries, which can be found in the Chat window under Settings (if required) .

3. Main Interface Layout

The interface is divided into three main areas:
- Header Toolbar: Contains TNC COM selection, MYCALL, GRID, R300/R600 buttons, frequency dropdown, and navigation tabs (Main, GPS, Beacon, Locator, CAT, Misc).
- Sidebar: Displays chats and the 'Beacons Heard' list with colour-coded freshness (green,

amber, red).
- Main Page: Chat messages, input field, Encrypt/Decrypt buttons, Send button.

Outgoing messages have a green background , incoming messages directed to your callsign are orange text, and all timestamps are local to your position.

4. Connecting to Radios

Before connecting, enter MYCALL (your callsign) and GRID (Maidenhead locator, 6 (5KM) or 8 (500M) characters). These must be set before the Connect button becomes active.

Supported TNCs:
- LiNK500 TNC: Dual USB ports (Port A for data, Port B for CAT) and Bluetooth (connection tested:OK, RF TX RX needs testing in Bluetooth mode to confirm no lags in data).
- Teensy RPR TNC and SCS Tracker: Standard KISS-compatible devices.

Select COM port, click Connect. Successful KISS mode initialization is logged as [TNC] KISS mode active. Optional CAT control supports Icom (CI-V), Kenwood/ELECRAFT ASCII, and Yaesu Binary protocols. R300 and R600 symbol rates are selectable at runtime.

5. Message System

Messages are composed in the Main tab using the To: field, text input, and Send button at the bottom of the screen.. Directed messages use ACK confirmation; broadcasts use CQ. Messages are stored automatically in messages_v1.json for EMCOM and general message history. CLEAR MESSAGES button will clear the window and the messages in the messages_v1.json as well as any beacons in the link_graph.json, for a fresh start.

WARNING ENCRYPTION MAY BREACH LICENSE CONDITIONS: Encryption uses AES-256-GCM. The ACK system provides one-tick (sent) and two-tick (confirmed) indicators with retries. Auto-relay silently forwards unacknowledged messages once. AI-assisted chat generates replies through LM Studio or compatible endpoints. Message colours and UTC timestamps improve readability.

6. Encryption System

In the UK certain ‘User services’ can request encrypted messages be sent in emergencies or high priority situations, by RAYNET. So it is included for such emergency use. Use outside of this may breach license conditions.

Messages can be encrypted using AES-256-GCM. The encryption workflow:
1. Type Message in send message field and enter To: CALLSIGN. Press PADLOCK to Encrypt before sending.

2. Enter password; press Encrypt; ciphertext replaces plaintext with [ENC] prefix. SEND
3. Receiver presses Decrypt and enters same password. For operational purposes it would be suggestible that groups have a daily changing password based on a table of day date month, words assigned to each. In this case a user can quickly look at the table i.e Monday = aPple , 14 = rEd, January = JupiTter > aPplerEdJupiTer , for example.

No keys are stored; security depends on user password integrity.

7. Acknowledgment (ACK) Reliability System

Directed messages include an ACK ID (e.g., [ACK:84JQ]) 3 retry system. The receiving station automatically replies with an identical ACK frame.

Indicators:
✓ Sent (awaiting ACK)
✓✓ Delivered (ACK received)
! Failed (no ACK after retries)

Pending ACKs persist after restart until confirmed or expired.

8. Auto-Relay System

Stations may silently relay third-party messages once if both sender and receiver are known from recent beacons. Logged as [AutoRelay] events. Auto-relay reduces delivery failures across multi-hop paths. Can be toggled in settings.json. This is triggered on conditions that a call is heard to a callsign which is not ack'd, if the To callsign is in your beacon heard list, auto relay sends a one shot TX of the original message, it returns the same way but may use an alternative users system if they have the destination in their beacon heard list.

9. Beacon System

Automatic beacons broadcast callsign and position at intervals: Off, 5, 10, or 15 minutes. Each received beacon updates link_graph.json, forming a network topology with coloured freshness (green/amber/red). Entries older than the set threshold are pruned automatically. Beacons are sent with co-ordinates with a priority order of live GPS first and then Fixed co-ordinates and finally if none of those exist then the Maidenhead Grid is converted to lat lon co-ordinates. Note if you can find your maidenhead grid in 6 figures you have a 5KM precision, 8 figures or more and you are 500M precision to location. There are online sites to get 8+ figure maidenheads. This info is used to plot stations in the LOCATOR cardinal radar system tab.

10. Locator and Mapping

The Locator tab uses the Maidenhead system to plot nearby stations as compass markers showing distance and bearing from your station. positions.json stores known coordinates, which are removed after 24 hours. Your own position appears in the centre with colour-coded beacon indicators.

11. GPS Integration

Supports dual GPS sources: PC GPS and LiNK500 TNC and TEENSY RPR TNC GPS. NMEA sentences are parsed to update latitude and longitude fields. Fixed manual coordinates can be entered if no GPS is available. Data saved to positions.json. The PC GPS is refreshed every 5 minutes. If using the GPS from the TNCs this is a manual GET GPS process, as it requires the Chat system to exit KISS mode get GPS and reset KISS mode each time.

12. CAT (Computer-Aided Transceiver) Control

The CAT tab provides radio control via Icom CI-V, Kenwood/ELECRAFT ASCII, or Yaesu Binary protocols. Auto-QSY , for 24hour stations, switches between day and night bands(default 05:00–21:00 / 21:01–04:59). Radios and default baud rates are defined in radios.json. Manual frequency selection is available via dropdown.

13. Offline AI Assistant Mode

An optional AI enable checkbox connects to a local LM Studio or compatible API (127.0.0.1:1234). When enabled, it can auto-reply or summarize messages using a 'ham operator' personality. AI settings (endpoint, model, prompt) are adjustable in settings.json. Operators with AI, their beacons appear in lime green with [A.I] tag. LM studio is free and when downloaded enables a local OFFLINE A.I, the idea is that those with modern PCs can install local A.I with no internet required after download to serve other Robust Chat users on any topic they require i.e survival, mechanical, electronic, or anything you can think of!

14. Miscellaneous Features

- R300 / R600 speed toggle
- Auto-save of all settings
- Ctrl + Alt + Q diagnostic hotkey
- Unicode fonts (Segoe UI Emoji / Noto Color Emoji)
- Dark theme
- Compact, field-optimised layout

15. File System and Persistence

JSON files stored alongside executable:
- settings.json — configuration

- messages_v1.json — chat history
- link_graph.json — beacon links
- positions.json — known coordinates
- radios.json — radio definitions
- freqs.json — frequency list

Copy the entire folder to transfer to another system. Backup messages_v1.json regularly.

16. Troubleshooting and Diagnostics

Common issues:
- Missing pyserial/cryptography → reinstall dependencies.
- COM port busy → close other serial apps.
- CAT failure → check CIV address and baud rate.
- GPS missing → verify port and NMEA output.
- Messages missing → confirm JSON file paths in EXE build.
-Radio list/frequencies missing from drop downs > Check Radios.json and freq.json are present in the same folder as the program.

17. Version History and Credits

Version 2.20 Highlights:
- CAT integration
- Dual GPS (PC + LiNK500)
- Bluetooth detection for LiNK500
- ACK + Auto-Relay subsystems
- AES-256-GCM encryption
- AI assistant integration

Credits:
Development: O.Cohen M0OLI.
Documentation:13th Nov (2025)
Acknowledgments: Thank you to those initial testers of Robust Chat and promoters of the testing group Julian OH8STN, the details of which can be found on his website. Designer of the LiNK500 TNC Oliver Harms DL4KA. Robert Wolters DM4RW designer of the Teensy RPR TNC, and SCS Tracker team.