Package: axR (via r-universe)

July 16, 2026

Title Serial Communication and Data Retrieval for Axivity Devices

Version 0.1.2

Description Serial communication and data retrieval for Axivity AX3/AX6 accelerometer devices: device discovery, status, settings, data download, and reading recorded '.cwa'/AX6 binary files. Wraps the Open Movement Project's 'OMAPI' C library (vendored in src/omapi, BSD 2-clause, Newcastle University; see src/omapi/LICENSE.TXT), rather than reimplementing the serial protocol or binary file format directly. On Linux, device discovery requires 'libudev-dev' (Debian/Ubuntu) or 'systemd-devel' (Fedora/RHEL) to be installed separately -- see SystemRequirements. macOS and Windows use OS frameworks/APIs directly, with no separate install step. Part of the Circadia Lab R ecosystem.

License MIT + file LICENSE

Encoding UTF-8

Roxygen list(markdown = TRUE)

Depends R (>= 4.1.0)

Imports Rcpp (>= 1.0.0)

LinkingTo Rcpp

SystemRequirements libudev

Suggests covr, pkgdown, testthat (>= 3.1.4), tibble, knitr, rmarkdown

VignetteBuilder knitr

Config/testthat/edition 3

Lifecycle experimental

URL https://axr.circadia-lab.uk, https://github.com/circadia-bio/axR

BugReports https://github.com/circadia-bio/axR/issues

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

Repository https://circadia-bio.r-universe.dev

Date/Publication 2026-07-16 12:14:51 UTC

RemoteUrl https://github.com/circadia-bio/axR

RemoteRef v0.1.2

RemoteSha 893b154bed8abe50f0a11ce1580df39dc2915606

Contents

`axivity_copy_data` *Copy recorded data directly off a mounted Axivity volume*

Description

A fallback for when `axivity_discover()`/OMAPI device access isn't working, but the device's USB mass-storage volume mounts and is visible in Finder/`diskutil` regardless – a common split, since the storage side and OMAPI's own IOKit-level device discovery are independent paths (see `NEWS.md` for the discovery issues hit so far). This bypasses OMAPI and `device_id` entirely: it's a plain file copy, nothing more. `axR` doesn't parse `.cwa` contents – see `mrpheus` or `zeitR` for that.

Usage

```
axivity_copy_data(
  device_path,
  dest_dir,
  pattern = "\\.cwa$",
  overwrite = FALSE
)
```

Arguments

device_path	Character. Path to the mounted device volume, e.g. "/Volumes/CWA17_46171". Find this in Finder, or with list.files("/Volumes").
dest_dir	Character. Destination directory for downloaded files. Created (recursively) if it doesn't already exist.
pattern	Character. Regex to filter which files are copied, matched case-insensitively. Default "\\.cwa$".
overwrite	Logical. Overwrite existing files at the destination. Default FALSE.

Value

Character vector of destination file paths that were successfully copied.

---

axivity_discover *Discover connected Axivity devices*

Description

Wraps OMAPI's OmGetDeviceIds() plus a handful of per-device status calls into a single data frame. Unlike serial-port probing, this uses OMAPI's own device discovery – including its platform-specific finder (IOKit/DiskArbitration on macOS, SetupAPI on Windows, udev on Linux) – so it should behave the same way OmGui does on the same machine.

Usage

```
axivity_discover()
```

Value

A data frame with one row per connected device: device_id, serial, port, path, firmware_version, hardware_version, battery_level. Zero rows if no devices are connected.

axivity_download — *Download recorded data off an Axivity device*

Description

Wraps OMAPI's `OmBeginDownloading()`, which runs the download on a background thread inside the library. `axR` doesn't parse `.cwa` contents – see `mrpheus` or `zeitR` for that.

Usage

```
axivity_download(
  device_id,
  dest_file,
  offset_blocks = 0L,
  length_blocks = -1L,
  blocking = TRUE
)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`dest_file`	Character. Destination file path (existing files at this path are truncated).
`offset_blocks`	Integer. Start offset of the download, in blocks. Default `0`.
`length_blocks`	Integer. Length to download, in blocks. Default `-1` (all).
`blocking`	Logical. If `TRUE` (default), block until the download completes, fails, or is cancelled – equivalent to calling `axivity_download_wait()` immediately after. If `FALSE`, return as soon as the download starts; poll with `axivity_download_status()`.

Value

If `blocking = TRUE`, a list with `status` (one of `"complete"`, `"error"`, `"cancelled"`) and `value` (a diagnostic code if `status` is `"error"`). If `blocking = FALSE`, invisibly `NULL`.

axivity_download_cancel — *Cancel an in-progress download from an Axivity device*

Description

Cancel an in-progress download from an Axivity device

Usage

```
axivity_download_cancel(device_id)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().

---

axivity_download_status

Check or wait for an Axivity device's download progress

Description

Check or wait for an Axivity device's download progress

Usage

```
axivity_download_status(device_id)

axivity_download_wait(device_id)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().

Value

A list with status ("none", "error", "progress", "complete", or "cancelled") and value (percentage complete if status is "progress", a diagnostic code if "error").

---

axivity_enable_debug_log

Enable OMAPI's internal debug log

Description

A diagnostic escape hatch for when axivity_discover() isn't finding a device you expect it to. OMAPI logs internally via its own OmLog(), but axR's default log target is NULL (so compiled code doesn't write to stderr unprompted – see NEWS.md). This re-enables it.

Usage

```
axivity_enable_debug_log(file = NULL)
```

Arguments

file	Character path to a log file, or NULL (default) to log to stderr. A file is more reliable for diagnostic purposes: OMAPI's discovery thread logs from a background pthread, and raw stderr writes from a non-R thread don't always reach the R console/terminal depending on the frontend – a file sidesteps that ambiguity.

Details

Important: this only controls where log lines go. Whether anything is logged *at all* is controlled by OMAPI's debug level, which is read from the `OMDEBUG` environment variable once, at `OmStartup()` time – i.e. before `library(axR)` runs. If you're not seeing log output after calling this, set `OMDEBUG` (e.g. `Sys.setenv(OMDEBUG = "3")`) *before* loading axR, in a fresh R session, then call this function and retry.

Value

Invisibly, the OMAPI status code (negative indicates failure, e.g. the file couldn't be opened).

`axivity_get_accel_config` *Get or set an Axivity device's accelerometer configuration*

Description

Get or set an Axivity device's accelerometer configuration

Usage

```
axivity_get_accel_config(device_id)

axivity_set_accel_config(device_id, rate, range)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`rate`	Sampling rate in Hz (6 = 6.25, 12 = 12.5, 25, 50, 100, 200, 400, 800, 1600, 3200). Negative = low-power mode.
`range`	Sampling range in +/- G (2, 4, 8, 16).

Value

`axivity_get_accel_config()` returns a list with `rate` (Hz) and `range` (+/- g).

`axivity_get_accelerometer` *Read an Axivity device's current accelerometer values*

Description

Read an Axivity device's current accelerometer values

Usage

```
axivity_get_accelerometer(device_id)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.

Value

A named numeric vector `c(x, y, z)` in units of *g* (raw values are in 1/256 *g*, converted here).

`axivity_get_battery` *Query an Axivity device's battery level and health*

Description

Query an Axivity device's battery level and health

Usage

```
axivity_get_battery(device_id)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.

Value

A list with `level_pct` (0-99% = charging, 100% = full) and `recharge_cycles` (lower is better).

axivity_get_data_info *Get information about an Axivity device's recorded data*

Description

Get information about an Axivity device's recorded data

Usage

```
axivity_get_data_info(device_id)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().

Value

A list with size_bytes, filename (path on the device's own filesystem, not yet downloaded), block_size, offset_blocks, num_blocks, and start/end (POSIXct, the time range of the recorded data).

axivity_get_delays *Get or set an Axivity device's delayed activation window*

Description

Get or set an Axivity device's delayed activation window

Usage

```
axivity_get_delays(device_id)

axivity_set_delays(device_id, start, stop)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().
start, stop	Each either a POSIXct (or coercible), -Inf (always record from now / OMAPI's zero sentinel), or Inf (never record / OMAPI's infinite sentinel).

Value

axivity_get_delays() returns a list with start and stop, each either a POSIXct, -Inf (OMAPI's "always"/zero sentinel), or Inf (OMAPI's "never"/infinite sentinel).

`axivity_get_ecc` *Get or set an Axivity device's error-correcting code (ECC) flag*

Description

Get or set an Axivity device's error-correcting code (ECC) flag

Usage

```
axivity_get_ecc(device_id)

axivity_set_ecc(device_id, enabled)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`enabled`	Logical. Enable or disable ECC.

Value

`axivity_get_ecc()` returns a logical.

`axivity_get_memory_health` *Query an Axivity device's NAND flash memory health*

Description

Query an Axivity device's NAND flash memory health

Usage

```
axivity_get_memory_health(device_id)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.

Value

A list with `spare_blocks` (higher is better, `0` = unusable) and `status` (`"ok"`, `"warning"`, or `"error"`, using OMAPI's documented thresholds).

axivity_get_metadata *Get or set an Axivity device's metadata scratch buffer*

Description

Get or set an Axivity device's metadata scratch buffer

Usage

```
axivity_get_metadata(device_id)

axivity_set_metadata(device_id, metadata)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().
metadata	Character. Metadata to store (up to 448 bytes; longer values are truncated by the device). URL-encode first if it needs to preserve non-ASCII characters.

Value

axivity_get_metadata() returns a character string, trimmed of trailing padding.

axivity_get_session_id *Get or set an Axivity device's session identifier*

Description

Get or set an Axivity device's session identifier

Usage

```
axivity_get_session_id(device_id)

axivity_set_session_id(device_id, session_id)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().
session_id	A value to set as the session ID.

Value

axivity_get_session_id() returns a numeric (session IDs can exceed R's 32-bit integer range).

axivity_get_time *Get or set an Axivity device's real-time clock*

Description

Get or set an Axivity device's real-time clock

Usage

```
axivity_get_time(device_id)

axivity_set_time(device_id, time)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().
time	A POSIXct (or coercible) date/time to set on the device.

Value

axivity_get_time() returns a POSIXct.

axivity_is_locked *Check or set an Axivity device's anti-tamper lock*

Description

Check or set an Axivity device's anti-tamper lock

Usage

```
axivity_is_locked(device_id)

axivity_set_lock(device_id, code)

axivity_unlock(device_id, code)
```

Arguments

device_id	Integer device ID, as returned by axivity_discover().
code	Integer lock code. 0 = no lock; 0xffff is reserved.

Value

axivity_is_locked() returns a list with locked and has_lock_code (logicals).

`axivity_read_cwa` *Read a .cwa/AX6 binary file into a tibble*

Description

Uses OMAPI's own binary file reader (`omapi-reader.c`, vendored from libomapi) to parse a `.cwa`/AX6 recording directly, rather than reimplementing the binary format from scratch. Returns one row per sample – ready to hand to `zeitR`, or any other downstream actigraphy analysis.

Usage

```
axivity_read_cwa(path)
```

Arguments

`path`	Character. Path to a `.cwa`/AX6 file.

Details

Unlike the rest of axR, this function doesn't talk to a live device at all – it works on a file already on disk (e.g. one retrieved with `axivity_copy_data()` or `axivity_download()`), and doesn't require `axivity_discover()` to have found anything.

Value

A tibble (or plain data frame, if the `tibble` package isn't installed) with one row per sample:

timestamp `POSIXct`, UTC, with sub-second precision

x, y, z Accelerometer readings, in g

gx, gy, gz Gyroscope readings, raw units (only present if the recording has a gyroscope, e.g. AX6 in GA/GAM mode)

mx, my, mz Magnetometer readings, raw units (only present if the recording has a magnetometer, e.g. AX6 in GAM mode)

light Raw light sensor reading

temperature_c Temperature in degrees Celsius. **Unverified, possibly wrong** – OMAPI's conversion (`OM_VALUE_TEMPERATURE_MC`) hardcodes a formula for one specific temperature sensor chip (MCP9700); a comment beside it in the vendored source notes an alternate formula for a different chip (MCP9701), suggesting this may be hardware/revision-specific. Cross-check against OmGui's own reading for the same file before relying on this.

battery_pct Battery percentage, at the time of this block

sample_rate Sampling rate in Hz, at the time of this block

with `device_id`, `session_id`, and `metadata` attached as attributes. `timestamp`, `x`/`y`/`z`, and `device_id` have been verified correct against a real AX3 file (cross-checked device_id against `ioreg` and the Axivity config web tool); `temperature_c` has not.

axivity_reset *Erase an Axivity device's data storage and commit settings*

Description

Wraps OMAPI's `OmEraseDataAndCommit()`. Staged settings changes (delays, session ID, metadata, accelerometer config) only take full effect when this is called.

Usage

```
axivity_reset(device_id, level = "quickformat")
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`level`	One of `"none"` (commit metadata only, not recommended – can cause a data/metadata mismatch), `"delete"` (remove and recreate the data file), `"quickformat"` (recreate the filesystem), or `"wipe"` (clear all NAND blocks, cleanest but slowest).

axivity_self_test *Run an Axivity device's built-in self-test*

Description

Run an Axivity device's built-in self-test

Usage

```
axivity_self_test(device_id)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.

Value

A list with `passed` (logical) and `diagnostic_code` (an opaque, firmware-specific code; `0` means passed).

axivity_send_command *Send a raw command to an Axivity device*

Description

An escape hatch wrapping OMAPI's `OmCommand()`, for anything not covered by axR's typed functions. Not generally recommended – OMAPI's own docs note that incorrect use could lead to unspecified behaviour.

Usage

```
axivity_send_command(device_id, command, expected = "", timeout_ms = 2000L)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`command`	Character. The command string to send.
`expected`	Character. The expected response prefix, or `""` if not specified.
`timeout_ms`	Integer. Timeout in milliseconds. Default `2000`.

Value

Character. The device's raw response.

axivity_set_led *Set an Axivity device's LED colour*

Description

Set an Axivity device's LED colour

Usage

```
axivity_set_led(device_id, colour)
```

Arguments

`device_id`	Integer device ID, as returned by `axivity_discover()`.
`colour`	One of `"auto"` (default device-controlled behaviour), `"off"`, `"blue"`, `"green"`, `"cyan"`, `"red"`, `"magenta"`, `"yellow"`, `"white"`.

Index