aiwa pb-3 battery replacement

Table of Contents

USB-C rechargeable LiPo replacement for the AIWA PB-3/PB-4 battery pack.

1. Specs

Parameter	Original PB-3	This Design
Chemistry	Lead-acid	LiPo
Cell voltage	2V	3.7V nominal
Output voltage	2V	2.0V (adjustable 2.0-2.4V)
Capacity	470mAh	500-1500mAh
Charging	Proprietary	USB-C 5V

2. Safety

- Buck DC-DC steps 3.7V → 2.4V
- **OVP (Zener/TVS)** clamps output <3V if buck fails
- DW01A+FS8205A BMS: OV/UV/OC protection

3. Reverse Engineering Notes

3.1. PB-3 Reverse-Engineering Notebook

3.1.1. Objective

Build a present-day PB-3 pack that wakes an AIWA HS-PC202MII. That deck is the primary test bed and end user. The wall wart is missing, the stock SLA is dead, and donor packs only exist as forum photos, so every measurement and assumption comes from images plus clone teardowns.

3.1.2. What is Known Today

Topic	Details	Confidence
Mechanical envelope	Caliper photos show 68 mm × 17.8 mm × 8 mm overall and a 52 mm × 15 mm × 6 mm cell bay. Thin plastic shell, PCB island at the connector end with fuse and ballast resistor.[1]	High. Need one more sample to average tolerance.
Electrical interface	Two-pin keyed blade plus PB-4 top pads. Clone pack shows 3.67 V LiPo feeding a converter that outputs 2.41 V on the blades; wiring solders directly to the IC pins.[s2g] HS-PC202MII schematic is unavailable, so reverse engineering relies on photos and continuity tests once the clone arrives.	Low. Need deck-side schematic to confirm charge routine and pinout.

Topic	Details	Confidence
Chemistry and capacity	Original PB-3 spec sheet rated 470 mAh SLA. Community teardowns hint at 700–800 mAh service packs, but clones today ship 500 mAh LiPo with buck down to 2.4 V.[s2g]	Medium. Need an unopened OEM cell for proof.
Community precedents	Baidu and Stereo2Go builds use 3D-printed shells plus Li-ion or LiPo stages, usually without fault clamps or plated blades.[s2g]	High. Reuse photos only with permission.

3.1.3. Research Findings

3.1.3.1. Stereo2Go thread highlights

- AliExpress PB-3/PB-4 USB-C pack runs £14, lists 900 mAh, hides 500 mAh LiPo, spits 3.67 V raw and 2.41 V at the blades.[2]
- Shell size is 68 mm × 17.8 mm × 8 mm. Battery cavity maxes at 52 mm × 15 mm × 6 mm, so 500 mAh is the ceiling until cells improve.[2]
- 3D printed case flexes, screw bosses die fast, only top contacts get plating.[2]
- Pack leads land right on converter pins. Bare copper hovers over inductors. No strain relief.[2]
- If the buck converter fails, the deck eats the full 3.7 V. Need a clamp or crowbar that kills the fault.[2]
- PB-S5 board is the same. Swap the marked 100 kΩ resistor for 120 kΩ to hit 2.1 V. 0603 footprint.[2]
- Search phrase "500mAh lipo battery 6mm" finds cells that fit; nothing larger ships yet.[2]
- Wickedkraft sells a pricier variant on eBay. No autopsy yet but likely the same guts.[2]

3.1.3.2. Derived Requirements

- USB-C charging. Retire the fragile Aiwa wall wart.[2]
- Runtime must beat the 470 mAh SLA. Target >900 mAh effective at 2V output.[2]
- Respect 68 mm × 17.8 mm × 8 mm overall and 52 mm × 15 mm × 6 mm cell cavity. Keep PB-4 top contacts alive.[2]
- Plate every blade. Reinforce the connector window so owners stop snapping plastic.[2]
- Terminate the cell on pads with relief. No bare wires surfing active parts.[2]
- Add a fault clamp so output never drifts past 2.4 V even if the buck fails. Make PB-S5 vs PB-3 voltage a resistor change.[2]

3.1.4. Host Deck Reality

- Runner: AIWA HS-PC202MII. This deck will see every prototype first and is the reason the project exists.

- Constraints: OEM wall wart is missing, OEM PB-3 pack is dead, and voltage measurements of it are not relevant (0v). Reverse engineering starts with images of clone PCBs.

3.1.5. Research To-Do List

1. **Imagery**: pull clean teardown photos from Baidu Tieba (□□□ PB-3) and Japanese blogs. Capture rulers or redraw.
2. **Connector survey**: borrow a PB-3 pack or host deck. Log pitch, depth, retention, plating.
3. **Power sweep**: record float voltage, discharge curve, deck charge current while the donor SLA still lives (if there is one which there isn't).
4. **Archive**: dump every measurement, photo, and trace into this repo before CAD starts.

3.1.6. Proposed Replacement Strategy

SLA is dead weight. Teardown of Aliexpress clone confirms the working approach.

LiPo 1S (3.7V nominal) + buck to 2.0V

+ Matches existing clone design (LAOWANG_DIY board) + 500mAh cell → 900mAh effective at 2V output + Buck topology (step-down) matches 3.7V→2.0V requirement + BMS often on cell flex PCB

Baseline design: **1S LiPo 500mAh + buck converter regulated at 2.0V** matching the original PB-3 output.

3.1.7. Electrical Block Requirements

- Cell: 500mAh LiPo pouch, ~6mm thick, matches 52×15×6mm cavity
- BMS: On cell flex PCB (clone) or discrete DW01A+FS8205 (our design)
- Buck: SOT-23-6 IC, 2.2µH inductor, configured for 2.0V output
- Rfb: 100k = 2.4V, 120k = 2.1V (per Stereo2Go mod). Adjust for 2.0V target.
- Charge path: USB-C 5V → charger IC (SOT-23-6) → CC/CV 4.2V to cell
- OVP: Zener/TVS clamp <3V to protect deck if buck fails (improvement over clone)

3.1.8. Mechanical Considerations

- Stay inside the original outline so the bay latch still works. 0.8 mm FR-4 maximizes cell room.
- Scan or caliper a donor shell. Build a STEP model for interference checks.
- Reuse the original keyed blade if possible. If not, design a flex tail or custom edge connector footprint.
- Rough CAD now lives in `openscad/pb3-pack.scad` (OpenSCAD). Dimensions trace back to `measurements/IMG_9373.jpeg–IMG_9385.jpeg` plus the Stereo2Go thread captured in `research/`.

First Solid CAD Model

```
// PB-3 battery pack simplification: single solid block matching the outer envelope.
```

```
// Dimensions drawn from measurement photos (measurements/IMG_9373.jpeg…IMG_9385.jpeg)
// and the Stereo2Go teardown cached in research/.
// Units: millimetres.

module pb3_pack_fitting_test() {
    outer_len = 68;
    outer_width = 17.8;
    outer_height = 8;
    textH=.5; textS=10;
    wiggle=.01;
    difference(){
        cube([outer_len, outer_width, outer_height]);
        translate([textS/2, outer_width/2,outer_height-textH])
linear_extrude(.5+wiggle) text("v0.1",size=textS,valign="center");
    }
}

pb3_pack_fitting_test();
// To test fitting, run OpenSCAD and export to STL. Then 3D print a test block to
verify fit in the PB-3 housing.
```

[pb3 pack] | *openscad/pb3-pack.png*

Figure 1. output image

3.1.9. Living CAD + Automation

- `openscad/pb3-pack.scad` is the print gauge. Version is embossed on the block so prints always tell you which assumption set they follow.
- `kicad/pb3-replacement.kicad_pro` (plus `.kicad_sch`/`.kicad_pcb`) tracks the electrical build. It is tagged V0.1 in the title blocks so KiCad exports line up with the physical gauges.
- `.github/workflows/cad-build.yml` is the single entry workflow. Jobs: `openscad`, `plantuml`, `kicad` → `publish` which copies artifacts into `docs/generated/` and commits.

3.1.10. Reverse-Engineering Workflow

1. De-pot a PB-3 with heat and patience. Photograph each layer.
2. Trace the PCB. Mark every component, resistor, and fuse.
3. Recreate the old schematic in KiCad. Understand what the deck expects before changing it.
4. Map new requirements against the host behavior.
5. Draft hierarchical KiCad sheets: Cell+BMS, Buck, Connector/NTC. Assign refs early.

3.1.11. Future KiCad Artifacts

- `kicad/pb3-replacement.kicad_pro` — full project.
- `mechanical/pb3-shell.step` — shell model for fit checks.
- `library/connector_aiwa_pb3.kicad_mod` — blade footprint.

3.1.12. Validation & Test Plan

Functional * Charge through USB-C. Log voltage and current curves. * Run a hungry deck like HS-PX505. Verify buck holds 2.0V ±0.1V during FF/REW. * Measure output with multimeter: target 2.0V (orig PB-3) or 2.4V (clone default).

Safety * Bake at 40 °C ambient. Keep silicon <85 °C. * Trip short-circuit protection under 20 ms.

Documentation * Export schematic PDFs and BOM straight from KiCad. * Keep PlantUML diagrams in `images/` so architecture matches the board.

3.1.13. Open Questions

- Do any decks sniff a third pin? Need proof before adding logic.
- Does the host trickle charge while on AC? Impacts design.
- Did Aiwa rely on SLA resistance rise for end-of-life? Might need a small gauge IC to mimic it.

3.1.14. Next Immediate Steps

1. Source a PB-3 donor for teardown.
2. Measure connector plating and friction.
3. Spin initial KiCad sheets with placeholders.
4. Prototype the buck on eval hardware and log noise against the audio path.

3.2. Measuring the PB-3 and context

This directory holds every measurement photo, caliper log, and derived dimension used to build the PB-3 models.

3.2.1. Measurement Photos

3.2.1.1. Original PB-3 SLA pack

3.2.1.1.1. Width [67.73, 67.75]

Figure 2. Width calipered underside (67.73mm)

Figure 3. Width calipered top (67.75mm)

3.2.1.1.2. Height [8.19, 8.12]

Figure 4. Height calipered side (8.19mm)

Figure 5. Height calipered side (8.12mm)

3.2.1.1.3. Depth [17.65, 17.55]

Figure 6. Depth calipered top (17.65mm)

Figure 7. Depth calipered top (17.55mm)

3.2.1.2. Battery holder

Figure 8. battery holder Depth

Figure 9. battery holder Width

Figure 10. battery holder Height

Figure 11. battery holder inside view

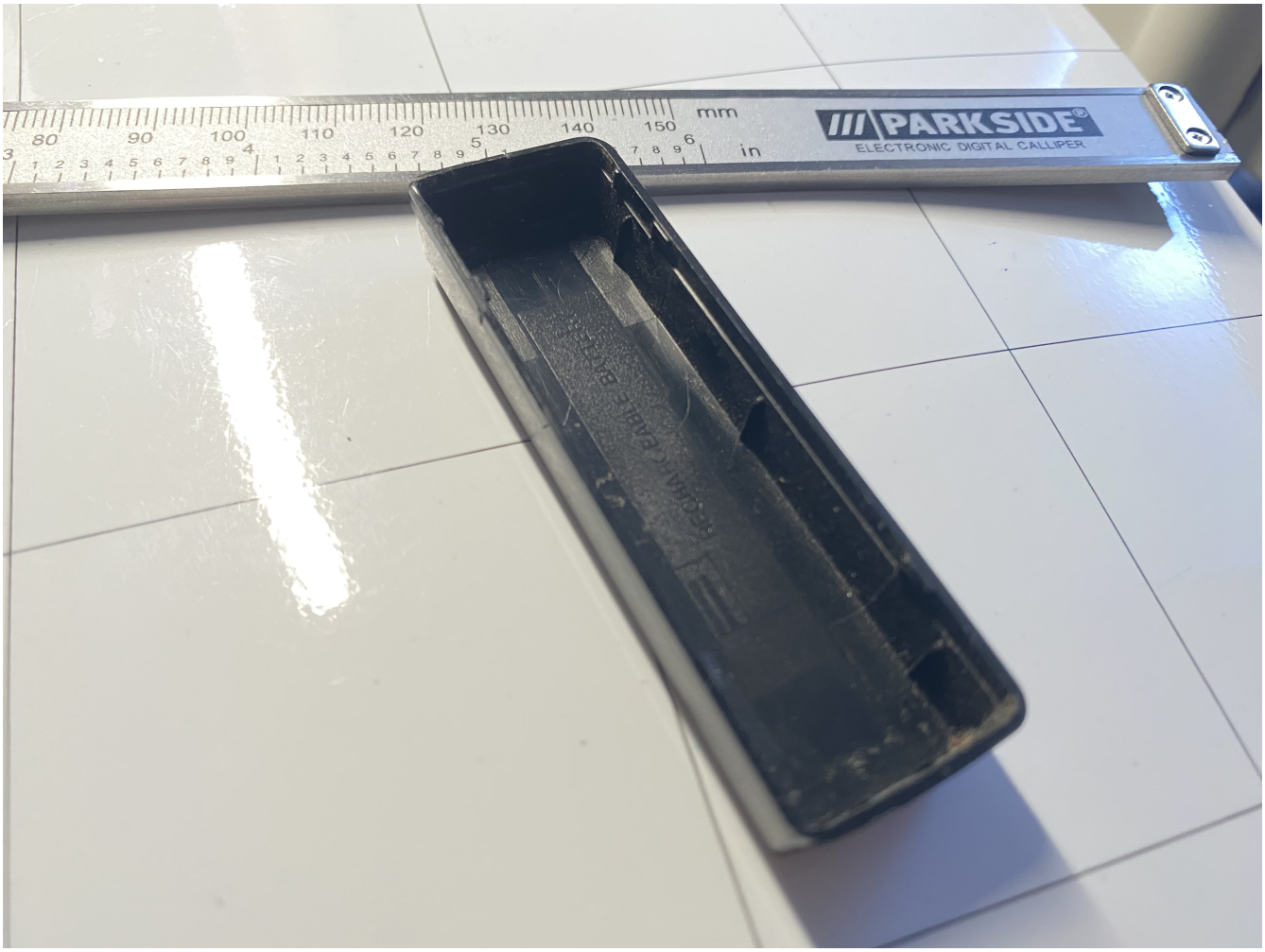

Figure 12. battery holder inside view

Figure 13. battery holder inside view

Figure 14. battery holder inside view

4. Architecture Decisions

4.1. Architecture Decision Records

4.1.1. Philosophy

Hemingway

Short sentences. No fluff. Say it once.

DRY (Don’t Repeat Yourself)

Write once. Reference everywhere. No copy-paste.

KISS (Keep It Simple, Stupid)

Simple beats clever. If you need comments, simplify.

NO MAGIC

Explicit over implicit. No hidden behavior. No surprises.

4.1.2. What

ADRs document **why** we chose this over that.

Each record:

- States the decision
- Lists alternatives considered
- Explains consequences

4.1.3. Why

Memory fades. Context dies. ADRs preserve intent.

When someone asks "why this way?", point to the ADR.

4.1.4. Format

ADR template

```
= ADR-NNN: Title

Date: YYYY-MM-DD
Status: Accepted | Deprecated | Superseded

== Context
Problem statement. One paragraph.

== Decision
What we chose. One sentence.

== Alternatives
- Option A: why rejected
- Option B: why rejected

== Implications
+ Good consequence
+ Good consequence
- Bad consequence
- Mitigation for bad consequence
```

4.1.5. Index

4.1.5.1. ADR-001: Commit Generated Artifacts Next to Source

Date: 2025-12-12
Status: Accepted

4.1.5.1.1. Context

This repo is a reverse engineering workbook. PlantUML diagrams, OpenSCAD models, KiCad schematics are both source **and** documentation. End users clone the repo to read design history, not to rebuild artifacts.

Generated outputs (PNG, STL, PDF) answer: "What does this look like?" Keeping them divorced from source breaks context. Viewing design requires tools. History loses meaning without rendered outputs.

4.1.5.1.2. Decision

Commit generated artifacts next to their source files.

4.1.5.1.3. Alternatives

Separate `docs/generated/` directory

+ Clean source tree + CI manages all outputs

- Divorces outputs from source context
- Requires CI to view design
- Clone without CI = incomplete artifact
- AsciiDoc references break (`image::openscad/foo.png` fails if PNG lives in `docs/`)

Ignore all generated files

+ Smallest repo + No merge conflicts on binaries

- Useless offline
- GitHub PR can't show visual diffs
- Design evolution invisible in history

Submodule for outputs

+ Separates binary history

- Overcomplicated for solo project
- Two clones to see one design

4.1.5.1.4. Implications

Positive:

+ `openscad/pb3-pack.png` lives next to `openscad/pb3-pack.scad` + AsciiDoc `image::` paths match file tree + Offline clone shows design without rebuilding + GitHub renders PNGs inline in PRs + History shows design evolution (old PNG = old intent) + No CI dependency for documentation

Negative:

- Repo size grows with each design iteration
- Merge conflicts possible on binary files (rare for solo)

- CI must commit and push outputs

Mitigations:

- Keep STL/PNG <5MB each (current sizes: <500KB)
- Use `git lfs` if files exceed 10MB
- CI commits with `[skip ci]` to avoid recursion
- For huge experimental outputs, use scratch `build/` dirs (gitignored)

4.1.5.1.5. Files Affected

Source directories commit their outputs:

- `openscad/*.{stl,png}` — rendered models
- `images/*.{png,svg}` — diagram exports
- `kicad/*.{pdf,svg}` — schematic/board renders

CI workflow updated to:

1. Generate artifacts into source directories
2. Commit outputs back to repo
3. Push to main branch

Scratch builds (local experiments) use `*/build/` and remain gitignored.

4.1.5.2. ADR-002: Use Text-Based CAD and Documentation Formats

Date: 2025-12-12
Status: Accepted

4.1.5.2.1. Context

Reverse engineering requires clear, searchable, versionable documentation. Design intent must survive iteration. Ambiguity kills projects. Binary formats (PNG, SVG, proprietary CAD) hide structure, resist search, and depend on artistic interpretation.

Need tools that:

- Version cleanly in git
- Search with grep
- Feed to LLMs without image parsing
- Embed measurements explicitly
- Avoid subjective interpretation

4.1.5.2.2. Decision

Use text-based formats for all CAD and documentation:

- **AsciiDoc** for all docs
- **PlantUML** for block diagrams
- **OpenSCAD** for mechanical models
- **KiCad** for PCB (S-expression text format)

4.1.5.2.3. Alternatives

Binary formats (Visio, Fusion 360, PNG diagrams)

+ Industry standard tools + Rich GUI workflows

- Not searchable (grep useless)
- Binary diffs meaningless
- LLM can't parse structure
- Locked to specific tools
- Measurements hidden in pixels
- Interpretation varies by viewer

Markdown + hand-drawn SVG

+ Simple, portable

- SVG is XML (verbose, hard to write)
- Hand-drawn = ambiguous dimensions
- No parametric models
- Design intent lost in pixels

Wiki or Google Docs

+ Collaborative + Rich media

- Not in git (history scattered)
- Export fragile
- Offline broken
- Search locked to platform

4.1.5.2.4. Implications

Positive:

+ `grep "68.*17.8.*8" openscad/` finds all references to pack dimensions + `git diff` shows exact measurement changes: `outer_len = 67` → `outer_len = 68` + LLM can ingest `.puml` and suggest corrections: "2.4V output conflicts with 2.0V spec" + PlantUML connections are explicit: `CELL -> BUCK : 3.0-4.2V` (no guessing from arrows) + OpenSCAD embeds tolerances: `wiggle=.01` (visible in source, not CAD property panel) + KiCad `.kicad_sch` is S-expression: searchable for part numbers, net names + Docs live next to code: `openscad/pb3-pack.scad` + README.asciidoc in same commit + Dead docs impossible: if code changes, doc changes (same file or adjacent) + No priest needed: `outer_len = 68` means 68mm, not "roughly this wide"

Negative:

- Steeper learning curve (PlantUML syntax, OpenSCAD code)
- No WYSIWYG (must render to see)
- Collaboration requires text-editing skill
- Binary export still needed for viewing (PNG, STL)

Mitigations:

- CI auto-generates viewable outputs (PNG, STL, PDF) from text sources
- Commit both source and renders (ADR-001)
- Include examples in repo (`images/pb3-architecture.plantuml` as template)
- LLMs can write PlantUML/OpenSCAD from specs (reverse engineering ally)

4.1.5.2.5. Examples

Searchable dimensions:

```
outer_len = 68;    // grep finds this
outer_width = 17.8;
```

Explicit connections:

```
CELL --> BUCK : 3.0-4.2V  // no guessing voltage
BUCK --> OUT : 2.0V       // explicit output
```

Versionable structure:

```
- BUCK --> OUT : 2.4V
+ BUCK --> OUT : 2.0V  // git diff shows intent change
```

LLM-friendly input: "Here's the PlantUML. Does the 2.0V output match the schematic's buck feedback resistor?"

4.1.5.2.6. Tools Used

- AsciiDoc: prose, specs, workbook
- PlantUML: block diagrams, architecture
- OpenSCAD: mechanical models, fit tests
- KiCad: schematic + PCB (text-based S-expression format)

All text. All searchable. All versionable. All LLM-ingestible.

4.1.5.3. ADR-003: Use GitHub Actions for All Build Automation

Date: 2025-12-12
Status: Superseded by ADR-006 (single job pipeline)

4.1.5.3.1. Context

Generated artifacts (PNG, STL, PDF) must stay synchronized with source files. Manual builds fail. Developers forget to regenerate. Wrong versions ship. "It works on my machine" breaks collaboration.

Need automation that:

- Runs on every commit
- Links builds to exact source version
- Prevents stale outputs
- Documents build process explicitly
- Survives author memory loss

4.1.5.3.2. Decision

Use GitHub Actions for all CAD rendering and artifact generation.

No local build scripts as primary workflow.

4.1.5.3.3. Alternatives

Local build scripts only

+ Fast iteration + No network dependency

- Forget to run = stale outputs committed
- "Works on my machine" inconsistencies
- Process lives in author’s head
- New contributor: "How do I build?"
- No proof outputs match sources

Manual screenshot/export workflow

+ Simple tools + No scripting

- Wrong version screenshots plague repos
- No traceability (which source made this PNG?)
- Forgot to save = weeks hunting mismatch
- Undocumented steps = tribal knowledge

Pre-commit hooks

+ Automatic on commit

- Slow local commits (blocks developer)
- Hard to debug when broken
- Environmental differences break hooks
- Still local = "works on my machine"

Makefile + CI

+ Standard Unix pattern + Local + remote builds

- Two build paths = drift risk
- Makefile syntax arcane
- Windows developers struggle
- Process still partly hidden

4.1.5.3.4. Implications

Positive:

+ **Sync guaranteed**: outputs regenerated every push, impossible to commit stale PNG + **Traceable**: Actions log links to exact commit that built artifact + **Documented workflow**: `.github/workflows/cad-build.yml` IS the build documentation + **Preview while editing**: PlantUML/OpenSCAD extensions show live preview in VS Code, no surprise when CI builds + **Repeatable**: same Docker images, same results, always + **Inspectable**: workflow file shows thought process: "first OpenSCAD, then PlantUML, then publish" + **No magic**: workflow is code, not author's mental model requiring MRI to extract + **Works-on-merge**: GitHub Actions runs on forks, contributors see build before PR + **Blame trail**: git log shows when output changed and why (CI commit message references source change)

Negative:

- Network required to see final outputs (mitigated: local preview extensions)
- Slower feedback than local build (mitigated: preview extensions, small changes iterate locally)
- CI quota consumption (mitigated: build only on source file changes)
- Initial setup cost (one-time, then keeps working)

Mitigations:

- VS Code extensions preview PlantUML/OpenSCAD during editing (instant feedback)
- Actions trigger only on paths: `openscad/, images/`, `kicad/**` (skip unrelated commits)
- Local Docker containers available for manual iteration (same images CI uses)
- Workflow continues on job failure: `continue-on-error: true` (partial build better than none)

4.1.5.3.5. Why Not Magic

Actions EXPOSE process:

```
# This is the build documentation. No hidden steps.
- name: Render
  run: |
    docker run ... openscad -o /w/build/openscad/$s.stl /w/$f
```

Compare to undocumented mental model: - Author: "I open OpenSCAD, export STL, save PNG, commit both" - New contributor: "Which settings? What colorscheme? Why this size?" - Six months later: "How did I make this?"

Actions preserve intent. Author leaves. Process survives.

4.1.5.3.6. Workflow Structure

`cad-build.yml`:

1. **openscad job**: render STL + PNG for all `.scad` files
2. **plantuml job**: render SVG for all `.puml` files
3. **kicad job**: render PDF + SVG via KiBot
4. **publish job**: commit outputs back to repo (ADR-001)

Each job: - Runs in isolation - Uses pinned Docker image (versioned, reproducible) - Continues on error (partial build survives) - Logs exact commands (auditable)

4.1.5.3.7. Traceability Example

Commit: `fix: correct outer_len to 68mm`

Actions runs → generates new STL → commits:

```
ci: refresh outputs

Generated from:
- openscad/pb3-pack.scad (abc1234)
```

Git blame on `pb3-pack.stl` → points to CI commit → references source commit → full trace.

No "oops wrong image" mystery.

4.1.5.3.8. Living Documentation

Workflow file = build manual.

New contributor clones repo, reads `.github/workflows/cad-build.yml`, understands: - Which Docker images - Which commands - Which outputs - Which paths trigger

No MRI. No author interview. Just code.

4.1.5.3.9. Tools vs Process

Local preview: VS Code extensions (instant feedback while editing)

Canonical build: GitHub Actions (guarantees sync, documents process)

Both. Not either/or.

4.1.5.4. ADR-004: File Extension Conventions for Main vs Include Files

Date: 2025-12-12
Status: Accepted

4.1.5.4.1. Context

Repos accumulate many AsciiDoc and PlantUML files. Finding the main entry point becomes hard. Need convention to distinguish "main documentation" from "included fragments" and "build sources" from "style imports".

Without convention: - Which `.adoc` is the root doc? - Which `.puml` gets rendered? - Grep returns too many false positives

4.1.5.4.2. Decision

Use extension suffixes to signal file role:

AsciiDoc: - `.asciidoc` = main documentation (entry points) - `.adoc` = include fragments (not standalone) - `includes/*.adoc` = shared fragments (glossary, copyright, formatting macros)

PlantUML: - `.plantuml` = diagrams to render - `.puml` = style/config imports (not rendered)

OpenSCAD: - `.scad` = models to render - `include-*.scad` = libraries (modules/functions only, skip rendering)

Build outputs: - Always adjacent to source (ADR-001) - `docs/` = rendered AsciiDoc HTML only - `.png`, `.stl`, `.svg` live next to `.scad`/`.plantuml`

4.1.5.4.3. Alternatives

All use same extension

+ Simpler

- No way to find main docs fast
- Glob `*.adoc` catches 47 files
- New contributor: "Which one do I read?"

Separate directories

+ Clear grouping: `main/`, `includes/`

- Breaks locality (main doc far from includes)
- Path references fragile

Metadata in filename

+ Example: `README.main.adoc`

- Ugly
- Sorting broken

4.1.5.4.4. Implications

Positive:

+ `find . -name ".asciidoc"` → main docs only + `find . -name ".plantuml"` → diagrams to render + CI glob: `images/*.plantuml` (ignores `common.puml` style) + New contributor sees `README.asciidoc` and knows: "Start here" + Include detection: `grep "^include::" *.adoc` finds fragments + Clear intent: `.asciidoc` = public, `.adoc` = internal

Negative:

- Convention not obvious without docs
- Mix of extensions in directory listings

Mitigations:

- Document in ADR (this file)
- Enforce in CI: fail if `.asciidoc` includes another `.asciidoc`
- README mentions convention

4.1.5.4.5. Examples

```
README.asciidoc                      ← main doc (read this first)
pb3-notes.asciidoc                   ← main workbook
adr/README.adoc                      ← include fragment (embedded in parent)
adr/adr-001-foo.adoc                 ← include fragment
includes/include-glossary.adoc       ← shared fragment (glossary)
includes/include-copyright.adoc      ← shared fragment (license boilerplate)

images/pb3-architecture.plantuml  ← render this
images/common.puml                ← style import (not rendered)

openscad/pb3-pack.scad            ← render this
openscad/include-utils.scad       ← library (skip rendering)
openscad/pb3-pack.png             ← output (adjacent)
openscad/pb3-pack.stl             ← output (adjacent)
```

4.1.5.4.6. Directory Structure

```
docs/                   ← HTML output from asciidoc rendering only
includes/               ← Shared AsciiDoc fragments (glossary, copyright, macros)
openscad/*.{png,stl}    ← CAD outputs (adjacent to .scad)
```

```
images/*.{png,svg}       ← diagram outputs (adjacent to .plantuml)
kicad/*.{pdf,svg}        ← schematic/PCB outputs (adjacent to .kicad_*)
```

No generated artifacts in `docs/generated/`. That pattern is deprecated (ADR-001).

4.1.5.4.7. CI Impact

Workflows use extension filters:

```
for f in images/*.plantuml; do  # not *.puml
  plantuml "$f"
done

for f in openscad/*.scad; do
  [[ "$(basename "$f")" =~ ^include- ]] && continue  # skip libraries
  openscad -o "${f%.scad}.stl" "$f"
  openscad -o "${f%.scad}.png" "$f"
done
```

AsciiDoc rendering looks for `*.asciidoc` only.

4.1.5.5. ADR-005: Conditional imagesdir Management for Book Assembly

Date: 2025-12-12
Status: Accepted

4.1.5.5.1. Context

README.asciidoc assembles multiple `.adoc` fragments that contain images. Each fragment must work both:

1. **Standalone**: rendered directly (GitHub, local preview)
2. **Included**: assembled into main book

Problem: image paths break when included.

Example: - `images/README.adoc` contains `image::diagram.png[]` - Standalone: works (PNG is local) - Included in root README: fails (looks for `diagram.png` in root, not `images/`)

Need: conditional `:imagesdir:` that adapts to context without breaking either mode.

4.1.5.5.2. Decision

Use `:flag-book:` to conditionally manage `:imagesdir:`.

Main document (README.asciidoc):

Set book mode flag

```
//:flag-book: true
```

Each included fragment:

Save, override, restore imagesdir

```
//ifdef::flag-book[]
//:imagesdir-save: {imagesdir}
//:imagesdir: <subdirectory>
//endif::[]
//
//... content with images ...
//
//ifdef::flag-book[]
//:imagesdir: {imagesdir-save}
//endif::[]
```

4.1.5.5.3. Alternatives

Absolute paths in image macros

+ Works both standalone and included

- Fragile (breaks on repo rename)
- Verbose: `image::images/diagram.png[]` everywhere
- Not idiomatic AsciiDoc

Duplicate files

+ Clean separation

- Maintenance nightmare
- Violates DRY
- Drift guaranteed

Separate standalone vs include versions

+ Each optimized for context

- Double maintenance
- Violates single source of truth

Always use subdirectory paths

+ Simple

- Breaks standalone rendering
- GitHub can't preview

4.1.5.5.4. Implications

Positive:

+ Single source: one `.adoc` file, two render contexts + Standalone works: `images/README.adoc` previews correctly on GitHub + Book works: `README.asciidoc` assembles all fragments with correct image paths + No path duplication: `image::foo.png[]` not `image::images/foo.png[]` + Graceful fallback: if `:flag-book:` not set, standalone mode assumed + Restores state: `:imagesdir-save:` preserves any parent setting

Negative:

- Boilerplate: every fragment needs ifdef blocks
- Non-obvious: requires ADR to explain
- Fragile: forget closing block = broken subsequent includes

Mitigations:

- Document pattern in ADR (this file)
- CI-generated READMEs include pattern automatically
- Template in `includes/include-toc-standalone.adoc` for manual fragments
- Lint rule: fail if `:imagesdir:` changed without restore

4.1.5.5.5. Pattern

Step 1: Main doc sets flag

Main document

```
//= Main Document
//:flag-book: true
//
//include::subdoc/README.adoc[]
```

Step 2: Subdoc saves and overrides

Subdocument with conditional path

```
//= Subdoc
//ifdef::flag-book[]
//:imagesdir-save: {imagesdir}
//:imagesdir: subdoc
//endif::[]
//
//image::photo.png[]  // standalone: subdoc/photo.png, book: subdoc/photo.png
```

Step 3: Subdoc restores at end

Restore parent context

```
//... last image ...
//
//ifdef::flag-book[]
//:imagesdir: {imagesdir-save}
//endif::[]
```

4.1.5.5.6. Files Using Pattern

Manual: - `measurements/README.adoc`

CI-generated: - `images/README.adoc` - `openscad/README.adoc`

Template: - `includes/include-toc-standalone.adoc` (provides ifdef block)

4.1.5.5.7. Why Save/Restore

Simple reset (`:imagesdir:`) loses parent context:

Wrong: loses parent setting

```
//:imagesdir: assets         // parent sets this
//
//include::sub/doc.adoc[]    // sub sets :imagesdir: sub, then resets to empty
//                           // WRONG: parent's "assets" is lost
```

Save/restore preserves nesting:

Correct: preserves parent setting

```
//:imagesdir: assets
//
//include::sub/doc.adoc[]    // saves "assets", sets "sub", restores "assets"
//                           // CORRECT: parent context preserved
```

Enables arbitrary nesting depth without coordination.

5. Credits

- Stereo2Go forum — original research
- OpenSCAD: ghcr.io/donnels/openscad:main
- PlantUML: ghcr.io/donnels/plantuml:main
- KiBot: INTI-CMNB/KiBot@v2_dk8

[1] In-house photo set `measurements/IMG_9373.jpeg`–`IMG_9385.jpeg`.

[2] Rossoe et al., "Rechargeable battery replacement for AIWA PB-3 / PB-4," Stereo2Go forums, Sep 2023–Apr 2025. Offline copy: `research/Rechargeable battery replacement for AIWA PB-3 _ PB-4 _ Stereo2Go forums.html`. Original URL: https://stereo2go.com/

forums/threads/rechargeable-battery-replacement-for-aiwa-pb-3-pb-4.9025/