Probe Basic — Sidebar User Tab Not Rendering Content

Root cause and working fix for the sidebar-tab loading issue discussed in “Probe Basic - HAL pins not being created” (thread 59149)

Summary

The 'user side-bar tab' feature loads and registers correctly (confirmed in the startup log), but its content never becomes visible on screen — not even plain QLabels, only HAL widgets were being tested but the same failure affects any content. The 'user main' tab, built with an identical Python wrapper, works fine. Root cause: the wrapper widget that probe_basic hands to each container never gets its own layout or size, so it reports no usable size to whatever container places it. The main-tab container happens to force-fill its page regardless, masking the bug; the sidebar container apparently does not, and collapses the tab to nothing.

Symptoms Observed

- HAL pins for widgets placed in the sidebar tab (e.g. HalLedIndicator) appear correctly in halshow.
- The sidebar tab is correctly selected when its button is clicked (confirmed by watching the stack page switch).
- No content renders in the sidebar tab area at all — not the HAL widgets, not plain QLabels.
- No errors, warnings, or tracebacks appear in the startup log related to the sidebar tab.
- The identical setup (HAL widgets + plain labels) in the main window user tab renders correctly.

Diagnostic Log Evidence

The startup log (LOG_LEVEL = DEBUG) shows both tabs loading and registering successfully, with no errors between them:

```
[user_tabs] added 'template_main' as main tab 'user main'
[user_tabs] added 'template_sidebar' to the sidebar as 'user sb'
```

This rules out file path, naming, qrc, and permissions issues — the tab is genuinely instantiated and added to its container. The failure is purely in rendering/sizing after that point.

Root Cause

Both `template_main.py` and `template_sidebar.py` use an identical `UserTab` wrapper:

```
class UserTab(QWidget):
    def __init__(self, parent=None):
        super(UserTab, self).__init__(parent)
        ui_file = os.path.splitext(os.path.basename(__file__))[0] + ".ui"
        ui_path = os.path.join(os.path.dirname(__file__), ui_file)
        self.ui = _load_ui(ui_path, self)
        _adopt_ui_identity(self, self.ui)
```

`_load_ui()` uses PySide6's QUiLoader, which (per the existing code comment) nests the loaded UI as a *separate child widget* rather than making the given parent the root widget — unlike PyQt's `uic.loadUi(path, baseinstance)`. `_adopt_ui_identity()` copies the objectName and dynamic properties (including the 'sidebar' flag probe_basic reads for placement) from the loaded UI onto the wrapper — but it does **not** copy the actual size/geometry. So:

- self.ui (the real content) has a real, applied geometry from the .ui file (e.g. 179×511 for the sidebar template).
- self (the wrapper widget that probe_basic actually adds to its container) has no layout and no explicit size — Qt has nothing to derive a size hint from.

The main-tab container appears to force-resize whichever page widget it's given to fill the full page area regardless of that widget's own size hint, so the missing layout goes unnoticed there. The sidebar container most likely lays out its tab(s) based on each widget's own reported size (reasonable, since it may need to stack multiple sidebar panels in a narrow column) — and gets nothing usable from the empty wrapper, so the tab collapses to zero visible content.

Fix

Give the wrapper widget a real layout and propagate the loaded UI's actual size onto it, in `template_sidebar.py` (config-folder file, no permissions issues):

```
from PySide6.QtWidgets import QWidget, QVBoxLayout   # add QVBoxLayout

class UserTab(QWidget):
    def __init__(self, parent=None):
        super(UserTab, self).__init__(parent)
        ui_file = os.path.splitext(os.path.basename(__file__))[0] + ".ui"
        ui_path = os.path.join(os.path.dirname(__file__), ui_file)
        self.ui = _load_ui(ui_path, self)
        _adopt_ui_identity(self, self.ui)

        # Propagate real size and wrap self.ui in a layout so the
        # container has something concrete to lay out.
        self.setMinimumSize(self.ui.minimumSize())
        self.setMaximumSize(self.ui.maximumSize())
        self.resize(self.ui.size())

        layout = QVBoxLayout(self)
        layout.setContentsMargins(0, 0, 0, 0)
        layout.addWidget(self.ui)
```

Result: confirmed working — all three sidebar HAL LED indicators and their labels (TEST1/TEST2/TEST3) now render correctly alongside the sidebar's MAN/AUTO/MDI controls.

Suggested Next Step

Since `template_main.py` has the same latent gap (it just happens not to matter for that container), the same two-line size-propagation + layout change is worth applying there too, and ideally folding into the shipped templates so future users of the sidebar user-tab feature don't hit this. Happy to share the full patched config folder if useful for testing against the actual sidebar container code in probe_basic.py.

Reference: forum thread “Probe Basic - HAL pins not being created”, forum.linuxcnc.org/qtpyvcp/59149