
Mustang 2 Companion Display Changelog (Raspberry Pi + web remote)

Covers changes to the Raspberry Pi companion display (`PiDisplay/`: `display.py`, `radio_ui.py`, `pilink_client.py`, `web_relay.py`) and the browser-based web remote (`web/remote.html`, `web/remote_mobile.html`). **None of these require reflashing the Teensy** unless a note says otherwise — just re-sync `PiDisplay/` to the Pi and restart `radio_ui.py` (and/or hard-refresh the browser for a web-remote-only change). Teensy firmware changes (new PiLink wire messages/ids, changes to what the sketch sends) are tracked separately in the firmware changelog.

For the full, blow-by-blow UI-bug history of each web remote page individually, see `CHANGELOG_remote.md` (desktop/laptop page) and `CHANGELOG_remote_mobile.md` (phone page) — this document gives the project-level view across both the Pi's own screen and the browser.

2026-10-04 — CW decode fix; Vol/AGC removed from CW page — current

- `pilink_client.py`: accepts both CW_DECODE sizes (24-byte text-only from early V1.0.33 builds, 42-byte with page values); adds `page_ok`. `display.py/web` show “-” for page values when `page_ok` is false.
- `display.py`: Vol:/AGC: removed from the CW settings page.

2026-10-04 — CW settings page (needs firmware V1.0.33)

- `display.py`: page 1 Settings items tagged `settings_p1`; new `_build_cw_settings_page()` (tag `settings_cw`), `set_settings_page_cw()`, `set_cw_page()`; `set_settings()` fills the shared values (Vol/AGC slot, Decoder, Temp, Load) and switches pages on `decoder_on`, like `SetSettingsPanePage()`. `set_home_screen_mode()` restores the right page after the secondary-screen toggle.
- `pilink_client.py`: CW_DECODE_FMT extended (42 bytes). `radio_ui.py`: passes the page values to the display and the web relay.

2026-10-04 — Decoded CW line under the waterfall (needs firmware V1.0.33)

- `pilink_client.py`: MSG_CW_DECODE (0x0E) → `on_cw_decode`.
- `radio_ui.py`: `_handle_cw_decode()` uses the running character count to append only the new characters from the 15-character window to a 400-character line; relays it to the web remotes when it changes.
- `display.py`: `_build_cw_decode_pane()` / `set_cw_decode()` - “CW” label (green when locked), Courier text trimmed to fit (newest on the right), WPM on the right, in the old State of Health spot under the waterfall.
- `web_relay.py`: `broadcast_cw_decode()`, also sent to a browser when it logs in.

2026-10-04 — Brighter 3D waterfall

- `display.py`: `_WF3D_BACK_BRIGHT` 0.18 → 0.5 with a `t ** 1.6` fade (`_WF3D_FADE_POWER`); bottom colour stops now (40,130,255) / (60,170,255); new `_WF3D_BASE` = 0.05 minimum row height in `_wf3d_normalize()`.

2026-10-04 — Waterfall coloured with the T41's own colour index (needs firmware V1.0.32)

- `pilink_client.py`: MSG_WATERFALL (0x0D) → `on_waterfall`.
- `radio_ui.py`: `_handle_waterfall()` keeps the latest row, converted to the 0-255 bin scale that lands on exactly that gradient entry; `_handle_spectrum()` passes it (if under 0.5 s old) to the display and the web relay.
- `display.py`: `push_spectrum_row(bins, wf_colors=None)` colours the 2D waterfall from `wf_colors` when present (peak-held with the waterfall speed setting like the bins); trace and 3D waterfall unchanged.
- `web_relay.py`: spectrum messages carry an optional `wf` field.

2026-10-04 — Waterfall blue much brighter

- `display.py`: `_brighten_blue()` strengthened (gamma 0.4, green raised halfway toward full scaled by the blue level, a touch of red); pure blue is now #1f80ff. 3D waterfall bottom stops now (20,100,255) and (40,150,255).

2026-10-04 — Brighter waterfall blue; frequency readout hidden on calibration screens

- `display.py`: `_brighten_blue()` applied to the waterfall colour table (blue part only); 3D waterfall's bottom two colour stops brightened.
- `display.py`: new `_raise_fullscreen()`, called after the Freq Font digits are redrawn (and after the audio pane's tick labels are rebuilt), so a full-screen calibration/BIT/equalizer page stays on top. Fixes the frequency readout showing through the Transmit IQ calibration screen.

2026-10-04 — Transmit IQ, Frequency, Power and S meter level calibration (needs firmware

V1.0.30)

- New `cal_screens.py` (mixed into `Display`): full-screen Transmit IQ, Frequency and Power calibration screens laid out like the T41's, with tap buttons in place of the instructions column. Copy this new file to the Pi along with the others.
- `pilink_client.py`: parses `TXIQ_CAL` / `FREQ_CAL` / `POWER_CAL` (`parse_cal()`, one `on_cal(kind, data)` callback).
- `radio_ui.py`: `_handle_cal()` shows/hides the screens and relays them to the web remotes; `_CAL_ACTIONS` maps every button to the T41 button id or encoder tick (Exit = button 1, Stop = button 13, new in V1.0.30); arrow keys and the mouse wheel drive the screen's main values; web command `cal_action`.
- `display.py`: Menu > Calibration items S meter level / Frequency / Transmit IQ / Power enabled; `_fullscreen_active()` covers the new screens (toolbar hidden, frequency ruler suppressed, live spectrum fed to the Transmit IQ / Frequency panes).
- `web_relay.py`: `broadcast_cal()`.

2026-10-03 — PI Spectrum Fill starts Off at every boot (firmware V1.0.29 for the radio's own fill)

- `display.py` hides the spectrum fill bands at startup and marks Off in Display > PI Spectrum > PI Spectrum Fill.
- `radio_ui.py` no longer restores `spectrum_fill` from `~/.config/mustang2/pi_prefs.json` (the other saved Pi settings are still restored).
- `web_relay.py` starts with the PI fill Off, so web remotes logging in get Off until it's picked from the Display menu.

2026-10-03 — Layout: REC/SWR right, 3-row toolbar, Vol: to settings (no firmware change)

- **Pi screen layout:**
 - REC moved to the right edge of the screen, and the SWR/PWR readout moved right to sit just beside it.
 - Toolbar in three fixed rows. Row 1: Menu, Display, Audio, Mic, CW, Vol, Band, Mode, Demod. Row 2: Zoom, Filter, Noise, Decode, Home, Course, Fine. Row 3: Wheel:.
- **Vol: moved from the data line above the spectrum to the settings pane** (Pi: the free slot beside TX Atten; web/phone: settings grid). The data line's remaining 8 readouts are re-spaced. On the Pi the gaps are sized from typical values, so they look even in normal use.
- **After the Vol button, Up/Down arrows also change the volume** (web and Pi), not just Left/Right.

2026-10-03 — Audio / Mic / CW dropdowns; Vol becomes a button (no firmware change)

- **New Audio, Mic and CW toolbar dropdowns** (Pi, web, phone), next to Display. They are the T41's own **Audio Options**, **Microphone** and **CW Options** menus, copied item for item and in the same order (e.g. Audio: AGC Off/Long/Slow/Medium/Fast, Adjust Equalizers, Toggle Autonotch, the four Noise Reduction choices; Mic: Mic gain, USB drive, USB RX level; CW: WPM ... CW Manual Mode). Picks send the same menu selection as before (and jump straight there with firmware V1.0.28+). Those three categories are no longer listed under Menu.
- **Vol is now a plain button**. One click sets the radio's Volume knob back to audio volume (from AGC/Mic/STn if needed) and points the mouse wheel and Left/Right arrows at it. Mic gain, sidetone and AGC are reached through the new Mic / CW / Audio dropdowns.
- Pi toolbar: rows are now split by width (the top row runs further right, where there's room), and "Course Inc"/"Fine Inc" are shortened to "Course"/"Fine" like the web, so the second row stays clear of the SWR readout. Phone: dropdowns in the right-hand toolbar columns open toward the left so they stay on screen.
- Pi: `_OWN_DROPDOWN_CATEGORIES` / `_OWN_DROPDOWN_LABELS`, `_make_plain_button()`. Web: `buildMenu(..., {only})` for each, `buildVolButton()`.

2026-10-03 — Arrow keys for Filter/Volume/tuning (no firmware change)

- **Arrow keys on every version (Pi, web, phone with a keyboard):**
 - **Left/Right** turn whichever knob the wheel is set to (Filter Lo/Hi Cut or Volume, picked from those dropdowns), one step per press. With nothing picked they tune.
 - **Up/Down** always tune (Shift = fine tune).
 - In a Menu parameter box every arrow turns the Filter knob; on the equalizer screen Left/Right pick the band and Up/Down change its level.
 - Pi: new `display.bind_arrow_keys()` / `radio_ui._on_arrow_key()`. These are ignored while typing in the Network page's password box.

2026-10-03 — Nixie tubes redrawn (no firmware change)

- **Freq Font > Nixie redrawn** to look like real Nixie tubes, from the user's reference picture:
 - a tall glass tube with dark end caps and a warm halo;
 - a red-amber honeycomb mesh inside;
 - thin wire digits glowing yellow-orange, drawn as smooth strokes (wide and dim through thin and bright) rather than a font;
 - a full tube of its own for the decimal point, with a glowing dot at the bottom.
- Same shapes on the Pi (`freq_fonts.py`, Tk canvas) and the web (`drawNixieTube()`, canvas glow and gradients).

2026-10-03 — Smooth 3D waterfall, DSP: on data line, Freq Font styles (no firmware

change)

- **3D waterfall smoothed (Pi):** rows are now drawn as one image with a true colour gradient (Pillow), so the horizontal colour steps are gone. It looks like the web version, and it's faster (about 13 ms per frame vs about 20 ms with bands). It falls back to colour bands if Pillow (python3-pil) isn't installed.
- **"DSP:" (RX DSP gain, dB) moved to the data line** above the spectrum, between AGC: and Scale:, and removed from the settings pane (Pi and web). On the Pi the readouts are now spaced with equal gaps based on their widest values, so nothing overlaps with 9 items. On the phone the line wraps to two rows.
- **New Display > Freq Font:** Original / Nixie (orange digits in glass tubes with a faint yellow glow) / VFD (green seven-segment, unlit segments faintly visible) / Dot Matrix (blue 5x7 dots). Shared across the Pi and web remotes and remembered across restarts.
 - Pi: `new_freq_fonts.py`, `set_freq_font()`, `set_pi_freq_font()`.
 - Web: `drawFreqStyled()`/`renderFreq()` on a `#freqCanvas`.
 - Messages: `pi_freq_font_set` / `pi_freq_font`.

2026-10-02 — Menu highlights + Network (Wi-Fi) page (no firmware change)

- **Menus highlight the current / last-selected choice** wherever there are two or more options:
 - **Pi:** radio-style entries with a green dot and bold green text. That covers every toolbar dropdown (Band, Mode, Demod, Zoom, Noise, Course Inc, Fine Inc, Decode, Vol, Filter, Home), every Menu/Display group, and the Pi-only flyouts.
 - **Web:** a green check on Menu/Display items (`.submenu-item.selected`). The toolbar `<select>`s already pre-highlighted the current choice.
 - Driven by the radio's real state where it's known: STATUS for Band/Mode/Demod and the T41's current menu item, SETTINGS for Zoom/Noise/Inc/Decode/Vol. So changes made on the T41's front panel show too. The Pi-only settings (fill, 2D/3D, speed, brightness) follow their shared state, and brightness is now broadcast as well (`pi_brightness`).
- `display.py`: `_add_radio_entry()`, `mark_dropdown()`, `mark_menu_item()`, `set_brightness_marker()`, `_open_network_screen()`; `radio_ui.py` calls the marks from STATUS/SETTINGS; `web_relay.py` adds `broadcast_pi_brightness()`.
- **New Menu > Network page on the Pi screen** (`network_screen.py` + `wifi.py`, using NetworkManager's `nmcli`):
 - Shows the current SSID, 2.4/5 GHz band, signal and IP address.
 - Lists the visible networks: signal, band, LOCK/open, connected/saved.
 - Rescan, Connect, Forget and Close buttons.
 - A secured, unsaved network opens a password box with an on-screen keyboard (one-shot Shift, ?123 symbols, Show/Hide, Clear, backspace).
 - Scans and connects run on a worker thread, so the radio display never freezes. A failed new connection's half-made profile is removed.
 - Pi only: the web remotes don't get it, because switching networks from a browser on the current one would cut it off.
 - If Connect says "Not authorized", there's a one-time polkit rule in `PiDisplay/README.md` ("Network screen").

2026-10-02 — Toolbar dropdowns moved left (no firmware change)

- `display.py`: toolbar starts 8 toolbar-font spaces (about 32px) further left.

2026-10-02 — Display gets its own dropdown (no firmware change)

- **Display has its own toolbar dropdown**, between Menu and Vol, and is no longer a category under Menu. Its items are the same grouped list (Spectrum floor, scale, smoothing, decay, trace smoothing, Spectrum shadow, Spectrum fill, Brightness, PI Spectrum), now one level shallower. Picks still send Display's real primary index, so nothing changes on the Teensy.
- Pi: `OWN_DROPDOWN_CATEGORIES = ("Display",)`, plus a toolbar entry "Display" after "Menu". Web: `OWN_DROPDOWN_CATEGORIES`, and `buildMenu(container, {label, id, only})` builds both dropdowns. The `#menuWrap/#menuBtn/#menuPanel` styles became classes, so the two share styling, and opening one closes the other.

2026-10-02 — Display menu regrouped; 3D peaks blended (no firmware change)

- **Menu > Display regrouped** into sub-menus. The T41's own menu numbers are unchanged; this only changes how the menu is shown:
 - Spectrum floor > Auto spectrum floor / Manual spectrum floor / Spectrum floor
 - Spectrum scale, Spectrum smoothing, Spectrum decay, Spectrum trace smoothing
 - Spectrum shadow > On / Off
 - Spectrum fill > On / Off
 - Brightness > 25-100%
 - PI Spectrum > PI Spectrum Fill (On/Off) / PI Waterfall (2D/3D) / PI Waterfall Speed Defined once as `_MENU_LAYOUT` (`display.py`) / `MENU_LAYOUT` (web); every number in it is the item's real `DisplayOptions[]` index. Greyed-out items still work as before.
- **3D waterfall peaks blended:** the Pi now uses 12 colour bands (was 6), sampled from the same colour ramp as the web gradient. Each new row is 1-2-1 smoothed across neighbouring columns (valleys between nearby peaks fill in; peaks keep their height) and mixed 35% with the previous row (`_WF3D_TIME_BLEND`), so ridges look continuous. Row outlines are now dark navy instead of bright blue. The Pi takes about 20 ms per frame in testing.

2026-10-02 — Pi remembers its display settings (no firmware change)

- The Pi now **remembers its own display settings across restarts**: PI Waterfall (2D/3D), PI Waterfall Speed, PI Spectrum Fill and Brightness. They're saved to `~/.config/mustang2/pi_prefs.json` (written atomically) whenever one changes, from the Pi menu or any web remote, and re-applied when `radio_ui.py` starts. Web tabs pick them up on login. Delete the file to return to the defaults.
- `radio_ui.py`: `_load_pi_prefs()`, `_save_pi_prefs()`, `_apply_saved_pi_prefs()`; the four setters save after applying. Only `radio_ui.py` changed.

2026-10-02 — Waterfall speed setting (no firmware change)

- New **Menu > Display > “PI Waterfall Speed”** flyout: **Fast** (one waterfall row per spectrum frame, about 15 rows/s, as before), **Medium** (1 per 2 frames), **Slow** (1 per 4) and **Very slow** (1 per 8). It applies to both the 2D and 3D waterfalls. The frames in between are peak-held into the row, so a short signal is never skipped, and the spectrum trace above stays real-time.
- Shared state across the Pi and every web tab, like PI Waterfall / PI Spectrum Fill. It starts at Fast. Pi: `set_waterfall_speed()`, `_set_pi_waterfall_speed()`, web command `pi_waterfall_speed_set {divisor}`, broadcast `pi_waterfall_speed`.

2026-10-02 — 3D waterfall option (no firmware change)

- New **Menu > Display > “PI Waterfall”** flyout with **2D / 3D**. 3D shows the newest spectrum row in front, with older rows stepping back (narrower and higher) and darkening until they fade out at the back. Blue foothills rise to yellow/orange/red peaks, and there's a dashed centre marker. Each row's noise floor is subtracted so signals stand up from a flat baseline.
- Shared state, like “PI Spectrum Fill”: picking it on the Pi or any web tab switches all of them, and a newly opened tab starts in the current mode. The default is 2D. The 2D waterfall keeps scrolling underneath, so switching back shows current history.
- **display.py**: `_PI_WATERFALL_OPTIONS`, a “PI Waterfall” flyout appended last in the Display list (never shifts a real `DisplayOptions[]` index), `_wf3d_normalize()`, `_build_waterfall_3d()`, `set_waterfall_3d()`, `_redraw_waterfall_3d()`. Tuning constants are `WF3D_*` at the top of the file (rows, columns, bands, depth, inward scale, height, gain, fade). Tkinter has no gradients, so each row is 6 band-clamped polygons.
- **radio_ui.py**: `_set_pi_waterfall_3d()` (single source of truth), web command `pi_waterfall_3d_set`.
- **web_relay.py**: `broadcast_pi_waterfall_3d()`, plus the current mode is sent on login.

V1.0.28 (2026-10-01) — Menu picks jump straight to the item

Uses firmware **Mustang 2 V1.0.28**'s new `PILINK_MSG_MENU_SELECT` (0x14).

- **pilink_client.py**: `MSG_MENU_SELECT = 0x14`, `send_menu_select(primary, secondary)`.
- **radio_ui.py**: `_select_menu_item()` sends `HOME_SCREEN` then a (“menu”, p, s) step (new `_run_sequence` atom) instead of `HOME`, `MAIN_MENU_UP` and one paced Filter tick per item. It's gated on `self.fw_version >= (1, 0, 28)`, parsed from the Settings message's `VERSION` string, so older firmware keeps the old path. Applies to the Pi Menu dropdown and the web/phone `menu_select` command alike.

V1.0.27 (2026-10-01) — Equalizer: Exit button, Edit RX / Edit TX buttons

Requires Teensy firmware **Mustang 2 V1.0.27** for “Exit” (older firmware ignores button 14 on this screen).

- **display.py**: the single “Editing: (switch)” toggle is replaced by “Edit RX” / “Edit TX” buttons (the side being edited is lit green and reads “Editing ...”). They use `on_eq_select_side`, the same path as tapping a pane, so they pick a side directly instead of toggling. New “Exit” button (yellow) next to “Save & Exit” via new `on_eq_exit_nosave`. Instructions text updated.
- **radio_ui.py**: new `_eq_exit_nosave()` (sends `BUTTON_ID_DFE`, 14) and web command `eq_exit_nosave`.
- **Web remotes**: same Edit RX / Edit TX and Exit buttons (see the remote changelogs).

V1.0.26 (2026-10-01) — Audio Equalizer screen (firmware V1.0.26); “TI:” renamed “CT:”

Requires Teensy firmware **Mustang 2 V1.0.26** (new PiLink message 0x09).

- **pilink_client.py**: new downlink `MSG_EQUALIZER = 0x09`, 32-byte payload `EQUALIZER_FMT = "<BBBB14B14B"` (active, rxtxSelection, cellSelection, increment, 14 RX levels, 14 TX levels). Delivered via new `on_equalizer` callback as {active, editing_tx, cell, increment, rx, tx}.
- **radio_ui.py**: `_handle_equalizer` plus `_eq_toggle_side` (button 15), `_eq_increment` (button 16), `_eq_cell` (Volume encoder), `_eq_level` (Filter encoder), `_eq_select_cell` (shortest-path Volume ticks), `_eq_exit` (Home = save & exit). New web commands `eq_select_cell {index}` and `eq_select_side {tx}`. Mouse wheel adjusts the selected cell's level while the screen is up.
- **display.py**: full-screen `_build_equalizer_pane` / `set_equalizer` mirroring the T41 screen — Receive and Transmit panes with 14 bars each (198 Hz – 4 kHz), selected cell in green, edited side in blue, readout, increment, and a controls column (Editing RX/TX, Cell ◀▶, Level – +, Incr, Save & Exit). Tap a bar to select it; tap the other pane to switch RX/TX. Toolbar hidden while active.
- **web_relay.py**: `broadcast_equalizer()` forwards {"type": "equalizer"}.
- **Web remotes**: same equalizer overlay on /remote and /mobile.
- Readout strip label **“TI:” renamed “CT:”**.

V1.0.25 (2026-10-01)

Requires reflashing the Teensy (new `PILINK_MSG_BIT_RESULTS` downlink message - see `CHANGELOG_firmware.md`). - **I2C BIT test results screen ported to the Pi and both web remotes** (Menu > Diagnostics > “I2C BIT test”), at the user’s request (“the I2C BIT Test needs to be ported over the the PI and both remotes”). Mirrors the Teensy’s own `DrawBIT()` - one PASS/FAIL line per board (front panel, BPF, RF, RF SI5351, K9HZ LPF), plus a “Home” button (the only control that screen’s own `Loop.cpp` switch handles - no save step, unlike Receive IQ’s Save & Exit). `pilink_client.py` parses the new `PILINK_MSG_BIT_RESULTS` frame; `radio_ui.py`’s `_handle_bit_results()` feeds `display.py`’s new `set_bit_results()` (a full-screen opaque backdrop pane, same pattern as the Receive IQ screen) and `web_relay.py`’s new `broadcast_bit_results()` (a plain `{“type”: “bit_results”, “data”: ...}` pass-through) for both web remotes. The frequency-ruler bleed-through guard added for the Receive IQ screen (`set_spectrum_freq_scale()`) now also covers this screen, for the same reason - it’s the same full-screen-backdrop-raised-above-everything shape. - **Pi-only follow-up fix (no reflash): title/rows moved down**, after the user reported the screen rendering “too far up and off the page to see anything but the bottom of the last entry showing under the full menu dropdowns” on the real Pi (both web remotes were fine). Root cause is the same one `set_spectrum_freq_scale()`’s bleed-through fix already diagnosed for the frequency ruler: the Pi’s two-row toolbar (Menu/Vol/Band/.../Home) is built from real `tk.Menubutton` widgets embedded into the canvas via `create_window()`, which Tkinter always renders above plain canvas-drawn items regardless of `tag_raise()` - and the toolbar’s own height is a FIXED pixel constant (`_TOOLBAR_BTN_H/_TOOLBAR_ROW_GAP`), not scaled the way this pane’s content is, so on this user’s actual screen proportions the original title (y=40)/first row (y=100) left too little headroom. Moved the title to y=70, the first row to y=150 (row spacing 36->38), and the Home button to y=360 - all comfortably within the 0-480 logical screen with wide margin either side, matching the clearance the Receive IQ pane’s own real content already uses successfully. - **Pi-only follow-up fix #2 (no reflash): toolbar now hidden instead of worked around**, after the user reported the same issue persisting: “Same issue, need to remove the dropdown menus as the page has a ‘Home’ button. But everything is jammed at the top overwriting itself.” Repositioning the BIT pane’s own content (the fix above) wasn’t the real fix - the toolbar’s `create_window()`-embedded widgets render above canvas content no matter where that content sits, so the toolbar itself needed to go away, not the pane’s layout. Every toolbar button’s canvas item id is now captured (`_toolbar_window_item_ids`, `_build_vfo_pane()`), and a new `_set_toolbar_visible()` helper hides all of them (`state="hidden"`) while the I2C BIT test screen is active - its own “Home” button makes the toolbar redundant there anyway. The Receive IQ calibration screen’s toolbar is now hidden the same way for consistency, since it has the identical overlap issue in principle. The BIT screen’s title/row/Home-button y-positions from the previous fix are unchanged (confirmed correct by the user: “The Title ... is fine where its at”). - **Pi-only follow-up fix #3 (no reflash): all 5 board rows were overwriting each other at the top-left**, after the user reported “All the data labels and data are all jammed at the top left, overwriting themselves. The I2C Status Report label and the Home button are good.” Root cause: `_build_bit_pane()`’s per-row loop computes each row’s y with a local `scaled(x0, y0, w, h)` helper that returns `(x0*sx, y0*sy, w*sx, h*sy)`, but unpacked it as `ry, _, _, _ = scaled(0, row_y0, 0, 0)` - grabbing the FIRST element (`0*sx == 0`, the x component) instead of the SECOND (`row_y0*sy`, the actual scaled y). Every row’s text item was created at y=0 regardless of `row_y0`, so all 5 rows landed exactly on top of each other at the very top of the pane. Fixed to unpack the y from the correct slot (`_, ry, _, _ = scaled(...)`); the title and Home button were unaffected since they already used the full 4-tuple correctly. - **Several menu items greyed out and made non-selectable**, at the same request (“On the Calibration menu grey out all but the Receive IQ. On the EEPROM menu grey out Print Data to Serial... If the menu item is greyed out it should not be selectable for now”), on the Pi and both web remotes - the front panel is unaffected and keeps every item fully selectable: Calibration > S meter level/Frequency/Transmit IQ/Power (only Receive IQ stays enabled), Diagnostics > Restore receive EQ/Buffer print, EEPROM > Print Data to Serial. `display.py`’s menu dropdown now checks a `_MENU_DISABLED_ITEMS` set before building each leaf (`state="disabled"` on the Tk menu item skips the command AND renders it grey); both web remotes do the same with their own `MENU_DISABLED_ITEMS` set and a new `.submenu-item.disabled` CSS class (no click handler attached at all, so nothing can be sent). Add/remove entries here as more Pi/web screens get ported - “I2C BIT test” itself was removed from this list by this same change.

Unreleased

- **Receive IQ calibration screen: “Stop”/“Exit” buttons + instructions added to both web remotes**, at the user’s request to bring them to parity with the Pi’s own screen (“On Web and Phone remote calibration screens need to add the ‘Stop’ and ‘Exit’ buttons same as the PI screen” → “Add instructions at bottom”). Both buttons send the same repurposed virtual-button ids the Pi sends (`DECODER_TOGGLE=13` for Stop, `DPE=14` for Exit) via the existing generic `{“cmd”:“button”, “id”:N}` shape - no backend changes needed. A new `#rxiqCalInstructions` block below the table mirrors the Pi screen’s instructions text. **No reflash needed** if the Teensy is already on V1.0.24 (these buttons’ firmware-side handling shipped then); re-sync `PiDisplay/` and hard-refresh both browsers.
- **Receive IQ calibration screen (Pi only): frequency-ruler bleed-through fixed, instructions font enlarged, Adjust/Table panes grown**, follow-up to the layout fix + Stop/Exit below, at the user’s request after trying it again (“Delete the band freq numbers at the bottom of the spectrum display. Increase the font size of the instructions. Increase the Current Band and Results panes height adding to the bottom so the GEN results are not cutoff at the bottom of the pane. Keeping both panes equal”). **No reflash needed** - Pi-only, `display.py`.
 - **Frequency-ruler bleed-through:** `set_spectrum_freq_scale()` (the Home spectrum pane’s tick/label ruler, fed at 10Hz from STATUS) deletes and recreates its canvas items on every call with no tag, so freshly-created items always land on top of the stacking order - including on top of this screen’s own full-screen opaque backdrop, which can only cover items that already existed when it was raised. The ruler was bleeding through on top of the Adjust/Table panes below it. Fixed by having `set_spectrum_freq_scale()` clear its items and skip drawing entirely while `self._rxiq_cal_active`, resuming normally the moment the screen goes inactive.
 - **Instructions font:** 11pt → 14pt.
 - **Adjust/Table panes grown 230px → 255px tall, added to the bottom** (y0 unchanged, kept equal to each other): the real cause of “GEN cutoff” wasn’t the screen edge (the previous round’s upward shift already cleared that) but the Table box’s OWN bottom border - the GEN row (last of 13) landed only 6px above it. The extra 25px gives it real clearance.
- **Receive IQ calibration screen (Pi only): layout fix + Stop/Exit buttons**, follow-up to the screen added just below, at the user’s request after trying it (“Move label and data ‘Delta:’ down one line and over to the left edge. Move the spectrum pane, current band and results pane up 2 lines, as the GEN line is cutoff on the bottom... Need to add a ‘Stop’ button to stop the calibration when in Auto Mode if possible. And an ‘Exit’ button, keep the ‘Save & Exit’ button, it can mimic the Home - Home Screen dropdown”). **Requires reflashing the Teensy to V1.0.24** (new `Loop.cpp` button cases - see `CHANGELOG_firmware.md`) in addition to re-syncing `PiDisplay/`. Scoped to the Pi’s own screen only, not the web remotes, per the request.
 - `display.py`’s `_build_rxiq_cal_pane()`: Delta: label/value moved to the left edge and down one line (now sits just inside the

- spectrum pane's own top-left corner); the spectrum pane, Adjust box ("Current Band"), and the 13-row all-bands table all moved up ~2 lines so the table's last row (GEN) now ends well clear of the screen's bottom edge instead of landing right on it.
- **New "Stop" button:** aborts an in-progress Auto sweep early (see `CHANGELOG_firmware.md`'s `RequestRXIQAutoStop()` entry for why the full sweep can look hung on GEN, the last band). Sends the `DECODER_TOGGLE` virtual-button id (repurposed only on this screen via a new `BUTTON_ID_DECODER_TOGGLE` reuse in `radio_ui.py`'s `_rxiq_cal_stop()`).
- **New "Exit" button:** leaves the screen without forcing a save, distinct from the existing "Save & Exit". Sends a new `BUTTON_ID_DFE` virtual-button id (repurposed only on this screen) via `radio_ui.py`'s new `_rxiq_cal_exit_nosave()`.
- Control column's button rows shrunk (48px → 36px tall) to fit the two new rows (Amp/Phase/Incr/Auto/Band/Stop/Exit/Save & Exit, 8 rows total) in the same column height as before.
- **Receive IQ calibration screen added to the Pi and both web remotes** (Menu > Calibration > Receive IQ), at the user's request ("Need to add the Calibration Screen from T41 to the PI. Just need to copy/adjust/fit to the PI screen" → "Lets start with Receive IQ and once thats done we'll move to the other 3" → "Yes the PI needs tpo control calibrataion the same as the T41 would"). **Requires reflashing the Teensy to V1.0.23** (new `PILINK_MSG_RXIQ_CAL` telemetry message - see `CHANGELOG_firmware.md`) in addition to re-syncing `PiDisplay/`.
 - **Full remote control, not just a mirror** - Amp/Phase adjust, step-size (Increment) toggle, Auto-tune, Band up/down, and Save & Exit all work from the Pi's touchscreen and both browser remotes, exactly as they would from the T41's own knobs/buttons. No new uplink protocol was needed: every control sends the same `PILINK_MSG_VIRTUAL_ENCODER/` `PILINK_MSG_VIRTUAL_BUTTON` messages already used for menu navigation and the Vol/BW/Band/Mode/Demod dropdowns (Volume encoder → Amp, Filter encoder → Phase, BEARING button → toggle 0.01/0.001 step, SPARE button → auto-tune, `BAND_UP/BAND_DN` → cycle band, `HOME_SCREEN` → save & exit).
 - **display.py:** new full-screen pane (`_build_rxiq_cal_pane()`, shown/hidden via the existing `_tag_new_items()` tagging pattern) with a Delta readout, a bordered Adjust box (Band/Amp/Phase/Increment), a bordered 13-row all-bands table (Band/Amp/Phase/Delta), a live two-sideband spectrum trace (reusing the spectrum bins already flowing through `push_spectrum_row()` - no new wire message needed for this part either), and tappable `Amp±/Phase±/Incrment/Auto/Band±/Save&Exit` buttons. `radio_ui.py` wires these to the existing `_send_button()/` `_send_encoder_tick()` helpers via six small new callback methods.
 - **pilink_client.py:** new `MSG_RXIQ_CAL/RXIQ_CAL_FMT` struct unpacking `PiLinkRXIQCalMsg` into `{active, current_band, increment_index, amp[], phase[], delta[]}`.
 - **web_relay.py:** new `broadcast_rxiq_cal()` pass-through, same shape as `broadcast_settings()/broadcast_parameter()`.
 - **Both web remotes** gained a matching full-screen overlay (band/amp/ phase/increment/delta readout, 13-row table, and the same six controls), sending the existing generic `{"cmd":"button","id":N}/ {"cmd":"encoder","encoder":...,"increase":...}` shapes - no backend command changes were needed on the web side either.
 - **Deliberately left out of both web remotes:** the live two-sideband spectrum mini-plot the Pi's own screen shows (`PaneSpectrum` on the T41) - a scope reduction for this first pass; everything else (readouts, table, full control) is at parity with the Pi.
 - **Deferred to future passes,** per the user's own sequencing: Frequency, Transmit IQ, and Power calibration screens.
- **Both web remotes gained the SWR/PWR readout and fixed a missing "(USB)" demod suffix,** after the user reported "Both remotes web and phone are missing the demod reading like the PI screen 20M / DATA (USB) and also the SWR 1.0 PWR 0.0 label/readings."
 - **SWR/PWR readout was never built on either web remote** - the data (`status.swr/status.forward_power_w`) was flowing over the WebSocket the whole time (part of every "status" broadcast), just never rendered. New `#swrText` element in `statusBlock`, right under REC/XMT (same relative position as the Pi's own screen, where the SWR/PWR pane sits flush against `txrx_status`), updated in `onStatus()` with the same SWR `{swr:.1f}` PWR `{forward_power_w:.1f}` W format and red-on-TX/white-on-RX coloring as `display.py`'s `set_swr()`. `remote_mobile.html`'s version shrunk to 12px font to fit its row-layout `statusBlock` on a narrow phone screen.
 - **"(USB)" demod suffix bug:** both pages' `MODULATION_NAMES` table had 0: null instead of 0: "USB" - present since each page was first built, so modulation 0 (USB, a very common default) always silently dropped its suffix, showing e.g. "20M / DATA" instead of "20M / DATA (USB)" while every other modulation value displayed correctly. `display.py`'s own `_MODULATION_NAMES` had 0: "USB" right from the start - only these two separately-maintained JS copies had the wrong value. Fixed to 0: "USB" in both files.
 - **No reflash needed** - purely a web-remote change. Requires only a hard-refresh of any open browser remote (no Pi restart needed either, since this doesn't touch `radio_ui.py/display.py/web_relay.py`). `remote_mobile.html` bumped to V1.0.30; `remote.html`'s version display mirrors the Teensy's own telemetry, so no separate bump there.
- **New independent "PI Spectrum Fill" toggle for the Pi's own screen and both web remotes** (Menu > Display > "PI Spectrum Fill", On/Off) - supersedes the mirroring entry directly below this one. The user tried the mirrored behavior and asked for it to be tunable separately instead: "Need to be able to tune it on/off separately. T41 On/Off and PI/Remotes On/Off." The T41's own Menu > Display > "Spectrum fill on/off" setting (firmware-side, unaffected by this change) now controls only the T41's own screen; this new toggle controls only the Pi's own canvas + both browser remotes, shared/broadcast among those three but independent of whatever the T41 itself is set to. Firmware V1.0.22's `PiLink spectrumFillTraceEnabled` field is still sent and still parsed by `pilink_client.py` (no firmware change, no reflash needed for this), it is simply no longer consumed to drive Pi/web rendering.
 - `display.py:` `T41_MENU`'s "Display" category gained a second Pi-only sentinel, "PI Spectrum Fill", appended after "Brightness" (same pattern - a Pi-only submenu entry with no firmware counterpart, added at the very end so it doesn't disturb the real entries' index values). New `on_pi_spectrum_fill_select` callback param threaded through `_init/_build_vfo_pane()`, building a 3rd-level flyout (`_PI_SPECTRUM_FILL_OPTIONS`, On/Off) the same way "Brightness" does. `set_settings()` no longer calls `_set_spectrum_fill_trace_enabled()` from the T41-sourced `PiLink` field - that method is now Pi-local-only, called exclusively from `radio_ui.py`'s new `_set_pi_spectrum_fill_enabled()`.
 - `radio_ui.py:` new `_set_pi_spectrum_fill_enabled(enabled)` is the single source of truth for this shared toggle - writes to the Pi's own canvas (`display.set_spectrum_fill_trace_enabled()`) and broadcasts to every connected browser (`web_relay.broadcast_pi_spectrum_fill()`) in one call. Wired to the Pi's own touchscreen flyout via `on_pi_spectrum_fill_select=` and to a new "pi_spectrum_fill_set" web command in `_handle_web_command()`. `_MENU_REAL_LENGTH_OVERRIDE`'s comment updated to note it now has two trailing Pi-only sentinels ("Brightness", "PI Spectrum Fill") after the real "Display" entries - the override

value itself (11) is unchanged, since only entries trailing after the real ones were added.

- `web_relay.py`: new cached `self.pi_spectrum_fill_enabled` (default `True`, unlike other broadcast state this needs caching so a newly-connecting browser tab gets the current value immediately, since there's no periodic PiLink re-send to eventually catch it up the way T41-sourced settings get). Sent right after `login_ok` as a `{"type": "pi_spectrum_fill", "enabled": ...}` message; new `broadcast_pi_spectrum_fill(enabled)` updates the cache and broadcasts the same message to every open tab.
- `web/remote.html/web/remote_mobile.html`: same "PI Spectrum Fill" 3rd-level flyout added to `buildMenu()` (mirroring the "Brightness" flyout pattern), sending `{cmd: "pi_spectrum_fill_set", enabled}`. New `state.piSpectrumFillEnabled` (default `true`) and `onPiSpectrumFill(enabled)` handler for the "pi_spectrum_fill" message type. `drawSpectrumStatic()`'s fill gate changed from reading `state.settings.spectrum_fill_trace_enabled` to `state.piSpectrumFillEnabled` - the outline (`drawTrace()`) is unaffected either way. `remote_mobile.html` bumped to V1.0.29; `remote.html`'s version display mirrors the Teensy's own telemetry, so no separate bump there.
- **No reflash needed** - purely a `PiDisplay//web-remote` change. Requires re-syncing `PiDisplay/` to the Pi, restarting `radio_ui.py`, and a hard-refresh of any open browser remote.
- **Pi's own spectrum pane and both web remotes now mirror the T41's Menu > Display > "Spectrum fill on/off" setting** instead of always drawing filled - at the user's request, for visual consistency with the radio (the choppy-audio bug this setting fixes was already confirmed to be T41-side only, unrelated to this page's own rendering). **Superseded by the independent-toggle entry above** - kept here for history; the PiLink field this relied on is still sent by firmware but no longer drives Pi/web rendering.
 - `display.py`: new `_set_spectrum_fill_trace_enabled()` method hides/shows the pre-built fill-band polygons (`self._spec_fill_bands`) rather than recreating them; `push_spectrum_row()` skips repositioning them while hidden (no added per-frame cost). Called from `set_settings()`. The trace line itself (`self._spec_trace_id`) is unaffected and stays visible either way, same as the T41's own outline mode.
 - `web/remote.html/web/remote_mobile.html`: `drawSpectrumStatic()` now reads `state.settings.spectrum_fill_trace_enabled` (defaulting to `true`) and skips `drawTraceFill()` when it's false; the outline (`drawTrace()`) is unaffected. `remote_mobile.html` bumped to V1.0.28.
 - `pilink_client.py`: `SETTINGS_FMT` gained a 23rd field (`spectrum_fill_trace_enabled`, `uint8`) matching firmware V1.0.22's new `PiLinkSettingsMsg.spectrumFillTraceEnabled` - 58 bytes total, up from 57. Same "whole Settings pane goes stale until reflashed" caveat as every prior field addition to this struct.
 - `web_relay.py` needed no change - `broadcast_settings()` forwards the whole parsed settings dict as-is, so the new field reaches the browser automatically once `pilink_client.py` parses it.
 - **Requires reflashing the Teensy to V1.0.22** (for the new PiLink field) **and re-syncing PiDisplay/ to the Pi, restarting radio_ui.py, and hard-refreshing any open browser remote.**
- **T41_MENU's "Display" category** (`display.py`, mirrored in `radio_ui.py`'s `_MENU_REAL_LENGTH_OVERRIDE`, `web/remote.html`, `web/remote_mobile.html`) **gained "Spectrum fill on"/"Spectrum fill off"**, matching firmware V1.0.21's new `DisplayOptions[]` entries (`ED.spectrumFillTraceEnabled`, the fix for the choppy-audio root cause - see `CHANGELOG_firmware.md`). All four Pi/web copies of this menu list are hand-maintained mirrors of the real Teensy menu, so a firmware-side addition always needs the matching label(s) added here too, in the same order, plus `_MENU_REAL_LENGTH_OVERRIDE`'s "Display" count bumped (9 -> 11) so the Filter-encoder wraparound math (`radio_ui.py`'s `_select_menu_item()`) still lines up correctly around the Pi-only "Brightness" sentinel entry appended after it. Needed for the menu option to actually show up on the Pi's own screen and both web remotes after reflashing V1.0.21 - it was already visible on the T41's own screen (firmware draws directly from its own `DisplayOptions[]`), just missing from these separately-maintained copies. `web/remote_mobile.html` bumped to V1.0.27 for this; `web/remote.html`'s version display mirrors the Teensy's own telemetry, so no separate bump there. **Requires re-syncing PiDisplay/ to the Pi and restarting radio_ui.py, plus a hard-refresh of any open browser remote** (no reflash needed - the firmware side already shipped in V1.0.21).
- **WebRelay (web_relay.py) audio-delivery timing diagnostics added**, after `latency="high"` (below) shipped and the user confirmed NO xrun lines printed while audio was still cutting out - ruling out the capture side entirely and pointing at delivery instead (the asyncio broadcast thread, or something further downstream). Two new counters: `broadcast_audio_pcm()` now times the gap between successive calls from `AudioCapture`'s own callback thread (should be ~40ms; a bigger gap means something delayed that thread's own scheduling, broader than an xrun) and logs `WebRelay: audio callback gap #N: NNms past 80ms`; a new `_broadcast_audio_async()` times how long a chunk sits queued on the asyncio thread before `ws.send()` actually runs, logging `WebRelay: audio send delay #N: NNms past 80ms` - if THIS climbs while the callback gap stays near 40ms, the delay is on the broadcast/asyncio side, most likely this process's Tkinter GUI work (different thread, same GIL) starving the asyncio thread of turns. Both rate-limited the same way `audio_relay.py`'s xrun counter is. **Requires restarting radio_ui.py.**
- **AudioCapture (audio_relay.py) opened with latency="high", plus xrun logging**, at the user's follow-up report that audio was still "cutting out very rapidly" after the playback-side sample-rate fix below - which pointed away from the browser entirely, since neither client-side fix touched it. `PortAudio`'s default (unset) latency uses the smallest internal capture buffer the driver allows; this process also runs a Tkinter GUI on the same Python GIL, and if drawing work ever holds the GIL long enough that `AudioCapture`'s callback can't run before that small buffer empties, `PortAudio` reports an input overflow/underrun and audio drops out - rapidly and repeatedly if the contention is frequent. `latency="high"` asks for a bigger, more forgiving buffer (a bit more capture-side delay, on top of the browser's own `AUDIO_LEAD_S`, in exchange for a lot more slack). `_callback()` now also counts and prints status flags (previously received but silently ignored) - rate-limited console output (`AudioCapture: xrun #N (...)`) so a next occurrence actually confirms or rules out this cause instead of guessing again. **Requires restarting radio_ui.py** (this is server-side, not a browser refresh).
- **Web remote choppy-audio bug fixed** - on `AudioPcm()` in both `remote.html/remote_mobile.html` was creating each chunk's `AudioBuffer` labeled with `audioCtx.sampleRate` instead of the PCM's actual, fixed capture rate (`AUDIO_SAMPLE_RATE`, 44.1kHz - matches `config.py`'s `AUDIO_SAMPLE_RATE` on the Pi). Browsers are free to ignore the `{sampleRate: 44100}` requested when the `AudioContext` is created and hand back their own hardware-native rate instead (48kHz is common, especially on phones), and when that happens every chunk was being mislabeled - playing pitch/speed-shifted, and (more audibly) causing `audioNextPlayTime`'s schedule to drift out of sync with real chunk arrival timing, needing periodic resyncs (audible glitches). Found via the user's report that RAISING the jitter buffer (`AUDIO_LEAD_S`, see below) made it "even worst" - the wrong direction for a jitter problem, right direction for a drift problem (a bigger buffer just let more drift accumulate before each resync). `AudioBufferSourceNode` resamples an `AudioBuffer` to the context's real rate automatically on playback, so labeling it with its true source rate fixes this regardless of what rate the context actually landed on.
- **Web remote audio jitter buffer raised 0.15s → 0.4s** - tried first, at the user's initial report ("Sound on remotes is choppy"), on the

(wrong) theory that WiFi jitter was outrunning the buffer. Made it worse, which is what pointed at the real bug above. Reverted back to 0.15s - a jitter buffer that size is plenty once the schedule itself isn't drifting.

- **Scale: nudged 2 character-spaces left** on the New Line 2 strip (`_build_newline2_pane()`), at the user's request ("Only on the Pi Display move the Scale: readout 2 character spaces to the left"). Applied as a pixel offset (`font.measure("00")`) on top of the existing even-spacing frac, rather than changing the frac itself. **Pi's own screen only** — `remote.html/remote_mobile.html` unchanged.
- **Spectrum pane height reduced; New Line 2 strip moved above the spectrum's top border**, at the user's request: "reduce the spectrum's pane height, there is a lot of wasted space as the signals never reach that high. Then move the readouts that are currently in the spectrum display to above the top line for the spectrum display."
 - **Pi (display.py)**: `TEENSY_PANE_RECTS["spectrum"]` moved from (5, 95, 520, 345) to (5, 117, 520, 323) - top pushed down 22px, height trimmed by the same 22px so the bottom stays fixed against `state_of_health` below. This is now a deliberate Pi-rendering-only deviation from the Teensy's real `PaneSpectrum` rect (unchanged in hardware). `_build_spectrum_pane()`'s trace-box fraction cut from 150/345 (43.5% of the pane) to 90/323 (~27.9%) - the trace box was taking far more height than the live signal ever used; the freed space goes to the waterfall. `_build_newline2_pane()` moved from `y0=92` (3px *inside* the old spectrum border, overlapping the trace box) to `y0=93`, now sitting in the 24px gap between `freq_band_mod`'s bottom (90) and the new spectrum top (117) - clearly above the border instead of drawn over it.
 - **Web (remote.html/remote_mobile.html)**: the Vol:/AGC:/Scale:/Floor:/Zoom:/TI:/FT:/Noise: strip moved OUT of the spectrum canvas entirely, from a `drawNewLine2()` canvas draw into a new `#newline2Row` DOM element (plain flexbox row, `justify-content: space-between`) positioned above `#mainRow` in the page flow - `#mainRow` is `flex:1` so it automatically gives up this row's height to `#specCanvas`. Updated via the new `updateNewLine2()` (DOM text, called from `onSettings()`) rather than redrawn every spectrum frame. `specGeometry()`'s `traceH` share of the remaining canvas height cut from 40% to 25%, for the same "wasted space" reason as the Pi side - freed height goes to the waterfall. Mobile's strip uses an 11px font (its own `#newline2Row` CSS) to match the page's already-reduced spectrum-readout size; desktop's stays 14px.
- **New Line 2 strip switched to plain even column spacing** (`_build_newline2_pane()`), at the user's explicit request ("adjust the spacing for all readouts to even the gaps between the readouts") - replaces the 3 content-weighted passes (each giving wider columns like Scale:/Floor: more room than short ones) with simple, equal 1/8 increments:
`0.000/0.125/0.250/0.375/0.500/0.625/0.750/0.875`. **Pi's own screen only** - the request specifically said "In the Pi", so `remote.html/remote_mobile.html` keep the previous content-weighted fractions (`0.00/0.09/0.23/0.44/0.58/0.68/0.78/0.89`) unchanged. If a wide value (e.g. Scale:'s "12.5dB") ever overlaps its neighbor again under even spacing, that's the even-gaps/content-width trade-off this request explicitly chose.
- **Menu > Display > Brightness combined into one flyout submenu** (`display.py`'s `_build_vfo_pane()`, `remote.html/remote_mobile.html`'s `buildMenu()`), at the user's same-day follow-up request. The 4 flat "Brightness 25%"/"50%"/"75%"/"100%" entries became a single "Brightness" entry under Display, rendered as its own nested cascade (`_BRIGHTNESS_PRESETS/BRIGHTNESS_PRESETS`, [25, 50, 75, 100]) - same Tk `add_cascade` mechanism the top-level categories already use, and the same `.menu-item/.submenu/.submenu-open` CSS reused a level deeper on the web side (with clearing scoped to the immediate submenu's children so opening the flyout doesn't also close its parent "Display" flyout). Picking a preset now calls `on_brightness_select(pct)` directly (Pi) or sends a new "brightness_set" command (web, handled in `radio_ui.py`'s `handle_web_command()`) - neither goes through `on_menu_item_select()/select_menu_item()` at all anymore, since "Brightness" is a single index slot again, not 4. `_T41_MENU`'s Display list is back down to 10 entries (9 real + 1 "Brightness" sentinel); `_MENU_REAL_LENGTH_OVERRIDE["Display"]` stays at 9 (the real firmware count) either way.
- **Menu > Display > Brightness added** (`display.py`'s `_T41_MENU` gained "Brightness 25%"/"50%"/"75%"/"100%" under the Display category; `remote.html/remote_mobile.html`'s `_T41_MENU` match), at the user's request, to control the Raspberry Pi Touch Display 2's own screen backlight — entirely Pi-side, no Teensy firmware involved. Writes `/sys/class/backlight/*/brightness` directly (glob'd rather than a hardcoded driver name). `radio_ui.py`'s `select_menu_item()` detects the "Brightness " prefix (`_PI_ONLY_MENU_PREFIX`) and short-circuits before sending any of the usual HOME_SCREEN/MAIN_MENU_UP/encoder/SELECT sequence to the Teensy — this is the first `_T41_MENU` entry with no firmware counterpart at all. `_MENU_REAL_LENGTH_OVERRIDE` gained a "Display": 9 entry (the real, firmware `DisplayOptions[9]` size) so the 4 new Pi-only entries don't throw off wraparound math for the real Display items around them. If the backlight write fails with a permission error (printed once to the terminal), add the `pilink-display.service` user to the `video` group and reboot/re-login — see the method's own comment for why that should normally already work out of the box on stock Raspberry Pi OS.
- **"Scale:" moved right, centered between AGC: and Floor:** on the New Line 2 strip (`_build_newline2_pane()/drawNewLine2()`, all three: Pi, `remote.html`, `remote_mobile.html`), at the user's follow-up request - the previous overlap fix (below) resolved the collision but left Scale: crowded right up against AGC: with a lot of unused space before Floor:. Column fractions recomputed a 3rd time, still budgeting the full "Scale: XX.XdB" width, borrowing the extra room from Zoom:/TI:/ FT:/Noise: (all short values):
`0.00/0.09/0.23/0.44/0.58/0.68/0.78/0.89`.
- **"Scale:"/"Floor:" columns no longer overlap** on the New Line 2 strip — Scale:'s widest value ("12.5dB") was overwriting part of Floor:'s label, reported right after Floor: shipped. Column fractions in `_build_newline2_pane()/drawNewLine2()` (all three: Pi, `remote.html`, `remote_mobile.html`) recomputed by actual character-count budget per column (label + widest possible value) instead of even spacing, giving Scale: and Floor: — the two widest columns — noticeably more room each.
- **"Floor:" reading added to the New Line 2 strip** (`display.py`'s `_build_newline2_pane()/update_newline2()`, `remote.html/remote_mobile.html`'s `drawNewLine2()`), right after Scale:, at the user's follow-up request. Shows "Auto" while `s.spectrum_floor_auto/spectrum_floor_auto` is set (Menu > Display > "Auto spectrum floor"), otherwise the signed manual `spectrum_floor/s.spectrum_floor` value (a raw screen-pixel baseline shift, -100..100 — not dB, unlike Scale:). Column fractions recomputed again, now for 8 columns. **Needs a Teensy reflash** — see `CHANGELOG_firmware.md` V1.0.18 for the new `spectrumFloor/spectrumFloorAuto` `PiLinkSettingsMsg` fields this depends on; until reflashed, the whole Settings pane/New Line 2 strip stays stale, same as Scale: before its own reflash.
- **Menu > RF Options > RX DSP Gain added to the Pi/web remote menu dropdowns** (`display.py`'s `_T41_MENU`, `remote.html/remote_mobile.html`'s `_T41_MENU`), mirroring the newly-un-hidden firmware menu entry (see `CHANGELOG_firmware.md` V1.0.17). Also fixed a wraparound-modulus bug this would otherwise have caused: the real firmware `RFSet[]` array is declared with a 6th, blank/unused slot and always wraps navigation modulo 6, so a 5-item Python list alone would send the Filter encoder to the wrong step count when navigating past the new item. Fixed with `radio_ui.py`'s new `_MENU_REAL_LENGTH_OVERRIDE = {"RF Options": 6}`, used only for the wraparound math — the visible dropdown still lists just the 5 real items. **Needs a Teensy reflash before the menu item does**

- **anything on the real radio** — re-syncing `PiDisplay/` alone only updates what the dropdown shows, not the firmware it's driving.
- **Pi-screen layout pass** (`display.py`), four separate requests bundled together:
 - **Pi's LAN IP address now shown in front of the clock** in the name/badge pane (top-right corner), e.g. `192.168.1.42 14:07`. New `_get_local_ip()` helper (no external dependencies — opens a UDP socket and reads the OS-assigned local address off it, sending no actual packets) is looked up once at startup and re-checked every 10th clock tick (~10s) rather than every second. Falls back to showing just the time if the Pi has no network route yet (e.g. Wi-Fi still associating at boot). Meant to make it quick to find the address to type into a phone/PC browser for the web remote, without SSHing in for `hostname -I`.
 - **SWR/PWR text moved next to REC/XMT**, at the user's report that it looked "far" from it. It was already inside its own narrow pane, just left-anchored (`anchor="w"`), which left a wide strip of blank pane background between the short text and the adjacent `txrx_status` pane. Switched to right-anchored (`anchor="e"`), which puts it flush against that pane's left edge — i.e. right next to REC/XMT. **Pi's own screen only** — the web remotes weren't part of this request and are unchanged.
 - **Audio Filter pane's bottom "0k/1k/.../6k" tick labels enlarged**, 8pt → 11pt, on the **Pi's own screen and the desktop/laptop web remote** (`remote.html`) **only** — the user confirmed they're fine as-is on the phone page, so `remote_mobile.html` is deliberately untouched. Still comfortably inside the 30px bottom margin already reserved below the pane's content box.
 - **Shutdown password prompt**: no shutdown/poweroff functionality exists anywhere in this codebase (`display.py`, `radio_ui.py`, `config.py`) — that prompt comes from the Pi's own OS (`sudo/raspi-config`), so there's nothing to change here. Left as a note for the user to handle directly on the Pi (e.g. a `NOPASSWD` sudoers rule scoped to `/sbin/shutdown//sbin/poweroff`, or `raspi-config`'s boot options).
- **Spectrum/waterfall color pass, at the user's request that the T41's own colors are "much brighter" than the Pi's, plus two related fixes**. Applied identically to `display.py`, `remote.html`, and `remote_mobile.html` unless noted:
 - **Spectrum trace/fill green brightened from #00ff66 to pure #00ff00**. The waterfall's own gradient LUT was already ported verbatim from the Teensy and correctly bright; this cyan-tinted green (the live trace line, its fill, and the frequency-scale ruler's near-center tick color) had drifted from the Teensy's actual `SPECTRUM_FILL_COLOR` (`RA8875_GREEN`, `RGB565 0x07E0` — pure (0,255,0)). Now matched exactly.
 - **Audio-filter (passband) highlight in the spectrum display changed from a dim navy (#000080) to a lighter blue (#3399ff)**, at the user's explicit request. Same hue, just brighter.
 - **"REC" on the Pi's own screen changed from white to green (#00cc44)** — the web remotes already did this on 2026-09-26; the Pi's own `display.py` (`_build_txrx_pane/set_txrx_status`) was never updated to match until now. XMT stays red on all three.
 - **Green→yellow→red spectrum-fill transition softened**, at the user's report that it "shows up as a very distinct line" rather than fading. The transition's pure-yellow anchor point (`SPECTRUM_COLOR_BREAK2`, matched to the waterfall's own LUT) is unchanged; a new `_SPECTRUM_FADE/SPECTRUM_FADE` constant (4/116) widens where the green fade starts and the red fade ends by a small, symmetric margin, and the blend within each zone now uses a smoothstep ease (zero slope at both ends) instead of a straight linear ramp — on the Pi this also moves where the solid-green base polygon hands off to the graded strips; on the web remotes (true canvas gradients, previously a sharp corner where a flat `addColorStop` met a linear one) a new `addSmoothFade()` helper adds several eased intermediate stops per transition to approximate the same curve.
- **Added `test_audio_cable.py`, a stand-alone diagnostic for the Teensy USB-audio cable** (run directly on the Pi, independent of `radio_ui.py/web_relay.py`): lists every audio device PortAudio sees, locates the one matching `config.AUDIO_DEVICE_NAME_HINT`, records a short sample, reports its peak/RMS level in dBFS (flagging near-silence or clipping), and saves a `.wav` so the recording can be copied off and actually listened to. Written at the user's request, now that a temporary Teensy-to-Pi USB cable is in place, to verify the cable/USB Audio build works before relying on the full app to tell them so. No changes to `audio_relay.py/web_relay.py` the web remote pages — the RX audio → browser pipeline (`audio_relay.py` capture, `web_relay.py` broadcast, `remote.html/remote_mobile.html` playback) already existed from 2026-09-26 and needed no code changes, only this new way to confirm the physical cable is actually carrying signal. **Pi-only** — copy `test_audio_cable.py` to the Pi alongside the rest of `PiDisplay/`.
- **"Power:" reading changed from "20 W" to "20W"** (no space before the unit), at the user's request to "move the W over" - on the Pi's own Settings pane (`display.py`), and applied consistently to both web remote pages too (`remote.html`, `remote_mobile.html` - see their own changelogs). Cosmetic only.
- **Web remote (mobile page) only: Audio Filter pane's 1kHz-tick ruler labels were too small and getting clipped at the bottom; S-meter's dBm readout was clipped on its right edge**. Both fixed - font/spacing tweaks and a wider right margin respectively. See `CHANGELOG_remote_mobile.md` (V1.0.15/V1.0.16) for the detail; nothing changed on the Pi's own screen or the desktop web page for either of these two.
- **Settings pane: tightened row spacing, T Temp(F)/PI Temp(F) reworked, both temps now warn/flash at high readings**. Row spacing tightened ~12%. "Temp(F):" (T41 CPU temp) relabeled **"T Temp(F):"** ("T" for Teensy - first landed as "T Temp:(F)", corrected same-day to match "PI Temp(F):"'s punctuation) and moved onto the same row as "PI Temp(F):", directly across from it (Power/Load now share the freed row). Both values turn yellow + blink at 85% of the part's own max operating temp, red + still blinking at 95%, back to solid green below that — thresholds researched against each part's real published limit: T41's NXP i.MX RT1062 (industrial) Tj max 105°C = 221°F (≈188.9°F/210.0°F breakpoints); this Pi 5's Broadcom BCM2712 throttle ceiling 85°C = 185°F, per Raspberry Pi's own thermal write-up (≈157.25°F/175.75°F breakpoints). **Pi-only** — re-sync `PiDisplay/` and restart `radio_ui.py`. No firmware reflash needed.
- **BW dropdown button on the Pi's own screen relabeled "Filter"** (was "BW") — brings it in line with the web remote, whose own BW dropdown has always shown "Filter" as its fixed label. Cosmetic only; the control's wiring is unchanged. **Pi-only** — re-sync `PiDisplay/` and restart `radio_ui.py`.
- **Investigated a report that selecting "Data" from the Mode dropdown lands on SSB instead**. Reading `Loop.cpp`'s `TOGGLE_MODE` handler: the SSB→CW→DATA→SSB three-way cycle only exists `#ifdef AUDIO_INTERFACE`; without it, `CW_RECEIVE` dispatches straight back to SSB and DATA is never reached at all — so selecting Data (2 presses from SSB) lands exactly on SSB (SSB→CW→SSB), matching the report precisely. `AUDIO_INTERFACE` is a Teensyduino-core macro set by the Arduino IDE's **Tools > USB Type** menu (needs "Serial + MIDI + Audio" — the same build option `audio_relay.py`'s live RX audio already depends on), not something in the checked-in source, so this isn't a `radio_ui.py/` firmware code bug — it means this particular reflash was built with a USB Type that doesn't include the audio interface. No source change made; re-select that Tools > USB Type option and reflash to make DATA reachable again.

- **Band selection now takes the shorter direction (BAND_UP or BAND_DN) instead of always cycling forward.** Investigated at the user's report that Zoom/Band selections "go through all the levels" one direction only. Reading the real firmware handler (`Loop.cpp`'s `HandleButtonPress()`) directly: Zoom (and Mode/Demod/Noise/Course Inc/ Fine Inc/Vol/BW) is genuinely a single-direction physical button with no "down" counterpart, so a virtual button press can't skip the intermediate levels — going 8x back to 1x via 16x really is the minimum number of presses in the only direction that exists. Band is different: `Loop.cpp` has a real `BAND_DN` alongside `BAND_UP`, so `_select_band()` now picks whichever direction takes fewer presses — band 0 to band 11 used to take 11 forward presses; now it's 1 press of `BAND_DN`. **Pi-only** — re-sync `PiDisplay/` and restart `radio_ui.py`. No firmware reflash needed. (The separate *visible pacing* between each of those presses — the actual "why does it have to go through all the levels" complaint — is fixed below in V1.0.16.)

V1.0.16 (2026-09-27)

- **Cycling through several presses of the same button (Zoom, Band, Mode, Demod, Noise, Course Inc, Fine Inc, Vol, BW) now happens near-instantly instead of visibly stepping through each intermediate level ~300ms apart.** This is the fix for the user's actual complaint — going from 1x to 8x zoom still passes through 2x/4x internally (Zoom has no "down" button, so intermediate levels can't be skipped — see the Unreleased Band note above), but it no longer happens as a slow, visible march; all the presses needed land together in one go. Root cause: `radio_ui.py`'s `_cycle_button_to()` sent one virtual-button press at a time, deliberately paced 300ms apart (`_run_sequence()/PRESS_SPACING_MS`) — pacing that exists to stop a *different* button id from clobbering the Teensy's single-slot button mailbox before the main loop drains the previous press, but pointless overhead when every press in a cycle-to-target run uses the *same* button id (no clobbering risk). Fixed by adding a new PiLink uplink message, `PILINK_MSG_VIRTUAL_BUTTON_REPEAT` (payload: button id + press count), that the Teensy queues back-to-back in one shot; `_cycle_button_to()` now calls `pilink_client.send_virtual_button_repeat()` once per selection instead of pacing N individual presses. `_select_menu_item()`'s own multi-button/encoder sequence (which mixes different ids) is untouched and still uses the old paced path, since the clobbering risk is real there. **Requires reflashing the Teensy with the updated `PiLink.h/PiLink.cpp`, reflashed together with re-syncing this updated `PiDisplay/` — `radio_ui.py` no longer has a fallback to the old one-press-at-a-time path, so Zoom/Band/Mode/Demod/ Noise/Course Inc/Fine Inc/Vol/BW selection will do nothing on an un-reflashed Teensy until both sides are updated together.** See `CHANGELOG_firmware.md` for the firmware-side detail.
- **Spectrum trace's fill-under-the-curve color now matches the waterfall's own color at the same signal level, on all three displays (the Pi's own screen, `remote.html`, and `remote_mobile.html`)** — the fade from green to yellow to red used to break at a fixed 1/3 and 1/2 of the pane's height; now it breaks at the exact amplitude fractions (32/116, 38/116, 64/116) where the waterfall's own gradient table actually shows those colors. Landed on the Pi's own screen first, then matched to both web remote pages the same day at the user's follow-up request. See `CHANGELOG_remote.md/CHANGELOG_remote_mobile.md` for the web-remote-specific detail (each page's own `SPECTRUM_COLOR_BREAK1/2/3` constants and `drawTraceFill()` gradient stops). **Pi-only** — re-sync `PiDisplay/` and restart `radio_ui.py`; hard-refresh both web remote browsers. No firmware reflash needed.
- **Vol:/AGC:/Zoom:/TI:/FT:/Noise: moved from the Settings pane into an always-visible strip across the inside top of the spectrum display**, on both the Pi's own screen and the web remote — no longer gated to the Teensy's secondary (5-pane) home screen.
- **Retired the state_of_health pane's "Power - W | Filter - Hz | PI Temp(F): -.-" status line**, moving each field to a better home: Filter now shows in the Audio Filter pane's own corner; Power/PI Temp moved into the Settings pane (Power also added to the web remote's Settings pane for parity).
- Bumped the web remote's Vol/AGC/Zoom/TI/FT/Noise strip and Filter: readout to match the Settings pane's own font size, on both displays.
- Dropped the dropdown indicator (the small dash/arrow after the label) from every toolbar dropdown on the Pi's own screen — Tk now centers each label on its own.
- Added touch jog sliders to the web remote for Tune/Fine Tune/Volume/ Filter (phone/tablet support, before the desktop/mobile page split).
- Reclaimed vertical room on phones (Wheel indicator hidden, `#statusBlock` reorder, 4-column toolbar grid) so the jog sliders weren't crowded against the bottom of the screen.

Web remote UI history (2026-09-26)

The web remote went through an extended run of phone-usability fixes this session — version-labeled "Mustang Mobile V1.0.1" through "V1.0.13" on the phone page, several applied to both pages. In order: split into a dedicated `remote_mobile.html` page (separate from the laptop page), `100vh-vs-100dvh` viewport fix, version-label collision + slider touch-action fix, dropped the redundant Teensy version readout + added a bottom safe-area spacer, fixed every toolbar dropdown showing its current value instead of a fixed centered label, fixed S-meter tick-label clipping, flattened a fragile `display: contents` toolbar grid that broke on the real phone, fixed REC being clipped/faded at the top (top safe-area inset, then a guaranteed flat padding) and set it to green, fixed the frequency-scale ruler running into the waterfall (fixed-pixel ruler band instead of a proportional one), and then fixed the waterfall disappearing as a side effect of that last fix (split the leftover canvas height between the trace and the waterfall instead of handing it all to the trace). Full details, in order, are in `CHANGELOG_remote_mobile.md` and `CHANGELOG_remote.md`.

V1.0.14 (2026-09-25)

Added (Pi-only, no reflash needed)

- **LAN-only browser remote control (mouse + USB rotary encoder):** new `PiDisplay/web_relay.py` runs a small WebSocket server on the Pi (default port 8765) that re-broadcasts the exact telemetry `radio_ui.py` already parses from PiLink to any connected browser, and forwards browser button/encoder commands to the same `_select_*()/send_virtual_button()/send_virtual_encoder()`

code path a touch on the Pi's own screen uses. Gated by a passkey (`config.WEB_PASSKEY`). **LAN-only by design** — plain `ws://`, no TLS.

- **Web remote: Audio Spectrum pane, Settings pane, and a real cascading Menu** — mirrors the Audio Spectrum pane (live magenta audio-FFT bars plus passband/CW-filter markers) and the Settings pane; the Menu control rebuilt from a flat grouped `<select>` into a nested/cascading popup much closer to the Pi's own Tkinter cascade widget.
- **Web remote: BW (Filter) dropdown, wheel-target fix, and a wheel deselect control** — new “BW” dropdown (Hi Cut/Low Cut); fixed the mouse wheel being hardcoded to always emulate Volume; picking from either the Vol or BW dropdown now “claims” the wheel for that knob, with a clickable “Wheel: Tune only / Volume / Filter” indicator on both the Pi's own screen and the web page to release it.
- **Live RX audio on the web remote, plus a Mute button** — new `PiDisplay/audio_relay.py` captures 44.1kHz mono PCM from the Teensy's USB audio device (via `sounddevice/PortAudio`) and streams it to every logged-in browser as raw binary WebSocket frames. A Mute button is entirely local to that browser tab. (Needs the Teensy's native USB port wired to a Pi USB-A port, and the sketch built with Arduino IDE's “Tools > USB Type > Serial + MIDI + Audio” — a build-option change, not a code reflash.)
- **Blue filter-bandwidth highlight on the Spectrum pane**, on both the Pi's screen and the web remote — a dim navy rect spanning the current receive passband, reusing the same `passband_low_hz/passband_high_hz` telemetry already flowing for the Audio Spectrum pane's markers.
- **Desktop icons to start `radio_ui.py` and shut down the Pi** — “Start Radio UI” refuses to start a second copy if one's already running; “Shutdown Pi” asks for confirmation before running `sudo shutdown -h now` (needs a one-time sudoers rule scoped to just that command).

Fixed (Pi-only, no reflash needed)

- **Remote Menu selection (Pi and web both) landed back on Home instead of sticking on the picked setting's adjustment screen.** Root cause: `radio_ui.py`'s shadow copy of the Teensy's Menu position (`_menu_primary_shadow/_menu_secondary_shadow`) started the session assumed at (0, 0) and could never self-correct once badly drifted. Hand-recalibrating the shadow's starting values to the Teensy's actual live position immediately fixed remote Menu selection on both the Pi and the web remote — a one-time snapshot, not a structural fix (see the firmware changelog for the permanent telemetry-based fix).
- **PiLinkSettingsMsg wire-size mismatch** (see the firmware changelog for the full story) — the Pi-side fix: reverted `pilink_client.py`'s `SETTINGS_FMT` back to the 53-byte, 19-field format the real hardware actually sends (`"<BhBIihBhBhbBBhB24sBBB"`), since a one-time diagnostic print against the live Teensy showed the checked-in `PiLink.h/.cpp` source don't match what's actually flashed.

V1.0.12 (2026-09-25)

Added (Pi display only, no reflash needed)

- **Select on the Menu parameter-adjustment overlay** — the red-boxed “Label: value” box now has a tap-able “Select >” affordance that sends the real front panel's SELECT button to confirm the value and return to Home, mirroring the physical action.
- **Pi mirrors the Teensy's secondary (“Toggle Layout”) home screen** — hides State of Health/SWR/Audio Spectrum/Settings (no equivalent on that screen), shows a new “New Line 2” status strip (Vol/AGC/Zoom/TI/ FT/Noise), and moves the S-meter/TX-RX panes up into the freed space, driven entirely by telemetry so it follows a screen switch made at the front panel too.

V1.0.10 (2026-09-25)

Changed (Pi display only, no reflash needed)

- Toolbar reordered and rebalanced into two fixed rows of 6 and 5, by count rather than by available width: Menu, Vol, Band, Mode, Demod, Course Inc / Fine Inc, Zoom, Noise, Decode, Home.

V1.0.8 (2026-09-25)

Changed (Pi display only, no reflash needed)

- “PI Temp:” label now reads “PI Temp(F):”.
- All toolbar dropdown buttons sized to fit their own label/content, later made 10% larger for easier touch targets.
- Menu now opens as a single button with real cascading flyout submenus instead of a second dropdown button appearing to its right.

V1.0.7 (2026-09-25)

Changed (Pi display only, no reflash needed)

- TX/RX pane relabeled “REC”/“XMT” (was “RX”/“TX”) to match the Teensy's own on-screen wording exactly.
- Removed the pane border around the SWR/PWR readout and the S-meter, matching the real Teensy, which borders neither.
- Settings pane: Vol/AGC top row enlarged to 20pt bold; all other fields and the version string enlarged to 14pt; labels now white, live values now green.
- Spectrum trace is now filled under the curve, graded green→yellow→red by height, matching the Teensy's own `FillSpectrumColumnGraded()/SpectrumAmplitudeColor()` instead of an unfilled line.
- Waterfall now uses the Teensy's own 117-entry color table (`gradient[]`) ported value-for-value, instead of a simplified green-only ramp.

V1.0.6 (2026-09-25)

Changed (Pi display only, no reflash needed)

- Moved the BAND –/BAND+ buttons from the Settings pane to the right of the frequency readout on the VFO pane.
- Fullscreen kiosk mode now reveals the Pi OS's own top panel when the mouse is moved to the top edge of the screen, without leaving fullscreen — Escape still exits fullscreen entirely.

Fixed (Pi-only, no reflash needed)

- Audio-filter/audio-spectrum pane's bottom margin was overlapping the top of the Settings pane directly below it; the pane now reserves the same 30-of-150px bottom margin fraction the Teensy's own pane does.

V1.0.4 (2026-09-25)

Fixed (Pi-only, no reflash needed)

- Waterfall was scrolling bottom-to-top instead of top-to-bottom — newest data now enters at the top and scrolls down, matching the Teensy's own `BTE_move()` direction.

V1.0.2 and earlier

The S-meter pane rebuilt to match the real T41's S-meter gauge (scale, tick labels, two-tone track), the SWR/PWR pane reverted to plain text after a bar-gauge version didn't match the real hardware, and an earlier fullscreen fix — all predate this changelog's start and aren't individually dated.