

# T41 Pi Display — Raspberry Pi 5 side

Runs on the Raspberry Pi 5 (2GB), drives the Raspberry Pi Touch Display 2 (7"), and talks the PiLink UART protocol to the Teensy 4.1. See the project doc `teensy-pi-uart-protocol.md` for the full protocol design and hardware decision history, and `INTEGRATION_PROCEDURE.md` for the full build steps.

**Updated 2026-09-26 (Mustang 2 V1.0.15).** Now running on real hardware, with a two-row toolbar of dropdowns, a LAN-only web remote with live RX audio, and desktop icons. Setup for those is in `INTEGRATION_PROCEDURE.md` Section 9; per-version history is in `../CHANGELOG.md`.

**This revision replaces the earlier Pi 3B + buydisplay ER-TFTM070-5V4 (RA8875/FT5316) build.** The Touch Display 2 is a standard DSI panel that Raspberry Pi OS auto-detects and drives as a normal framebuffer + 5-point capacitive touch input device - there's no SPI/I2C wiring or register-level display driver anymore, and the app is now a windowed Tkinter UI instead of drawing directly to the panel.

## Wiring (Pi 5, 40-pin header)

| Signal                           | Function                                   | Physical pin    |
|----------------------------------|--------------------------------------------|-----------------|
| DSI (22-way-to-15-way FFC cable) | Touch Display 2 video + touch              | DISP connector  |
| GPIO power cable, red            | Display +5V (shared with Pi's own 5V rail) | Pin 2           |
| GPIO power cable, black          | Display GND                                | Pin 6           |
| UART TXD                         | to Teensy RX                               | Pin 8 (GPIO14)  |
| UART RXD                         | from Teensy TX                             | Pin 10 (GPIO15) |

No SPI, I2C, or logic-level jumper to deal with - see `INTEGRATION_PROCEDURE.md` Sections 2-3 for the full power/DSI wiring, including why the Pi and display share one power feed.

## Pane mirroring (Teensy display <-> Pi display)

The Pi's UI in `display.py` deliberately mirrors the Teensy's own onboard display panes, defined in `PhoenixSketch/MainBoard_DisplayHome.cpp` (search that file for `WindowPanes` for the authoritative list). Both sides use the same pane **names** and the same relative layout, scaled from the Teensy's 800x480 screen to the Pi's 1280x720 canvas (factor: 1.6x width, 1.5x height).

| Teensy pane (MainBoard_DisplayHome.cpp) | Teensy rect (x,y,w,h) | Pi pane (display.py) | Pi rect (x,y,w,h) |
|-----------------------------------------|-----------------------|----------------------|-------------------|
| PaneVFOA                                | 5,5,515,50            | vfo                  | 8,8,824,75        |
| PaneFreqBandMod                         | 5,60,380,30           | freq_band_mod        | 8,90,608,45       |
| PaneSAMOffset                           | 390,60,110,30         | sam_offset           | 624,90,176,45     |
| PaneSpectrum                            | 5,95,520,345          | spectrum             | 8,143,832,518     |
| PaneStateOfHealth                       | 5,445,260,30          | state_of_health      | 8,668,416,45      |
| PaneSWR                                 | 535,15,150,40         | swr                  | 856,23,240,60     |
| PaneTXRXStatus                          | 710,20,60,30          | txrx_status          | 1136,30,96,45     |
| PaneSMeter                              | 515,60,260,50         | smeter               | 824,90,416,75     |
| PaneAudioSpectrum                       | 535,115,260,150       | audio_spectrum       | 856,173,416,225   |
| PaneSettings                            | 535,270,260,170       | settings             | 856,405,416,255   |
| PaneNameBadge                           | 535,445,260,30        | name_badge           | 856,668,416,45    |

**Hardware-verified end-to-end as of 2026-09-25** on a real Pi 5 + Touch Display 2 + Teensy 4.1 setup (frequency, band/mode, S-meter, PTT). `PiLink.h`'s STATUS/SMETER/SPECTRUM messages carry frequency (the corrected dial frequency, `GetTXRXFreq()`), band, mode, PTT, S-meter, spectrum bins, power level, filter bandwidth, keyer WPM, SWR/forward power, modulation, SAM carrier offset, effective passband edges, audio-spectrum span, and CW filter index - so every telemetry pane is now wired:

- `vfo`, `freq_band_mod` (band/mode names via `display.py`'s `_BAND_MODE_NAMES`, plus the `(LSB)/(SAM)`/etc. suffix via `_MODULATION_NAMES`), `spectrum`, `txrx_status`, `smeter` - hardware-verified.
- `swr` (mirrors `DrawSWRPane()`'s exact clamping and TX-active red/white coloring) and `sam_offset` (mirrors `DrawSAMOffsetPane()`, blank outside SAM) - wired and code-verified, not yet exercised live (need a keyer connected for

SWR/TX, and SAM mode selected for the offset reading).

- `audio_spectrum` mirrors `DrawAudioSpectContainer()` - note this Teensy pane is a static frequency-scale ruler plus passband/CW-filter marker lines, **not** a live audio FFT plot despite the name; the Pi side does the same (canvas lines repositioned on change, not a live waterfall) - wired and code-verified, not yet exercised live.
- `state_of_health` (repurposed to show power/filter/WPM - see the NOTE in `display.py`'s module docstring) - hardware-verified.

`settings` is telemetry-driven (`PILINK_MSG_SETTINGS`, 2 Hz). The BAND +/- buttons that used to live here were replaced by the toolbar's dropdowns (Menu, Vol, Band, Mode, Demod, Course Inc, Fine Inc, Zoom, BW, Noise, Decode, Home), which send `VIRTUAL_BUTTON` and `VIRTUAL_ENCODER` uplinks. `name_badge` needs no protocol change - it's driven by the Pi's own clock, same as the Teensy uses its own RTC.

**One Pi 5-specific gotcha, worth knowing if you rebuild this:** `config.py`'s `UART_PORT` must be `/dev/ttyAMA0`, not `/dev/serial0` - on a Pi 5 (unlike every earlier Pi), `/dev/serial0` points at the debug connector's UART, not the GPIO header. See `INTEGRATION_PROCEDURE.md` Section 9 for the full diagnosis if this bites you.

**Rule for future changes:** if you change what a Teensy pane shows (or add a pane), update `PiLink.h`'s wire format if new data needs to reach the Pi, and update the matching pane in `display.py`. If you change a Pi pane's layout, update this table (and the matching docstring table at the top of `display.py`) so it stays the source of truth for both sides.

The touch/mouse-friendly redesign (dropdown menus, etc.) is a later phase - for now the goal is this structural mirror: same pane names, same proportional layout, real data wherever the wire protocol already carries it.

## Setup on the Pi

```
sudo apt update && sudo apt full-upgrade -y
sudo rpi-eeprom-update -a && sudo reboot # required for Touch Display 2 auto-detect
sudo raspi-config # Interface Options -> Serial Port -> hardware: Yes, login shell: No
sudo apt install -y python3-pip python3-tk git python3-serial libportaudio2
pip install --break-system-packages websockets sounddevice # web remote + live audio (see requirements.txt)
python3 radio_ui.py
```

Then change `WEB_PASSKEY` in `config.py` and open `http://<pi-hostname>:8765/` from a browser on the same network for the web remote (LAN only - don't forward the port). Desktop icons to start the app and shut down the Pi are in `desktop/` (see `desktop/README.md`).

To run automatically on boot, see the header comment in `pilink-display.service` - it's now a **user** systemd service tied to the desktop session (with Desktop Autologin enabled), not a system service, since this app needs a graphical session to open its window.

## Files

- `display.py` — Tkinter-based windowed display: status bar, buttons, and a scrolling waterfall. Replaces `ra8875.py`.
- `pilink_client.py` — PiLink UART protocol client (mirrors `PiLink.h.cpp` on the Teensy exactly). Unchanged by the display swap - the wire protocol to the Teensy doesn't care what's drawing the UI.
- `radio_ui.py` — main app: ties telemetry -> window, button clicks/touches -> virtual button presses. Now queues PiLink events from its background thread and drains them on the Tk main thread, since Tkinter isn't thread-safe.
- `config.py` — UART port/baud and display geometry. No more SPI/I2C/GPIO pin numbers - the display needs none.
- `pilink-display.service` — user-level systemd unit to auto-start the app on boot inside the desktop session.
- `web_relay.py` — LAN-only WebSocket server started by `radio_ui.py`; mirrors the display to browsers at port 8765 (passkey-gated) and forwards their clicks through the same code the touchscreen uses.
- `web/remote.html` — the browser remote page served by `web_relay.py`.
- `audio_relay.py` — captures RX audio from the Teensy's USB audio device and hands it to `web_relay.py` for the browser (optional; turns itself off if the cable or `sounddevice` is missing).
- `desktop/` — "Start Radio UI" and "Shutdown Pi" desktop launchers and their scripts; one-time setup in `desktop/README.md`.

**Removed this revision:** `ra8875.py` (RA8875 SPI driver) and `ft5316_touch.py` (FT5316 I2C touch polling) - both only applied to the old RA8875-based panel. Touch now arrives as ordinary Tkinter button clicks via the desktop's own input handling.