

D-O AIO32 v2

ESP32-S3-based AIO Controller

User Handbook

Complete user handbook for the D-O AIO32 v2 controller board. Covers firmware upload, hardware overview, IMU setup, RC (iBus + SBUS), button interface, serial CLI, balance PID tuning, sound, display, battery monitoring, troubleshooting, and safety. Target audience: makers building their own D-O.

Version 2.1.1 — 2026-04-23

Disclaimer and Liability Notice

D-O AIO32 v2 is a do-it-yourself project. Building, wiring, configuring and operating the system involves work that, if done incorrectly, can cause damage to property or persons.

By using this documentation, hardware, and software you acknowledge:

- You are solely responsible for all work on your system.
- You have the required basic knowledge or seek qualified help.
- You verify voltage, polarity, fuses, connectors before every power-on.
- You only use components matching the application.
- You comply with local regulations.

The author and all contributors accept no liability for damage caused by incorrect wiring, wrong components, improper operation, or insufficient safety measures.

The documentation and software are provided **as is** without warranty of any kind. If in doubt, seek qualified help before working on the system.

Table of Contents

| | |
|---|-----------|
| Disclaimer and Liability Notice | 2 |
| Table of Contents | 3 |
| Introduction — What is D-O AIO32 v2? | 6 |
| Key Features | 6 |
| Relation to the Mega AIO line | 6 |
| Repository and support | 7 |
| Quick Start | 8 |
| Hardware Overview | 10 |
| Key components | 10 |
| Pin map | 11 |
| Servo labels on the board | 11 |
| Uploading the Firmware | 13 |
| Board settings | 13 |
| Required libraries | 13 |
| Upload procedure | 14 |
| IMU Setup and Calibration | 15 |
| IMU modes | 15 |
| Calibration procedure | 15 |
| Madgwick beta | 15 |
| RC Receiver Setup (iBus + SBUS) | 17 |
| Channel map | 17 |
| iBus setup (default) | 17 |
| SBUS setup (no inverter) | 17 |
| Button Interface | 19 |
| Button labels and functions | 19 |
| Debug button events | 19 |
| Serial CLI Reference | 20 |
| Top-level commands | 20 |

| | |
|--|-----------|
| pid — PID tuning | 21 |
| imu — IMU, calibration, Madgwick | 21 |
| drive — driving dynamics | 22 |
| rc — protocol + channels | 22 |
| sound — DFPlayer | 23 |
| display — TFT | 23 |
| balance — state control | 23 |
| feature — runtime flags | 24 |
| test — manual hardware test | 24 |
| debug — dumps + live streams | 25 |
| Balance and PID Tuning | 26 |
| Recommended starting values | 26 |
| Tuning procedure | 26 |
| Sound (DFPlayer Mini + SD) | 27 |
| Track map (identical to Mega line) | 27 |
| SD card format + filenames | 27 |
| Speaker wiring | 28 |
| Volume and mute | 28 |
| Display and Status LED | 29 |
| Display modes | 29 |
| Always-on vs auto-sleep | 29 |
| NeoPixel state patterns | 29 |
| Battery Monitoring | 30 |
| Configuration | 30 |
| Calibration | 30 |
| Troubleshooting | 32 |
| Safety | 34 |

Part I

Getting Started

Introduction — What is D-O AIO32 v2?

The D-O AIO32 v2 is an ESP32-S3-based all-in-one controller for the self-balancing D-O droid from Star Wars: The Rise of Skywalker. It is the younger sibling of the Arduino-Mega-based "Control & Power Board" and targets the same droid, same transmitter setup, same sound files — but uses a modern microcontroller, dual IMUs with Madgwick quaternion fusion, a native colour display, button-based configuration and a full interactive CLI. Both product lines are maintained in parallel.

Key Features

- **ESP32-S3 all-in-one:** TENSTAR TS-ESP32-S3 module with native USB-C and 4 MB flash.
- **Dual-IMU fusion:** QMI8658C (on the TENSTAR module) plus LSM6DS3 (on the AIO32 PCB) combined by Madgwick 6-DoF AHRS with runtime-tuneable beta.
- **Dual RC protocols:** FlySky iBus (default) and Futaba/FrSky SBUS, both runtime-switchable via CLI — SBUS works without an external inverter thanks to ESP32 UART hardware invert.
- **Four servos** driven via native LEDC PWM (50 Hz, 14-bit) — crisp idle animations on mainbar + 3 head axes.
- **DFPlayer Mini** sound (Makuna library, non-blocking) with mood triggers, idle sounds, tilt warnings.
- **1.14" ST7789 colour TFT** with adaptive portrait/landscape layout, live pitch / battery / PID / mode display.
- **NeoPixel state indicator** with change-detected colour patterns for INIT / CALIBRATING / READY / RUNNING / ERROR / E-STOP.
- **6 labelled buttons (B1-B6)** for Start/Stop, Calibrate, Display, Debug/Select, Mode, Back/Emergency — all common actions without CLI.
- **Full serial CLI** (~80 commands): PID tuning, adaptive bands, IMU modes, RC protocol, drive dynamics, streaming debug dumps.
- **Adaptive 3-band PID** with linear blending between slow/medium/fast gains.
- **Arcade / Tank mixing modes** with deadband, expo, motor-ramping and dynamic lean angle.
- **Auto-baseline calibration** (B1-long): 2-second pitch average becomes the new target angle, persisted in NVS.
- **NVS persistence** for every tuneable parameter, including RC protocol, IMU mount, Madgwick beta, target angle.

Relation to the Mega AIO line

The AIO32 is **not a replacement** for the Mega AIO board — both are maintained as active product lines. The Mega line is simpler, proven, and has an extensive handbook of its own. The AIO32 targets users who want newer silicon, a display, colour LED feedback, button-based configuration and the cleaner architecture that comes with 240 MHz and 2 MB PSRAM. Both use the same iBus/SBUS channel map and the same sound-file layout on SD.

Repository and support

Source code, hardware KiCad files, handbook PDFs and issue tracker live in the shared PrintedDroid repository on GitHub — same repo as the Mega AIO line, different sub-folders:

- <https://github.com/PrintedDroid/D-O-Printed-Droid>

Commercial support, pre-built boards and droid kits: <https://www.printed-droid.com/>

Quick Start

From a wired-up AIO32 to rolling in about 30 minutes, assuming hardware is assembled. If something breaks, jump to the Troubleshooting chapter.

Wheels off the ground for first power-on. Balance PID can swing the droid violently during calibration.

1. Install Arduino IDE + ESP32 core

Arduino IDE 2.x. File → Preferences → additional boards URL: https://raw.githubusercontent.com/espressif/arduino-esp32/gh-pages/package_esp32_index.json. Boards Manager: install **esp32 by Espressif** version 3.3.7.

2. Install libraries

Library Manager: **Adafruit ST7789**, **Adafruit NeoPixel**, **DFMiniMp3 by Makuna**. (Madgwick is an own implementation in the sketch, no library needed.)

3. Select board

Tools → Board → **Adafruit Feather ESP32-S3 TFT** (the AIO32 uses the TENSTAR module with compatible pinout). Tools → USB CDC On Boot: **Enabled**, PSRAM: **Disabled**, Flash Size: **4MB**, Partition Scheme: **Default 4MB with spiffs**.

4. Open sketch

Open `D-O_AIO32_v2.1/D-O_AIO32_v2.1.ino`.

5. Upload

Hold BOOT on the AIO32, press RESET once, release BOOT after a second. Click Upload. After upload press RESET to run the sketch.

6. Open Serial Monitor at 115200 baud

You should see `=== D-O AIO32 v2.1.x ===` followed by init diagnostics. Type `help` for a CLI overview.

7. Calibrate IMUs

Place D-O upright and perfectly still on a level surface. Short-press **B2 (Calibrate)**, or type `imu cal`. Hold still for 5 seconds. Offsets are stored in NVS.

8. Baseline the target angle

After IMU calibration, long-press **B1 (Start/Stop)** (or send `balance baseline`). 2-second pitch average becomes the new target angle. Persists in NVS.

9. Check RC

Type `debug stream rc`. Move sticks — channel values must move smoothly 1000..2000. Press any key to stop the stream.

10. First balance attempt

Wheels on ground, step back. Short-press **B1 (Start/Stop)** to start balance (or `balance start`). If the droid tips on power-on, adjust target angle: `pid target -0.5` or similar, then re-baseline.

Tip: every single tuneable value in the sketch is reachable from the CLI, saved to NVS, and restored across reboots. No need to reflash for normal tuning.

Hardware Overview

The AIO32 v2 board is a single PCB carrying the TENSTAR ESP32-S3 module, a **Cytron MDD10A** dual-channel motor driver (10 A per channel — same as the Mega AIO line), a secondary IMU, a DFPlayer socket, four servo headers, a 6-button ADC ladder and a battery voltage divider. Same droid mechanics, same sound library, same transmitter map as the Mega line — only the controller brain is different.

Key components

| Component | Role | Notes |
|-----------------------|--------------------|--|
| TENSTAR TS-ESP32-S3 | Main controller | 240 MHz, 4 MB flash, native USB-C, ST7789 TFT 1.14" 135x240 integrated, QMI8658C IMU onboard |
| LSM6DS3TR-C | Secondary IMU | I2C 0x6A (SA0 pulled low via solder jumper on the module) |
| Cytron MDD10A | Motor driver | Dual-channel, 10 A continuous per channel. Same driver as the Mega line. |
| DFPlayer Mini | Sound | UART via Makuna DFMiniMp3 library, plays /mp3/NNNN.mp3 from SD |
| NeoPixel (on TENSTAR) | Status LED | GPIO 33, WS2812, 1 LED |
| 6 buttons (B1-B6) | Config + safety | Labelled B1..B6 on the PCB, each with short-press and long-press events |
| Battery divider | Pack voltage sense | On GPIO 16 (ADC2). Factor ~5.74 matches 47k/10k divider |

Pin map

| GPIO | Function | Notes |
|------|--|---|
| 2 | iBus / SBUS RC input | Serial1-RX. UART invert configured at runtime when SBUS is selected |
| 13 | Motor 1 DIR (Cytron MDD10A ch.1 direction) | |
| 11 | Motor 1 PWM (Cytron MDD10A ch.1 speed) | LEDC |
| 10 | Motor 2 DIR (Cytron MDD10A ch.2 direction) | |
| 9 | Motor 2 PWM (Cytron MDD10A ch.2 speed) | LEDC |
| 6 | Servo Mainbar — board label "Servo 3" (CH3) | LEDC 50 Hz 14-bit |
| 15 | Servo Head Pitch — board label "Servo 4" (CH4) | LEDC 50 Hz 14-bit |
| 14 | Servo Head Yaw — board label "Servo 5" (CH5) | LEDC 50 Hz 14-bit |
| 8 | Servo Head Roll — board label "Servo 6" (CH6) | LEDC 50 Hz 14-bit |
| 42 | I2C SDA | Both IMUs + optional BMP280 |
| 41 | I2C SCL | |
| 33 | NeoPixel status LED | WS2812 onboard |
| 5 | Buttons B1-B3 | |
| 12 | Buttons B4-B6 | Blocked when WiFi is active (ADC2 pin) |
| 16 | Battery voltage sense | ADC2-CH5 — also blocked during WiFi. Divider 47k/10k |
| 18 | DFPlayer TX (to module RX via 1 kΩ resistor) | UART2-TX |
| 17 | DFPlayer RX (from module TX) | UART2-RX |

ADC2 caveat: GPIO 12 (button ladder 2) and GPIO 16 (battery sense) are ADC2 pins. When WiFi is enabled by a future firmware, these pins become unavailable for analog reads. The current firmware does not use WiFi, so the conflict is theoretical. Plan a pin migration to ADC1 before enabling OTA.

Servo labels on the board

The four servo headers on the AIO32 are silk-screened "**Servo 3**" through "**Servo 6**". These labels map 1:1 to the FlySky iBus / SBUS channel numbers — the mainbar servo plugs into the header labelled "Servo 3" and receives RC channel 3, head pitch plugs into "Servo 4" = CH4, and so on. There is no "Servo 1" or "Servo 2" on the board because channels 1 and 2 are reserved for the two drive motors.

| Board label / Board-Label | Function / Funktion | FlySky / iBus CH |
|---------------------------|-------------------------|------------------|
| Servo 3 | Mainbar | CH3 |
| Servo 4 | Head Pitch / Kopf Pitch | CH4 |

| Board label / Board-Label | Function / Funktion | FlySky / iBus CH |
|---------------------------|-----------------------|------------------|
| Servo 5 | Head Yaw / Kopf Yaw | CH5 |
| Servo 6 | Head Roll / Kopf Roll | CH6 |

Uploading the Firmware

The AIO32 firmware lives in `D-O_AIO32_v2.1/D-O_AIO32_v2.1.ino`. It is a multi-file Arduino sketch; all `.h` and `.cpp` files in that folder belong to the same compilation unit. Compile target: Arduino IDE 2.x with the Espressif ESP32 core.

Board settings

| Setting / Einstellung | Value / Wert |
|-----------------------|--|
| Board | Adafruit Feather ESP32-S3 TFT |
| USB CDC On Boot | Enabled |
| CPU Frequency | 240 MHz |
| Flash Mode | QIO 80 MHz |
| Flash Size | 4 MB |
| Partition Scheme | Default 4MB with spiffs (1.2MB APP / 1.5MB SPIFFS) |
| PSRAM | Disabled |
| Upload Mode | UART0 / Hardware CDC |
| Upload Speed | 921600 (lower to 460800 if uploads fail) |
| Core Version | 3.3.7 (verified working; later 3.x versions should work too) |

Required libraries

| Library | Author | Why |
|--------------------|--------------------------|---|
| Adafruit GFX | Adafruit | ST7789 display primitives |
| Adafruit ST7789 | Adafruit | 1.14" colour TFT driver |
| Adafruit NeoPixel | Adafruit | WS2812 status LED |
| DFMiniMp3 | Makuna | DFPlayer Mini (non-blocking, supports playMp3FolderTrack) |
| SensorLib QMI8658C | Xinyuan LilyGO / TENSTAR | QMI IMU low-level driver |
| -- Madgwick -- | -- included -- | Own implementation in madgwick_ahrs.h/.cpp, no external dep |
| -- iBus/SBUS -- | -- included -- | Inline parsers in rc_receiver.cpp, no external dep |
| -- LSM6DS3 -- | -- included -- | Direct I2C register access in imu_handler.cpp |

Upload procedure

- Connect AIO32 via USB-C.
- Select the correct COM port (device shows up as "USB JTAG/serial debug unit" or "USB Serial").
- If the board does not enter bootloader automatically: hold **BOOT**, press **RESET**, release BOOT after a second.
- Click Upload. First upload takes ~90 seconds (esptool flashes bootloader + partition table + app).
- When upload completes, press **RESET** once to run the new firmware.
- Open Serial Monitor at **115200 baud**. Early boot log shows I2C scan, IMU init, NVS load, RC protocol, TFT init.

IMU Setup and Calibration

The AIO32 has two IMUs: the **QMI8658C** on the TENSTAR module (I2C 0x6B) and the **LSM6DS3TR-C** on the AIO32 PCB (I2C 0x6A, SA0 jumpered to GND). Both stream accel and gyro into a **Madgwick 6-DoF AHRS** that produces a quaternion, from which pitch/roll/yaw are extracted. You choose at boot which IMU is live, or let the firmware average both.

IMU modes

| Mode | Behaviour |
|--------------------|--|
| QMI_ONLY | Only the QMI8658C is read. Useful if the LSM is not populated or malfunctioning. |
| LSM_ONLY | Only the LSM6DS3 is read. Useful if the QMI is unreliable on a specific TENSTAR batch. |
| BOTH_AVG (default) | Both IMUs are read each tick. Accel and gyro are averaged before feeding Madgwick. Best noise rejection. |

Switch at runtime: `imu mode qmi|lsm|both`. Verify with `imu status`.

Calibration procedure

- Place D-O on a level surface, perfectly upright, perfectly still.
- Send `imu cal` via CLI, or long-press B4 (depends on button binding).
- Stay still for 5 seconds. The firmware prints live count-down, then averages 500 samples per IMU.
- Gyro bias (3 axes per IMU) is subtracted from all future readings. Accel is not bias-corrected (gravity is the reference).
- After calibration, send `imu bias show` to see the computed offsets. They are automatically written to NVS.
- Calibration survives reboots. Re-calibrate only if you move the droid to a new location or change IMUs.

Madgwick beta

Madgwick beta is the gradient-descent gain that mixes accel into the gyro-integrated quaternion. Lower = smoother, more gyro trust, higher drift risk. Higher = snappier, more accel weight, noisier. D-O default is **0.1**. Typical useful range 0.05..0.20.

Runtime adjust: `imu beta 0.12` then `save`. The change takes effect on the next IMU tick (no reboot).

Part II

Core Features

RC Receiver Setup (iBus + SBUS)

The AIO32 supports both FlySky iBus (default) and Futaba/FrSky SBUS. The big advantage over the Mega line is that the ESP32 UART can invert its RX line in hardware, so **SBUS works without an external inverter**. Switch between protocols at runtime via CLI — the change is persisted in NVS and survives reboots.

Channel map

Channel mapping is identical to the Mega AIO handbook — same transmitter config works on both controller boards.

| CH | Function / Funktion | FlySky default |
|------|---|----------------------------------|
| CH1 | Drive 1 (Steering in Arcade, Motor 1 in Tank) | Right Stick L/R |
| CH2 | Drive 2 (Throttle in Arcade, Motor 2 in Tank) | Right Stick U/D |
| CH3 | Mainbar Servo | not assigned — map to VrA |
| CH4 | Head Servo 1 (Pitch) | Left Stick U/D |
| CH5 | Head Servo 2 (Yaw) | Left Stick L/R |
| CH6 | Head Servo 3 (Roll) | VrB dial |
| CH7 | Sound Mute (2-pos) | SwA |
| CH8 | Sound Mode / Trigger (2-pos) | SwB |
| CH9 | Sound Mood (3-pos, neg/mid/pos) | SwC |
| CH10 | Sound Squeak (2-pos) | free — map to any unused control |

CH3 gap: FlySky transmitters (FS-i6, i6X, i6S) do not assign CH3 by default. Map CH3 to VrA or a switch in "Functions → Aux. Channels" or the mainbar servo will not move.

iBus setup (default)

- Bind FlySky receiver to transmitter (press bind button on receiver while powering it, activate bind mode on transmitter).
- Wire receiver iBus output (single wire, usually labeled "iBus" or "SENS") to AIO32 **GPIO 2**.
- Also wire receiver 5V and GND.
- Boot AIO32. CLI `rc status` shows iBus as active protocol; `rc channels` shows live channel values.

SBUS setup (no inverter)

- Bind FrSky / Futaba receiver to its transmitter (binding procedure is transmitter-specific).
- Wire receiver SBUS output directly to AIO32 **GPIO 2**. **No inverter needed** — the ESP32 handles invert in firmware.

- Also wire receiver 5V and GND.
- Boot AIO32. Switch protocol via CLI: `rc protocol sbus`, then `save`.
- Verify: `rc status` shows "SBUS" as active; `rc channels` shows live values moving with sticks.
- Rollback: `rc protocol ibus` then `save`.

Tip: you can keep both receiver types in your build and switch by unplugging one / plugging the other and sending one CLI command. The droid behaves identically after the switch.

Button Interface

The AIO32 has six physical buttons labelled **B1 through B6**. Each button covers one area of operation — balancing, calibration, display, debug, modes, and safety. All buttons support short press and long press; the short press runs the primary function of the label, the long press runs a related secondary function where it makes sense. Every press is also echoed to the serial monitor as `[btn] BN short|long` so you can verify wiring without leaving the CLI.

Button labels and functions

| Button | Label | Short press | Long press |
|--------|------------------|---|---|
| B1 | Start/Stop | Toggle balance — start in READY, stop in RUNNING. From ERROR / EMERGENCY-STOP, clears the fault and does a soft-reboot. | Auto-baseline target angle — 2-second pitch average while perfectly upright becomes the new neutral. |
| B2 | Calibrate | Run IMU calibration (keep droid still for ~5 seconds). Gyro bias is re-learned. | Enter the PID tuning menu on the TFT (use B3 to navigate entries, B4 to adjust, B6 to exit). |
| B3 | Display | Cycle display mode: Telemetry → Diagnostics → Display-Off → back. In PID menu: navigate to next entry. | Toggle "always-on" backlight (display never auto-sleeps). |
| B4 | Debug Select | Toggle serial debug output on/off. In PID menu: adjust the selected value. | Show the system info page on the TFT. |
| B5 | Mode | Cycle drive mixing mode (Arcade ↔ Tank). | Cycle IMU source mode (QMI → LSM → FUSION). |
| B6 | Back / Emergency | Back in menu (exit PID menu). Outside menus: show system info page as a "nothing to back out of" acknowledgement. | EMERGENCY STOP — motors + servos off immediately, enters ERROR state until B1 is pressed to clear. |

Debug button events

If a button seems unresponsive or triggers the wrong action, enable the serial trace with `debug stream buttons`. Every press is printed with its decoded button number and event type. Press any key in the serial monitor to stop the stream.

Serial CLI Reference

The AIO32 exposes a full interactive CLI on USB-serial at **115200 baud**. Every runtime-tuneable parameter is reachable, plus manual test commands for motors, servos, LED and sound, plus live streaming debug dumps. Use `help` for the category list, `help <category>` for detailed per-command syntax in-console. This chapter documents every command with at least one example so you never need to reach for the source code.

Top-level commands

| Command | Purpose | Example |
|-------------------------------|---|----------------------------|
| <code>help</code> | Show category list | <code>help</code> |
| <code>help <cat></code> | Detailed help for one category | <code>help pid</code> |
| <code>status / st</code> | Full system snapshot: state, IMU, RC, battery, servos | <code>status</code> |
| <code>version / ver</code> | Firmware version + build info | <code>version</code> |
| <code>save</code> | Write all runtime values to NVS (survives reboot) | <code>save</code> |
| <code>load</code> | Re-read NVS values, overwriting runtime state | <code>load</code> |
| <code>reset factory</code> | Wipe all NVS keys, reboot with defaults | <code>reset factory</code> |
| <code>restart / reboot</code> | Software restart (same as RESET button) | <code>restart</code> |

pid — PID tuning

| Command | Purpose | Example |
|------------------|--|-----------------|
| pid show | Print all gains (base + 3 bands) + target + autotune state | pid show |
| pid kp <v> | Base Kp (used when adaptive off) | pid kp 28 |
| pid ki <v> | Base Ki | pid ki 0.0 |
| pid kd <v> | Base Kd | pid kd 0.8 |
| pid slow kp <v> | Slow-band Kp ($ \text{throttle} < 0.2$) | pid slow kp 30 |
| pid slow kd <v> | Slow-band Kd | pid slow kd 0.9 |
| pid med kp <v> | Medium-band Kp (0.2..0.5) | pid med kp 22 |
| pid med kd <v> | Medium-band Kd | pid med kd 0.7 |
| pid fast kp <v> | Fast-band Kp ($ \text{throttle} > 0.8$) | pid fast kp 18 |
| pid fast kd <v> | Fast-band Kd | pid fast kd 0.5 |
| pid target <deg> | Balance target angle (pitch offset) | pid target -0.3 |
| pid autotune | Start bounded slow-band Kp sweep, pick min-RMS | pid autotune |

imu — IMU, calibration, Madgwick

| Command | Purpose | Example |
|-----------------|--|-----------------|
| imu show | Active mode + live pitch/roll/yaw + health | imu show |
| imu mode qmi | Use only the QMI8658C sensor | imu mode qmi |
| imu mode lsm | Use only the LSM6DS3 sensor | imu mode lsm |
| imu mode fusion | Average both sensors (default) | imu mode fusion |
| imu mode auto | Fall back to whichever sensor is healthy | imu mode auto |
| imu cal | Run 5-second calibration (keep droid still) | imu cal |
| imu bias show | Print calibration offsets currently in use | imu bias show |
| imu bias reset | Clear offsets (next imu cal will write new ones) | imu bias reset |
| imu beta <v> | Madgwick gradient-descent gain (0.03..0.5) | imu beta 0.12 |
| imu beta show | Current beta value | imu beta show |

drive — driving dynamics

| Command | Purpose | Example |
|----------------------|--|---------------------|
| drive show | All driving-dynamic settings at once | drive show |
| drive mode arcade | Arcade mixing (CH1 = steering, CH2 = throttle) | drive mode arcade |
| drive mode tank | Tank mixing (CH1 = motor1, CH2 = motor2) | drive mode tank |
| drive mode cycle | Toggle arcade <-> tank (same as B5 long) | drive mode cycle |
| drive shaping on off | RC deadband + expo enable | drive shaping on |
| drive deadband <v> | Stick deadzone (normalised 0..0.5) | drive deadband 0.05 |
| drive expo <v> | Expo curve (0 = linear, 1 = heavy) | drive expo 0.35 |
| drive ramping on off | Motor-output low-pass enable | drive ramping on |
| drive ramp <v> | Ramp rate (0..1; lower = smoother acceleration) | drive ramp 0.15 |
| drive lean on off | Dynamic forward-lean enable (lean into acceleration) | drive lean on |
| drive leanmax <deg> | Max lean angle at full throttle (0..15) | drive leanmax 6 |

rc — protocol + channels

| Command | Purpose | Example |
|---------------------|--|------------------|
| rc | Shortcut for <code>rc status</code> | rc |
| rc status | Active + stored protocol, connection, packet counts | rc status |
| rc protocol | Show currently active protocol | rc protocol |
| rc protocol ibus | Switch to FlySky iBus (default, no inverter needed) | rc protocol ibus |
| rc protocol sbus | Switch to SBUS (ESP32 HW invert, no inverter needed) | rc protocol sbus |
| rc channels / rc ch | One-shot dump of all channel values | rc channels |

sound — DFPlayer

| Command | Purpose | Example |
|----------------------|--|-----------------|
| sound show | Current volume, mute flag, track count on SD | sound show |
| sound volume <0..30> | Set DFPlayer volume | sound volume 22 |
| sound mute | Mute output (transmitter can toggle too) | sound mute |
| sound unmute | Unmute | sound unmute |
| sound test <n> | Play /mp3/NNNN.mp3 immediately | sound test 3 |
| sound stop | Stop current playback | sound stop |

display — TFT

| Command | Purpose | Example |
|-------------------------|---|------------------------|
| display show | Current rotation + mode | display show |
| display rotate <0..3> | Set fixed rotation (persists) | display rotate 1 |
| display rotate cycle | Next rotation (same as B3 short) | display rotate cycle |
| display mode telemetry | Live balance / battery / PID page | display mode telemetry |
| display mode diag | Diagnostic page with raw IMU + RC values | display mode diag |
| display mode off | Blank the display (backlight stays) | display mode off |
| display alwayson on off | Keep display on while running (same as B3 long) | display alwayson on |

balance — state control

| Command | Purpose | Example |
|------------------|--|------------------|
| balance start | Enter STATE_RUNNING (PID active, motors live) | balance start |
| balance stop | Back to STATE_READY (motors idle) | balance stop |
| balance estop | Emergency stop: motors cut, STATE_ERROR (same as B6 long) | balance estop |
| balance baseline | Auto-null target angle — 2-second pitch average becomes new target (same as B1 long) | balance baseline |

feature — runtime flags

Four flags are runtime-toggable. Two more (`adaptive_pid` and `madgwick`) are compile-time flags set in `config.h` — `feature show` displays them so you know the current build, but they cannot be flipped without re-flashing.

| Command | Purpose | Example |
|--|--|------------------------------------|
| <code>feature show</code> | All flags (runtime + compile-time) and their current state | <code>feature show</code> |
| <code>feature state_rx on off</code> | Tilt-warn + recovery sound triggers | <code>feature state_rx on</code> |
| <code>feature idle_sound on off</code> | Idle mood sound triggers | <code>feature idle_sound on</code> |
| <code>feature idle_anim on off</code> | Idle servo animations (mainbar + head) | <code>feature idle_anim on</code> |
| <code>feature sound on off</code> | Master sound enable (also toggles DFPlayer mute) | <code>feature sound on</code> |

test — manual hardware test

| Command | Purpose | Example |
|---|---|-------------------------------------|
| <code>test motor1 <-255..255></code> | Spin motor 1 at signed PWM value | <code>test motor1 120</code> |
| <code>test motor2 <-255..255></code> | Spin motor 2 at signed PWM value | <code>test motor2 -80</code> |
| <code>test motors stop</code> | Stop both motors immediately | <code>test motors stop</code> |
| <code>test servo <name> <0..180></code> | Drive a servo: mainbar, head1, head2, head3 | <code>test servo mainbar 120</code> |
| <code>test servo center</code> | All four servos back to 90 deg | <code>test servo center</code> |
| <code>test led <r> <g> </code> | Set NeoPixel RGB (0..255 each) | <code>test led 0 255 0</code> |
| <code>test sound <n></code> | Alias for <code>sound test n</code> | <code>test sound 7</code> |

debug — dumps + live streams

| Command | Purpose | Example |
|---------------------------|---|-----------------------|
| debug rc | One-shot RC channel dump | debug rc |
| debug imu | One-shot raw IMU dump (accel + gyro per sensor) | debug imu |
| debug buttons | One-shot button status dump | debug buttons |
| debug batt | One-shot battery voltage + divider factor | debug batt |
| debug stream rc [ms] | Live RC stream (default 200 ms). Any key stops. | debug stream rc 100 |
| debug stream imu [ms] | Live IMU angles + gyro rates | debug stream imu 200 |
| debug stream buttons [ms] | Live button events (default 50 ms) | debug stream buttons |
| debug stream batt [ms] | Live battery voltage (default 1000 ms) | debug stream batt 500 |
| debug stream off | Stop the stream (any keystroke also stops) | debug stream off |

Tip: put a complete one-liner like `pid kp 28 kd 0.9 target -0.3 && save` at the end of a tuning session to lock in every change at once. The CLI accepts one command per line; chaining multiple on one line is not supported, but a sequence of independent commands separated by line-breaks works fine when pasted.

Balance and PID Tuning

D-O balances on two wheels. A PID controller takes the pitch error (measured pitch minus target pitch) and produces a motor correction that both motors apply equally. The AIO32 uses an **adaptive 3-band PID**: separate kp/kd gains for slow / medium / fast wheel speeds, linearly blended. This lets the droid be stable at standstill and agile at speed without one compromise value for both.

Recommended starting values

| Parameter | Default | Range |
|---------------|----------|---|
| kp_slow | 28.0 | 20..40 |
| kp_medium | 22.0 | 15..35 |
| kp_fast | 18.0 | 10..30 |
| kd_slow | 0.9 | 0.3..1.5 |
| kd_medium | 0.7 | 0.3..1.5 |
| kd_fast | 0.5 | 0.2..1.2 |
| ki (global) | 0.0 | 0..0.5 (leave 0 unless you see a steady lean) |
| target_angle | -0.3 deg | -2.0 .. +2.0 (baseline via B1-long) |
| max_integral | 400 | 200..800 |
| madgwick_beta | 0.10 | 0.05..0.20 |

Tuning procedure

- **Start on a stand**, wheels clear of the ground, 4S pack connected.
- **Baseline** the target angle first: long-press **B1 (Start/Stop)** or `balance baseline`. Pitch must read near 0 deg at rest.
- **Engage balance** with short-press **B1 (Start/Stop)** or `balance start`. Droid should make small twitches but not oscillate wildly.
- If it oscillates fast: reduce `kp_slow` in 2-unit steps until stable.
- If it sags then twitches back: increase `kp_slow`.
- If it rings when you push it: increase `kd_slow` in 0.1 steps.
- Once *slow* band is stable, repeat with *medium* and *fast* bands. For *fast*, put it on the floor and drive it; observe oscillation during cornering.
- Save with `save`.
- If the droid drifts forward or backward at rest even after baseline: adjust `pid target` in 0.1 deg steps. Small negative (-0.3 to -0.8) is typical.

Tip: `pid autotune` runs a bounded slow-band kp sweep and picks the value that minimises RMS pitch error. Useful as a starting point on a new droid, but always fine-tune by hand for your specific mechanical build.

Sound (DFPlayer Mini + SD)

The AIO32 drives a DFPlayer Mini over UART (GPIO 9 TX, GPIO 10 RX) using the Makuna **DFMiniMp3** library. File addressing is **filename-based** via `playMp3FolderTrack(N)`, so the SD card copy order does not matter — the chip looks up `/mp3/NNNN.mp3` directly.

Track map (identical to Mega line)

| Track | Function / Funktion |
|-----------|---|
| 0001 | Startup ("battery charged") |
| 0002 | Default ("I am D-O") |
| 0003–0005 | Greetings |
| 0006–0009 | Negative moods |
| 0010–0014 | Positive moods |
| 0015–0020 | Squeaky wheel |
| 0021 | Tilt warning (droid leaning >15 deg) |
| 0022 | Recovery (droid re-centred after tilt) |
| 0023 | Low battery |
| 0024–0030 | Idle sounds (played while in READY state for N seconds) |
| 0031 | System ready (post-boot) |
| 0032 | Signal lost |

SD card format + filenames

- File system: **FAT16** or **FAT32**. Recent Windows likes to format as exFAT — DFPlayer does NOT read exFAT.
- Folder: `/mp3/` (lowercase). Create it manually in the root if it is missing.
- File format: **MP3** or **WAV**. Both work. MP3 is smaller and fine for voice clips; WAV avoids any decoder quirks for very short effect sounds.
- Filename rule: the first **four digits** are the track number, everything after that is free. `0001.mp3`, `0001_startup.mp3`, `0001_battery_charged.wav` all resolve to track 1 — whatever you find readable.
- Leading zeros are required: `1.mp3` does NOT work; it must be `0001.mp3` (or similar with the four-digit prefix).
- Card size: 2 GB..32 GB works well. 64 GB+ typically defaults to exFAT — reformat to FAT32.
- Bitrate: 128..192 kbps mono MP3 is fine. Higher bitrates work but waste space.

Speaker wiring

The AIO32 board has **two speaker outputs**, each with its own **+** and **-** terminal. Both outputs are **driven directly by the DFPlayer Mini** — no external amplifier on the board, and none needed. The two outputs are wired in parallel to the DFPlayer mono audio stage, so they carry the same signal. You can use:

- A **single speaker** on either of the two outputs — the other output stays unused.
- **Two speakers in parallel** (one per output), for example one in the head and one in the body. Both play the same mono signal.

Recommended per-speaker spec: **8 Ω , 2-3 W, 28-40 mm diameter**. If you run two 8 Ω speakers in parallel the combined load is 4 Ω — still within the DFPlayer spec, but you may need to keep volume below max. Always observe the **+** / **-** marking when wiring two speakers in parallel so they move in phase (otherwise they cancel).

Volume and mute

Volume is 0..30. Default 25. Via CLI `sound volume 22`, `sound mute`, `sound unmute`, `sound test 5` (plays track 0005). Mute can also be toggled from the transmitter (CH7 — SwA by default) or with B3 (Display) — the latter only when the display is the active short-press target; the primary mute shortcut is the transmitter or the `sound mute` CLI.

Display and Status LED

The AIO32 has two visual indicators: a 1.14" colour TFT showing live telemetry and a single NeoPixel showing the current system state by colour and pattern.

Display modes

The display auto-oriens itself based on `imu mode` and last physical orientation, or stays fixed if you choose. Three rotation options cycled by **B5-short** (or `display rotation auto|portrait|landscape`):

| Mode | Use case |
|-----------|---|
| auto | Display follows droid orientation — useful during build/test on the bench |
| portrait | Fixed portrait (240 tall, 135 wide). Main runtime mode |
| landscape | Fixed landscape (240 wide, 135 tall). Useful during service on a cradle |

Always-on vs auto-sleep

B5-long toggles always-on. When off, the display sleeps after 5 minutes of inactivity in READY state (never sleeps in RUNNING). Long-term always-on on OLED-like displays can cause burn-in; the ST7789 is not susceptible but the backlight LEDs last longer if they are off when idle.

NeoPixel state patterns

| State | Colour & pattern |
|------------------------------------|--|
| INIT | Yellow solid |
| CALIBRATING | Cyan pulse (1.2 s period) |
| READY | Green solid (or orange pulse when <code>batteryWarnActive</code>) |
| RUNNING | Blue pulse (2 s period) |
| ERROR | Red solid |
| EMERGENCY_STOP | Red slow blink (400 ms) |
| <code>batteryCriticalActive</code> | Red fast blink (200 ms) — overrides all other patterns |

Battery Monitoring

The AIO32 measures pack voltage through a 47k/10k divider on GPIO 16 (ADC2). Warn and critical thresholds are configurable; on critical a red-fast-blink NeoPixel pattern takes over and motors are gated off to protect the LiPo. Battery monitoring ships disabled by default — enable it via the `BATTERY_MONITOR_ENABLED` compile flag in `config.h` once you have verified the divider factor matches your pack.

Configuration

| Parameter | Default | Range |
|--------------------------------------|---------|---|
| <code>BATTERY_MONITOR_ENABLED</code> | false | true once divider is active |
| <code>battery_warning</code> | 13.6 V | 3.4 V per cell x 4 |
| <code>battery_critical</code> | 12.8 V | 3.2 V per cell x 4 |
| <code>voltage_divider</code> | 5.74 | compute as $(R1+R2)/R2$; default matches 47k/10k |

Calibration

To calibrate the divider factor precisely: measure actual pack voltage with a multimeter, then in CLI run `debug batt` to see what the ADC is reading. If the reading is off by X%, adjust `voltage_divider` by the same %. Save. Re-check.

Part III

Reference & Appendices

Troubleshooting

First resort for any issue: `status` in the CLI gives a full snapshot of IMU, RC, battery, PID, state. Second resort: `debug stream <kind>` for live data.

Boot hangs at "DFPlayer init..."

- SD card missing or not FAT32 — insert formatted card and reboot
- DFPlayer TX/RX swapped — GPIO 9 is AIO32 TX (goes to DFPlayer RX via 1k resistor), GPIO 10 is AIO32 RX (from DFPlayer TX direct)
- Workaround to get past boot: set `SOUND_CONTROLLER_ENABLED=false` in `config.h`, reflash, fix wiring, re-enable

Droid leans at rest even after calibration

- Run `balance baseline` (or long-press B1) with droid perfectly upright
- If still off, adjust `pid target` in 0.1 deg steps
- Check physical wheel diameter symmetry — unequal wheels tilt the target angle

RC shows no channels / all zero

- `rc status` should show `connected=Yes`. If No: check wiring to GPIO 2.
- iBus: verify receiver is bound and transmitter is on
- SBUS: `rc protocol sbus activated? save done?`
- `debug stream rc` to see live bytes. Channels should move 1000..2000 as sticks move

Buttons trigger wrong actions

- `debug stream buttons` to see live button events
- If a button never fires: check wiring at the B1..B6 header
- If the wrong button number is reported: contact support with the debug output — the decoding thresholds may need adjustment for your batch

Servo twitches violently at boot

- LEDC channel collision or interrupted power. Check 5V BEC can deliver enough current for all four servos at stall
- Verify all servo grounds are tied to the 5V BEC ground AND to the AIO32 ground

Display shows garbage or black

- ST7789 uses SPI on AIO32-internal pins; likely a bad connection of the TENSTAR module header
- Try `display mode splash` to force a known test pattern

Tilt warn fires at wrong angle (e.g., 15 deg reported as 5 deg)

- Madgwick quaternion is off — re-run `imu cal`
- Lower `imu beta` to 0.05 temporarily and observe stability
- Try a different IMU mode: `imu mode qmi` vs `imu mode lsm` vs `imu mode both` — one sensor may be contributing noise

NVS reads nonsense after factory reset

- Should not happen — factory reset wipes NVS partition. If it does: reflash full firmware (not just app partition) and re-calibrate IMU

Safety

The full liability disclaimer is on page 2. This chapter lists the practical safety rules for operating D-O.

| Rule | Details |
|------------------|---|
| LiPo handling | Use a LiPo-safe charger. Never charge unattended. Replace swollen packs. |
| Polarity | Verify every power connection before power-on. Reverse polarity can damage the ESP32 or motor driver instantly. |
| Voltage | 4S LiPo: 16.8 V full, 14.8 V nominal, 12.0 V empty. Never discharge below 3.0 V per cell. |
| Servo power | Use a separate 5-6 V BEC for servos (3-5 A). Do not power servos from the ESP32 3.3 V rail. |
| First power-on | Wheels off the ground. The PID can swing the droid violently during calibration and tuning. |
| Emergency stop | Long-press B6 (Back/Emergency) cuts motors and servos immediately. Practice reaching for it before first floor-driving. |
| Environment | Clear the area. A tipping droid at 4S current can hurt hands, pets, or your hardware. |
| SD card handling | DFPlayer reads FAT16/FAT32 only. Always safely eject before re-formatting — corrupt SD cards cause boot hangs. |