

piText

Text-to-waterfall transmitter for piHPSDR

User Manual – version 0.1

For Radio Amateurs

Contents

1. Introduction
2. Operating considerations
3. Requirements and installation
4. Setting up piHPSDR
5. Quick start
6. The window, control by control
7. How it works
8. Sideband, Mirror and Flip
9. TCI details
10. Suggested settings
11. Saved settings
12. Troubleshooting
13. Appendix A: Default parameters
14. Appendix B: Files

1. Introduction

piText lets you type a short message and “write” it into the waterfall of anyone watching your signal. It converts the text to audio whose *spectrogram* (frequency across, time down) spells out the message, and passes that audio to **piHPSDR** over the **TCI** (Transceiver Control Interface) WebSocket protocol. piHPSDR transmits it like any other SSB audio.

Nothing is encoded or decoded. The audio is a bank of unmodulated tones whose amplitudes are switched on and off to form pixels, so the message is only visible to a person looking at a waterfall (a local SDR display, a remote WebSDR, a skimmer screenshot). It will sound like a warble or a burst of noise on a speaker. The idea follows the image-painting tool *spectrum_painter* (github.com/polygon/spectrum_painter), applied to text.

Key facts:

- Audio stays within **300–2700 Hz**, inside a normal 2.7 kHz SSB passband.
- Output is mono, 48 kHz, peak level 0.5 of full scale.
- The radio’s sideband (USB/LSB) is read automatically so the text is not mirrored.
- Written in Python with a Tk window; runs on Linux Intel/AMD/RaspberryPi (developed on Raspberry Pi OS).

2. Operating considerations

As with any transmission, you are the control operator and are responsible for complying with your licence and the band plan.

- **Identify.** Transmit your callsign as your licence requires. The picture is not a substitute; consider including your callsign in the message itself and, if needed, identify by voice or CW before and after.
- **Where it is allowed.** Waterfall art is a form of data/image transmission. Use it only in segments and modes where such emissions are permitted by your administration, and never on a frequency in use. Listen first. Many operators restrict it to test periods, local contacts or experimental activity; check your national rules.
- **Bandwidth.** The signal occupies about 2.4 kHz of audio (300–2700 Hz), equivalent to a full SSB signal. Choose a clear spot.
- **Duty cycle.** A message is a continuous carrier-like transmission. A 60 second message is a 60 second key-down. Check your transmitter's duty-cycle limits and any PTT time-out, and keep messages short.
- **Drive and linearity.** piText makes a multi-tone signal. Keep drive and ALC modest so the PA stays linear; splatter would smear the text and cause interference. Compare against a normal voice peak and back off if the ALC is active.
- **First tests.** Use a dummy load, a low-power setting or your own receiver/WebSDR before going on air.

3. Requirements and installation

3.1 Requirements

- piHPSDR (the “Improved” version with TCI support) running and connected to your radio.
- Python 3.11 or later (the program uses `asyncio.timeout`).
- Python packages `numpy`, `Pillow` (with ImageTk) and `websockets`, plus Tk (`python3-tk`).
- piText may run on the same computer as piHPSDR (host 127.0.0.1) or on another machine on the network.

3.2 Installer

From the piText folder double click the install.py file using your favourite file manager or if you prefer, via a terminal run:

```
python3 install.py
```

The installer is a small graphical program. It:

1. checks the system,
2. installs the missing apt packages (`python3-tk`, `python3-numpy`, `python3-pil`, `python3-pil.imagetk`, `python3-websockets`) – you will be asked for your password,
3. adds a **Ham Radio** menu entry for piText (creating the Ham Radio sub-menu if needed),
4. creates a piText icon on the desktop.

`python3 install.py --dry-run` shows what would be done without changing anything.

The same program can uninstall the menu entry and icon.

If `python3-tk` is missing, the installer cannot open its window; install `python3-tk` first via a terminal with:

`sudo apt install python3-tk`.

And then run the installer again by double clicking the install.py file or running it in a terminal.

3.3 Manual start

```
pip install -r requirements.txt
python3 -m pitext
```

(Tk itself comes from the operating system package, not pip.) The window title shows the version number.

4. Setting up piHPSDR

1. Open the piHPSDR main menu and choose **CatTci**.
2. On the **TCI** line, set the port (default **40001**; the allowed range is 35000–55000) and tick **Enable**. Changing the port restarts the TCI server.
3. Select the transmit mode: **USB**/**DIGU** or **LSB**/**DIGL** on the transmitting VFO. Conventionally LSB below 10 MHz and USB above, but piText follows whatever the radio is set to.
4. Make sure the TX filter passes at least 300–2700 Hz. A narrower filter will cut the top or bottom of the lettering.
5. Turn off speech processing/compression, and disable any equaliser that would alter the audio. Set the drive for a clean signal (see section 2).
6. Tune to a clear frequency.

You do not have to select any audio input in piHPSDR. When piText sends, it keys the radio with `trx:0,true,tci;`, which selects TCI as the TX audio source for that transmission.

The remote machine's firewall must allow the chosen TCP port if piText is not on the same computer as piHPSDR.

5. Quick start

1. Start piText from the Ham Radio menu, the desktop icon, or `python3 -m pitext`.
2. Check **Host** (127.0.0.1 if piHPSDR is on the same computer) and **Port** (40001).
3. Type your message in the text box, e.g. `CQ CQ DE G4XYZ`. The preview updates as you type. The status line below the options gives the time on air.
4. Leave **Auto (LSB/USB)** ticked, so piText reads the radio's mode.
5. Click **Send**. The radio keys, the progress bar fills, and the status line shows the result. The text appears in the waterfall, reading with your head tilted to the left (see section 7.4).
6. Click **Stop** at any time to abort. piText un-keys the radio cleanly.

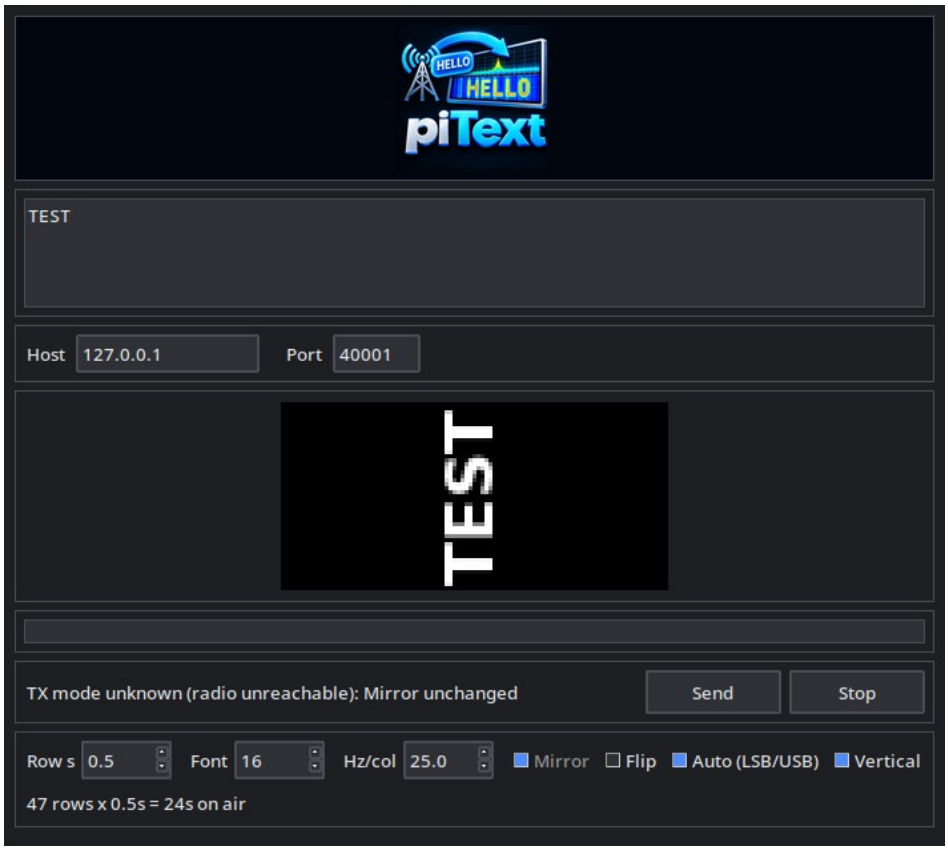


Figure 1: The piText main window.

6. The window, control by control

From top to bottom:

Item	Function
Message box	The text to send. Several lines are allowed; each new line is a separate line in the picture. Contents are saved between sessions.
Host, Port	Address of the piHPSDR TCI server (default 127.0.0.1, 40001). Leaving either field triggers a new check of the radio's mode.
Preview	Black-and-white picture of what the waterfall will show. In vertical mode it is drawn as the waterfall will display it: the first character sent is at the bottom. It reflects Font, Hz/col, Vertical, but not Mirror and Flip (these affect the transmitted audio, not the preview).
Progress bar	Fraction of the audio handed to the radio.
Status line	Last action: detected TX mode, "Transmitting...", "Done: …", "Stopped: …" or an error.
Send	Renders the audio and transmits it. With Auto ticked it first queries the radio's mode. Disabled while a transmission is in progress.
Stop	Aborts the transmission. The radio is returned to receive.
Row s	Seconds per picture row (0.04–5, default 0.5). Longer rows give slower, easier-to-see text on slowly scrolling waterfalls but take longer. Never less than 1/(Hz/col).
Font	Font size, 6–40 (default 12). In vertical mode the font shrinks automatically if the lines would not fit the 2.4 kHz band height.
Hz/col	Tone spacing, 5–100 Hz (default 25). Smaller spacing = more tones and finer vertical resolution of the lettering, but needs a longer row time (see section 7.3).
Mirror	Reverses the frequency order. Needed for LSB. Greyed out while Auto is ticked, and then set for you.
Flip	Reverses the time order, for waterfalls that scroll the other way (new data at the bottom).
Auto (LSB/USB)	Sets Mirror from the radio's TX mode, before every Send and at start-up (default on).
Vertical	Text runs along the time axis (default). Untick for horizontal, word-wrapped text.
"N rows x Ts = X s on air"	Number of picture rows, row time and total transmit time (excluding the 0.5 s of silence at each end).

7. How it works

7.1 Audio synthesis

The text is drawn with a bold TrueType font (DejaVu Sans Bold, falling back to Liberation Sans Bold) into a bitmap. Each bitmap **column** is one audio tone; each bitmap **row** is one time slice. With the defaults the tones run from 300 Hz to 2700 Hz in 25 Hz steps, which is **97 tones**.

The audio is an oscillator bank: one sine wave per column, with its amplitude following the pixel brightness, interpolated linearly between rows. The tones have fixed pseudo-random phases, which keeps the peak-to-average power ratio low and makes best use of the transmitter. The sum is scaled to a peak of 0.5 and padded with 0.5 s of silence each end.

7.2 Waterfall geometry

A waterfall plots frequency horizontally and time vertically, with new data entering at the top and scrolling down. piText therefore sends the picture bottom-first, so it appears the right way up.

7.3 Row time and tone spacing

Two adjacent tones spaced f Hz apart can only be told apart if each pixel lasts at least $1/f$ seconds. At 25 Hz that is 40 ms, the minimum row time. piText enforces Row s ≥ 1/(Hz/col).

The 0.5 s default is deliberately slow. Web-based SDR waterfalls typically scroll slowly; with 40 ms rows the whole message collapsed into a line or two on screen. If you only watch a fast local display, shorter rows are fine.

Time on air = number of rows × Row s. Examples with the defaults (vertical mode, Row s = 0.5, font 12):

Message	Rows	Time
CQ	22	11 s
73 DE G4XYZ	91	45.5 s
CQ CQ DE PITEXT	118	59 s
CQ CQ CQ DE G4XYZ G4XYZ K	202	101 s

On average each character costs about 8 rows, i.e. roughly 4 s of transmission at 0.5 s per row.

7.4 Vertical and horizontal modes

Vertical (default). The text runs along the time axis, so only the letter *height* has to fit in the 2.4 kHz band; the message length is limited only by how long you are willing to transmit. To read it, **tilt your head to the left**: the text runs bottom to top with letter tops pointing left (on USB). There is no word wrap; a new line in the message box is another line stacked in frequency, centred in the band. The font shrinks automatically to fit.



Figure 2: The preview of “CQ DE PITEXT” in vertical mode (left; first character at the bottom) and in horizontal mode (below).

Horizontal. The text runs along the frequency axis and reads normally, but is word-wrapped to the 2.4 kHz width, so only a few characters fit per line (about 97 pixels across) and the font must be small. Each line takes a block of rows. Use it for very short messages.

8. Sideband, Mirror and Flip

A waterfall shows RF frequency. On an upper-sideband signal, RF rises with audio frequency; on lower sideband, RF *falls* as audio frequency rises. Without correction, text sent on LSB would appear mirrored left-to-right. piText compensates with **Mirror**.

TX mode	Mirror
USB, DIGU	off
LSB, DIGL	on
CW, AM, FM, etc.	unchanged (piText does not touch it)

With **Auto (LSB/USB)** ticked, piText connects to piHPSDR, reads the transmit mode, and sets Mirror. It does this at start-up, whenever you leave the Host or Port fields, and just before each Send. The TX mode is that of VFO A, or VFO B when split is on. This check is read-only and never keys the radio. If the radio cannot be reached the status line says “TX mode unknown” and Mirror stays unchanged. Untick Auto to set Mirror yourself.

Flip reverses the time order. Use it if your waterfall scrolls upward from the bottom (new data at the bottom) so that text appears reversed top-to-bottom.

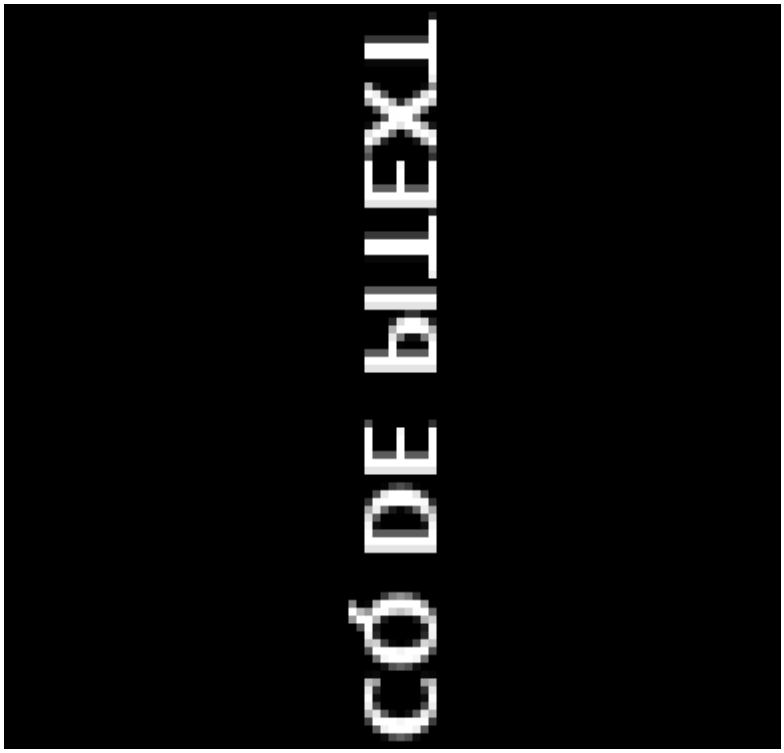


Figure 3: The same message mirrored in frequency (what a normal-orientation message looks like when received on the wrong sideband).

9. TCI details

For those interested in what happens on the wire. The behaviour follows piHPSDR's own `tci.c`/`tci_audio.c`.

- piText opens a WebSocket to `ws://host:port`.
- It sends `trx:0,true,tci;`: transmitter 0 on, audio from TCI. The server then expects 48 kHz float32 audio, stereo-interleaved; piText sends the same signal on both channels.
- The server is the clock. It emits a **TX_CHRONO** binary frame for every 512 samples (about 10.7 ms). piText answers each with a TX_AUDIO frame of 512 stereo frames (1024 floats).
- The server needs at least 4096 frames buffered before it starts to play, so on the first chrono piText sends a burst of 12 blocks, then keeps about 40 blocks (0.43 s) queued ahead.
- piText keeps reading the incoming messages throughout. If no chrono arrives within 5 s, the transmission is abandoned with an error.
- When finished (or stopped) it waits for the audio to drain, sends `trx:0,false;` and waits for piHPSDR's echo before closing the connection. Closing early could leave the radio stuck in transmit.

The final status line reports blocks sent, chrono frames, elapsed time and the largest gap between chrono frames (ideal about 11 ms). Large gaps suggest network or CPU trouble.

On a Raspberry Pi you may see `tx_audio_write: suppressing audio block` messages in piHPSDR. These come from the local PulseAudio TX monitor (headphones) overrunning its latency limit, not from the transmitted audio, and can be ignored.

10. Suggested settings

Purpose	Row s	Hz/ col	Font	Notes
Remote WebSDR (slow scroll)	0.5 (or more)	25	12	Defaults. Increase Row s if text looks squashed.
Local fast waterfall	0.1–0.2	25	12	Shorter transmission.
Finer, larger lettering	0.5	15–20	16–20	More tones; keep Row s ≥ 1/Hz-per-col. The font is shrunk automatically if it will not fit.
Short message, short time	0.3	25	10	Fewer, narrower letters.

Waterfall zoom, speed and FFT size on the receiving end have a large effect on appearance. Experiment on a dummy load or a local receiver and adjust Row s until the text looks right.

11. Saved settings

piText saves its settings in `~/.config/pitext/settings.json` (or `$XDG_CONFIG_HOME/pitext/settings.json`) when you click Send and when you close the window, and reloads them at start. The file is plain JSON:

Key	Meaning
text	Message text
host, port	TCI server address
row_seconds, font_size, pixel_hz	Row s, Font, Hz/col
mirror, flip, auto_mirror, vertical	The four check boxes

Delete the file to return to the defaults.

12. Troubleshooting

Symptom	Remedy
Text is mirrored left/right	Mirror is wrong for the sideband. Tick Auto, or set Mirror by hand (on for LSB/DIGL, off for USB/DIGU).
Text is upside down / reads in the wrong direction in time	Tick or untick **Flip**.
Message is a smeared blob or one thick line	Row s is too short for the waterfall speed. Increase Row s, and keep it ≥ 1/(Hz/col).
Top or bottom of letters missing	The radio's TX filter is narrower than 300–2700 Hz. Widen it.
Letters look fuzzy or have "splatter" around them	Drive/ALC too high or processing on. Reduce drive, disable the speech processor.
Error: connection refused / cannot connect	TCI not enabled in piHPSDR (CatTci menu), wrong host or port, or firewall.
"TX mode unknown (radio unreachable)"	The mode query failed. Check host/port. Mirror is left unchanged.
"no TX_CHRONO received from piHPSDR"	The radio did not start transmitting within 5 s. Check that piHPSDR is running with TCI enabled and connected to the radio, and that nothing is blocking transmit.
Radio stays in transmit	Press Stop in piText, or un-key in piHPSDR. Do not kill the program during transmission if you can avoid it.
Text is tiny	Many lines force the font to shrink so that they fit in the band height. Use fewer lines or a shorter message.
Window will not start ("No module named tkinter")	`sudo apt install python3-tk` or run `install.py`.

Appendix A: Default parameters

Parameter	Default
Sample rate	48 000 Hz (fixed by piHPSDR TCI)
Lowest / highest tone	300 Hz / 2700 Hz
Tone spacing (Hz/col)	25 Hz (97 tones)
Row time	0.5 s
Font size	12 (shrinks to fit in vertical mode, minimum 6)
Silence before / after message	0.5 s
Peak amplitude	0.5 of full scale
TCI host / port	127.0.0.1 / 40001
Vertical / Flip / Auto	on / off / on

Appendix B: Files

File	Purpose
pitext/render.py	Text to bitmap to audio
pitext/tci.py	TCI client, TX mode query
pitext/gui.py	Tk user interface
pitext/settings.py	Saved settings
install.py	Installer / uninstaller
tests/	Unit tests (`python3 -m unittest discover -s tests`)

References

- spectrum_painter – https://github.com/polygon/spectrum_painter
- piHPSDR – the TCI implementation is in `tci.c` and `tci_audio.c`.