TECHNICAL REFERENCE · SOFTWARE, PERIPHERALS & TRACK OPERATIONS

**GULF BLUE CONCEPTS**
Applied Vehicle Aerodynamics

# Peripherals, Track Management, In-Car Telemetry & Flagging

A software platform for running a track event and the vehicle on it — event and flag control, in-car telemetry, driver analysis, and a hardware controller for the peripherals that need one.

DOCUMENT CLASS
Platform technical reference

SUBSYSTEMS COVERED
Software · Peripherals · Track Ops

PREPARED FOR
Prospective teams & partners

REVISION
2026-09-10 · Firmware baseline

## 01 Executive summary

GBC is a software platform for running a track day, a club race weekend, or a full sanctioned event. It is built software-first: a cloud platform handles event setup, flag control, pit-wall communication, and driver data, and every other piece of the system plugs into that same platform rather than standing apart from it.

The active wing described later in this document is the clearest example of what GBC calls a **peripheral** — on-vehicle hardware the software can configure and monitor in real time. It is one peripheral the platform ships today, not the platform itself, and the architecture behind it is deliberately general enough to support others.

A second, independent half of the platform has nothing to do with any one car: **Track Management**, **Flagging**, and **Race Engineering** operate at the level of an entire event, run entirely in a browser, and need nothing installed to use.

Across both halves, the same design rule holds: anything that only depends on data set up ahead of time — a wing's positions, a mapped track, a set of sector points — is built to run standalone on the vehicle's own hardware. Anything that depends on a live decision made by a person elsewhere — a race engineer's message — is relayed live, because no on-vehicle controller can originate that decision on its own. Every session the platform touches, on either side of that line, is captured with millisecond precision and available for review the moment it ends.

Architecture - 02

# How the pieces connect

One vehicle-side controller, one phone link, one cloud platform serving three different audiences.

A peripheral like the wing is a local, self-powered controller. It talks to exactly one other device over the air — the driver's phone, over Bluetooth Low Energy — and to nothing else. That is a design choice, not a limitation: the actuator and the sensors that drive it stay usable even when there's no signal at all in the paddock.

The phone is the bridge. The mobile application configures the peripheral, streams its live telemetry up to the GBC cloud over a standard encrypted web connection, and relays pit-wall messages back down to the vehicle over the same short-range link. When the phone is present, everything works together. When it isn't, the peripheral keeps doing whatever it was last told to do on its own.

The cloud platform presents three distinct faces depending on who's using it: **Driver Analysis** for session review and setup, **Race Engineering** for a crew working a pit wall, and **Track Management** for the people running an entire event across every car on track. All three are ordinary browser-based views — nothing about using them requires installing anything.

**Figure 1.** The vehicle-side controller speaks to one peer, the phone, over a short-range link. All wide-area connectivity — and every audience the platform serves — routes through that phone link, never around it.

Hardware - 03

# Peripherals

The active wing, today's flagship example: a dual-chip controller, an inertial sensor, a GPS receiver, and an actuator driver, running its own logic loop independent of any phone.

## 3.1 Sensor suite

The controller carries a six-axis inertial measurement unit and a GPS receiver, and drives the wing's linear actuator through a closed-loop position controller.

- **Inertial measurement.** Three-axis accelerometer and three-axis gyroscope, sampled continuously. Lateral acceleration — the axis that matters for cornering-triggered wing behavior — is read directly from the sensor rather than estimated from chassis roll angle, and drives the on-screen G-meter directly.
- **GPS positioning.** A dedicated GPS receiver provides latitude/longitude, ground course, and fix-quality, independent of the phone's own location services. This is what makes GPS-based features — track mapping, lap timing, sector splits — available to the controller with the phone merely present, not actively computing anything.
- **Actuator feedback.** The wing's actuator is driven by a closed-loop controller that tracks a commanded target position against the actuator's own reported position, rather than an open-loop timed pulse.
- **Mounting calibration.** A one-time calibration routine re-references the accelerometer to however the controller is physically mounted, with a built-in sanity check that rejects a calibration attempt if the unit reports being tilted more than 20° at the moment of zeroing — catching a mis-mounted unit rather than silently baking in a bad reference.

## 3.2 Standalone operation

Once a configuration is pushed to the controller, it is stored in the controller's own on-board memory — not held in the phone's app state. Wing waypoints, a mapped track outline, a start/finish line, and sector points all persist on the vehicle across power

cycles and connection drops. Track Mode and Custom Mode both run their logic on the controller's own GPS and inertial readings; the phone is a configuration and telemetry channel, not the thing doing the flying.

> DESIGN PRINCIPLE
>
> A feature belongs on the vehicle, standalone, if it's driven by data that's fixed once it's uploaded — waypoints, a track outline, a sector list. A feature can only ever depend on a live phone connection if it's driven by a live human decision somewhere else — a pit-wall call — because the controller has no way to learn about that on its own. Section 17 lists every current capability against this split.

## 3.3 Configuring the peripheral

Wing Configuration lets a driver define discrete positions and push them to the controller directly over the live connection; once written, that configuration lives on the vehicle and plays back with or without the phone attached. Custom Mode layers live G-force triggering on top: the app watches the controller's real-time accelerometer stream and commands wing movement in response to cornering load, braking, or straight-line thresholds the driver sets. A driver can also record or select a mapped GPS track — a personal track library — and push its geometry to the controller the same way; once loaded, the vehicle has that track's start/finish and sector points and can act on them from its own GPS fix.

## 3.4 Onboard lap & sector timing

The controller times itself. Given a start/finish point and, optionally, a run of intermediate sector points, the firmware cycles through that sequence of targets lap after lap: each GPS crossing is timestamped, the interval since the previous crossing is computed on the vehicle, and the result is appended to an onboard log. None of this requires the phone to be connected; it requires only that the geometry was uploaded at some point beforehand.

Crossing detection uses a two-radius geofence around each target — an outer "arm" radius and an inner "trigger" radius — with a minimum time gate between crossings to reject GPS noise from re-triggering the same point. The same two-radius approach is used in the cloud for phone-recorded sessions (Section 15); the on-vehicle version

substitutes a time-based gate for a sample-count gate, since it can't assume the phone's known GPS polling rate.

## 3.5 Wireless link

The controller and the phone talk over a low-power, short-range wireless link with a modest payload ceiling per message — a hard budget that shapes every telemetry field the firmware chooses to send. It is local by nature, which is consistent with the platform's broader stance that a vehicle's own safety behavior should never depend on anything reachable over the open internet.

### ● Runs standalone

- Wing waypoint playback (Track / Custom Mode)
- G-triggered actuation from live accelerometer data
- GPS track following once a track is uploaded
- Lap & sector timing, logged on the vehicle
- Mounting calibration

### ◐ Requires the phone

- Live telemetry streamed to the cloud
- Pit-wall messaging to/from the driver
- Initial configuration & track/sector upload

Green items need nothing but a prior upload. Amber items need a live phone connection, because they carry live, off-vehicle state the controller cannot observe on its own.

Car Dynamics - 04

# Car dynamics

What the platform knows about the actual car underneath a driver, and how that turns a generic starting tune into one built around this specific vehicle.

## 4.1 What the platform knows about the car

Beyond make, model, and year, a driver can enter their car's weight, front/rear weight distribution, and rear tire size (width, aspect ratio, wheel diameter — the same three numbers printed on the tire itself), along with a VIN for a free decode of body style, drivetrain, engine, and trim. None of this is required, and none of it is invented — this platform doesn't estimate curb weight or tire size from a lookup table, because a track car has almost always been modified from its factory spec by the time it's actually being driven hard: a stripped interior, a roll cage, different tires than it left the factory wearing. The numbers that matter are the real, current ones, entered once by the person who actually knows them.

## 4.2 From car specs to a starting tune

Peripheral control (Custom Mode, Section 3.3) runs on driver-set thresholds — how much lateral G, braking G, and acceleration G before the wing reacts, and a speed-based curve for how far it opens. Every driver starts from the same defaults today, regardless of what they're actually driving. Car dynamics data turns that into a real starting point instead of a blind guess, along three specific, physically-grounded relationships:

- **Tire width sets the G-thresholds.** A wider tire has a higher real grip ceiling, so what counts as "cornering or braking hard enough to want the wing" is a genuinely bigger G number on a wide-tired car than a narrow one. Thresholds scale with tire width relative to a reference car, not the same fixed number for every vehicle.
- **Weight sets how far ahead the wing anticipates.** A heavier car has more inertia and is slower to actually develop full G through a transition — corner entry, trail-braking — so a heavier car benefits from the wing anticipating slightly earlier. Lighter cars, with a shorter transient period, keep a tighter, more reactive lead time.

- **Weight distribution sets how aggressively the wing helps balance the car.** A front-heavy car tends toward understeer and has less inherent rear grip to begin with — a more assertive rear wing curve helps rotation. A rear-heavy car already has rear grip on its side and benefits from a gentler curve, since piling on more rear downforce on a car that's already rear-planted can over-stabilize it and hurt turn-in.

## 4.3 Suggested, never assumed

This produces a suggested starting tune, not an automatic one — the same principle that governs every configurable part of this platform (Section 16). A driver reviews the suggested thresholds and curve before anything is saved or sent to the vehicle, and can adjust any of it by hand, same as if they'd set every value themselves. Car dynamics data makes the first guess a genuinely informed one; it never removes the driver from the decision.

Capture - 05

# In-car telemetry

What's measured in the car, how it gets off the vehicle, and how it survives a bad connection on the way.

## 5.1 What's captured live

While a session is running, the vehicle streams its current state continuously: GPS position, cornering and braking G-force, the wing's commanded and actual position, and the onboard lap/sector timer's live state. That stream is the same data the controller uses for its own standalone behavior (Section 3) — nothing is computed twice or estimated differently for display versus for control.

## 5.2 Streaming & link resilience

The mobile app forwards that stream to the cloud continuously and buffers it locally when it can't — so a temporary gap between the phone and the GBC servers doesn't drop data, it just delays delivery until the connection returns. Configuration sent down to the vehicle over the short-range link is chunked and acknowledged for the same reason: a large upload, like a full track outline, survives a momentary connection hiccup rather than arriving corrupted or partial.

## 5.3 Who sees it, and when

The driver sees their own live state on their own device, including flag and pit-wall alerts as they happen (Sections 7 and 10). A race engineer watching the same session from a pit wall sees it in real time as well, with the ability to send something back (Section 10). Everything captured live is also the raw material for after-the-fact review in Driver Analysis (Section 11).

Operations - 06

# Track management

Every car on track at once, flag control, pit-wall communication, and lap/pit-speed monitoring across the whole field — run by whoever operates the event.

Track Management is event-level, not driver-level: it gives race control a single view of every car on track, flag control by zone or full-course, pit-wall communication to any car, and lap and pit-speed monitoring across the field. Everything below — setup, invites, running the session, and the flags available — happens inside that one system.

## 6.1 Setting up an event

An event operator starts from a venue — a track layout saved from a previous event, or a new one mapped once and kept on file afterward. Setup places two things on that layout: a run of sector-split points for lap timing, and a set of flag-zone stations, each covering one section of the circuit and able to raise its own local flag independent of every other zone. This is the one place both are defined; everything else in this document that uses them — the driver's on-screen banner, the onboard sector timer — draws on what's set up here (Section 16 covers this customization in more detail).

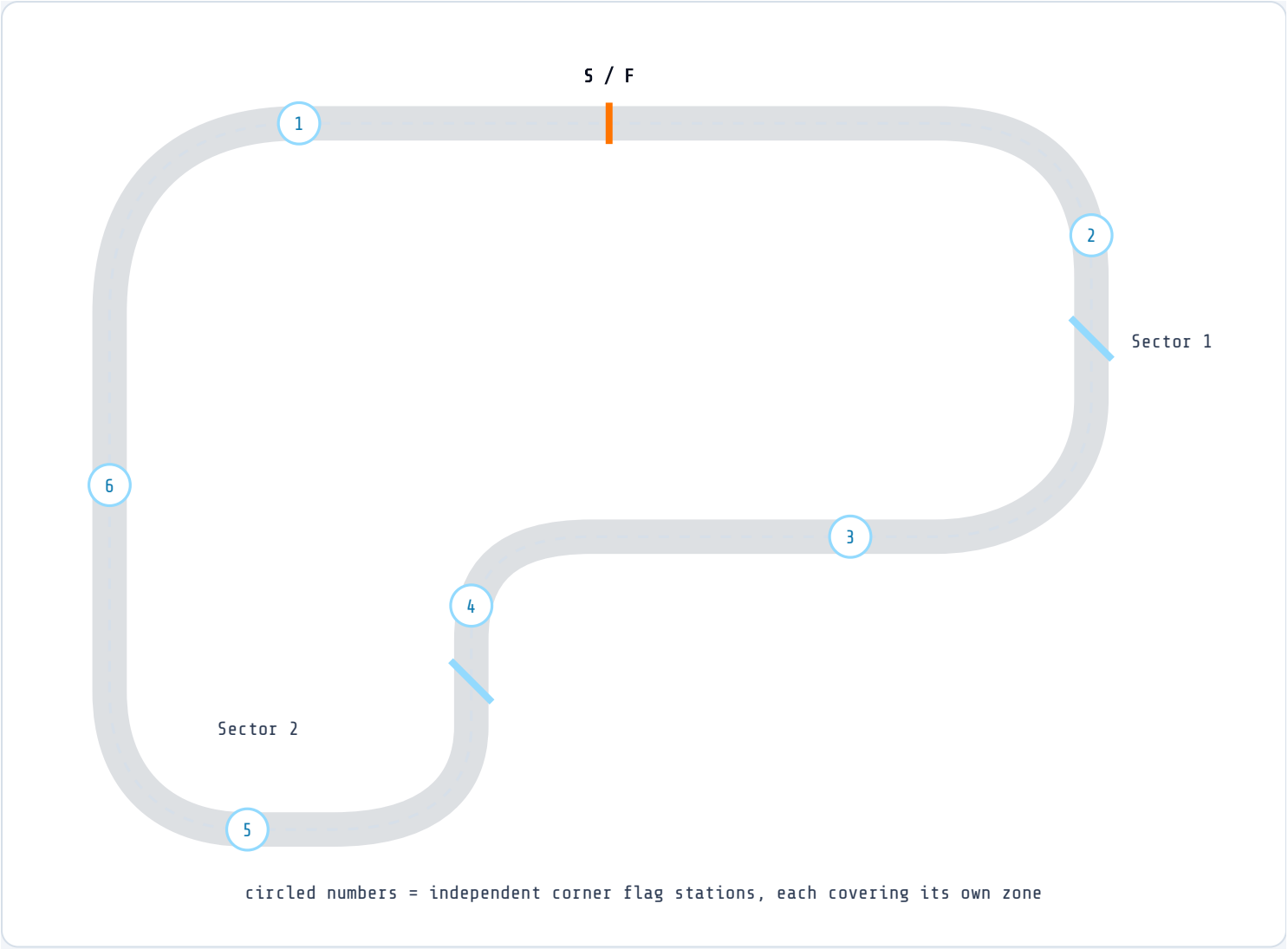
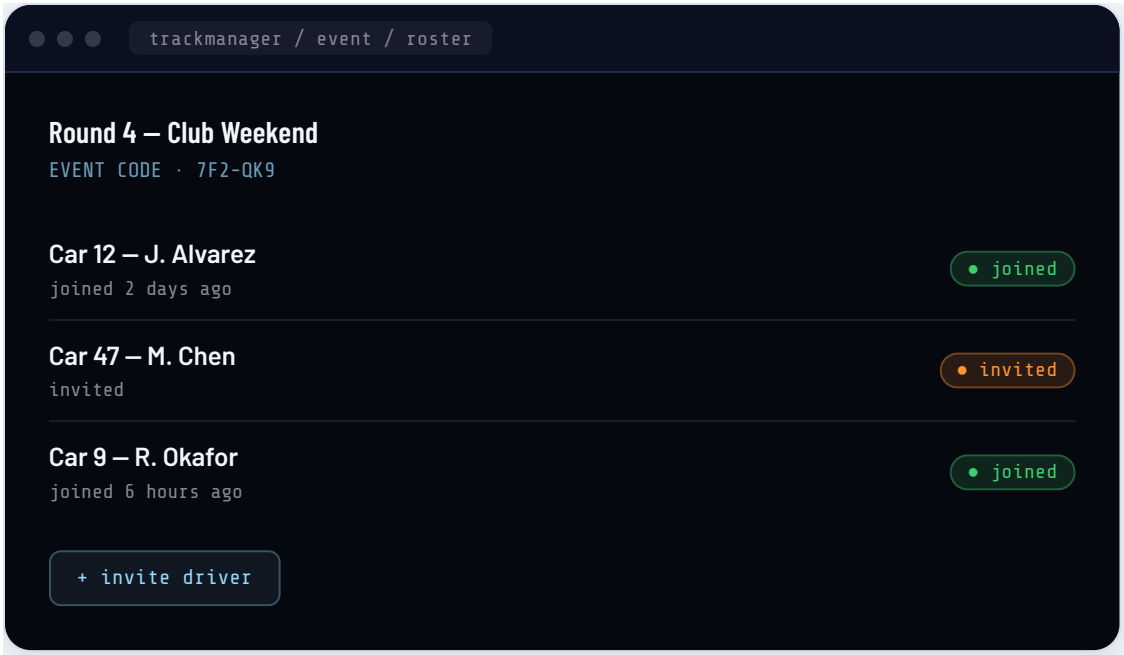



**Figure 2.** Sector splits and flag-station zones are placed once, on a representative circuit layout, when an event is set up — the same points the onboard sector timer and the flag system both use afterward.

## 6.2 Inviting drivers

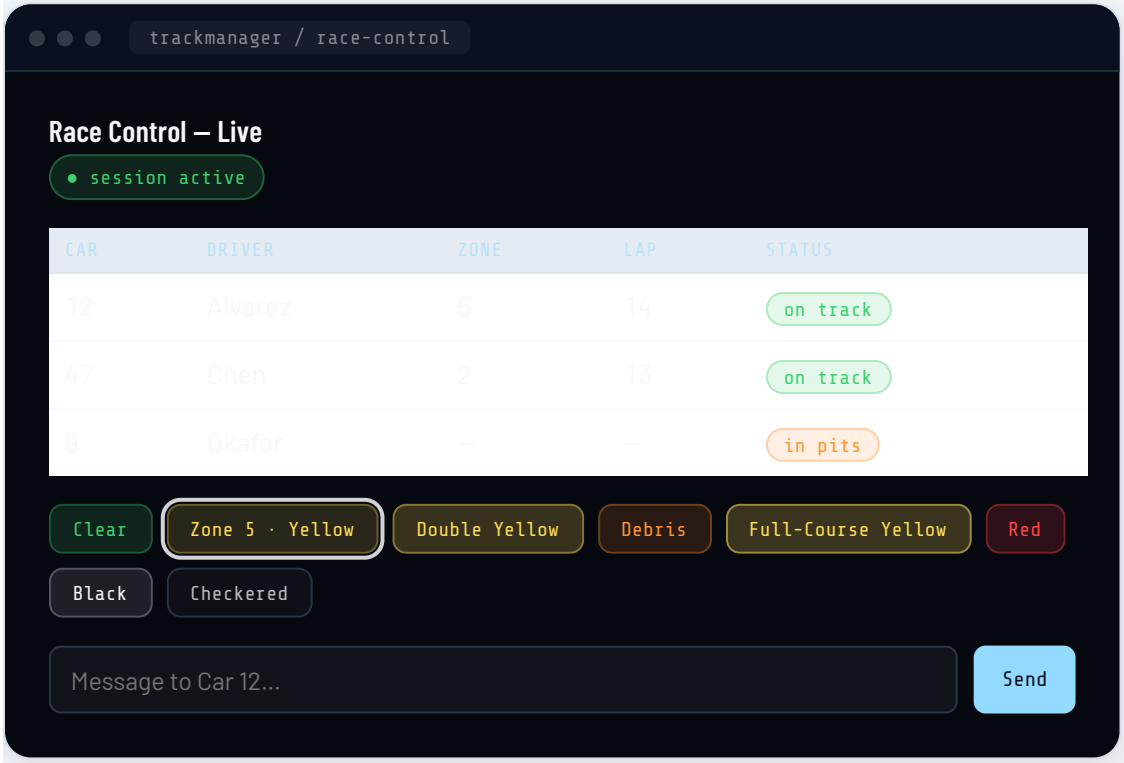
Drivers join a set-up event with an event code or a direct invite. Each driver's join status is visible to the operator as soon as they accept, and joining is what pulls that event's track, sector, and flag-zone layout down to the driver automatically — nothing about the specific venue has to be configured by hand on the driver's side (Section 13 covers this join flow in full).




**Interface, illustrative.** An operator invites drivers by code or direct invite; join status updates the moment each driver accepts.

## 6.3 Starting and running a session

Once drivers are on track, race control has a single live view of the whole field — every car's zone, lap, and status at once — and starts, monitors, and ends the session from there. Flag control and pit-wall messaging (Section 10) live in the same view, so calling a flag or reaching a specific car never means leaving the page that's watching the field.



**Interface, illustrative.** Race control sees every car at once, calls a flag by zone or full-course, and can message any car directly — all from the same view used to start and run the session.

## 6.4 Flags available

Race control can call any of the flags below, each scoped to a single zone or the full course (Section 7 covers exactly how each one reaches a driver, and why that scope is never blurred):

FLAGS AVAILABLE TO RACE CONTROL

| FLAG | SCOPE | MEANING |
|---|---|---|
| **Clear** | Zone or full-course | Condition has ended; normal racing resumes |
| **Yellow** | Zone | Caution in this zone only — no passing, be prepared to slow |
| **Double Yellow** | Zone | More severe caution in this zone — significant hazard, be ready to stop |
| **Debris** | Zone | Track surface hazard in this zone |
| **Full-Course Yellow** | Full course | Caution across the entire circuit, every car |
| **Red** | Full course | Session stopped |
| **Black** | Full course | Return to pits |
| **Checkered** | Full course | Session complete |

WHAT THIS MEANS FOR A TRACK OR SERIES

If you operate Track Management for your event, your participants do not need to have or install any software. Race control, corner marshal stations, and the event's pit wall all run as ordinary pages in a web browser. A driver only needs the companion mobile app if they want their own vehicle's live telemetry and peripheral control tied into the event — that's a choice each driver makes for their own car, not something the event operator has to distribute, install, or support across an entire field.

Race Control - 07

# Flagging

How a flag gets raised, who sees it, and what it's called.

## 7.1 Two levels of authority

The platform keeps a deliberate distinction between two different kinds of flag, end to end, because they mean different things to a driver:

- **Local, zone-level flags** — yellow, double yellow, and debris — raised by an individual corner station and scoped to that station's zone only. These tell a driver about a condition at one specific point on the circuit, not the whole track.
- **Full-course conditions** — full-course yellow, red, and black — called by race control for the entire circuit at once, through Track Management. These apply everywhere, to every car, regardless of where they are on track.

Collapsing that distinction is a real hazard, not just an imprecision: a driver who hears "full-course yellow" for what is actually a single marshal's local zone flag three corners behind them is being given worse information than no flag at all. The platform carries the distinction through every surface a flag touches — the on-screen banner, the spoken alert, and race control's own event view — so a local call is never announced as a full-course one, or the reverse.

## 7.2 Where a flag is raised

A corner marshal at a station raises a flag for their own zone from a simple station interface built for exactly that job: four large, unambiguous buttons, nothing to interpret under pressure. The interface reads the call back to the marshal who raised it — a spoken confirmation of what was just set — so they know their input registered correctly before turning back to the track. Race control raises full-course conditions from a separate, event-wide view in Track Management that shows every zone at once (Section 5.3).


**Interface, illustrative.** A marshal raises a flag for their own zone from four large buttons; the spoken readback confirms exactly what was just set before they turn back to the track.

## 7.3 Where a flag is seen and heard

Every flag, local or full-course, reaches the driver two ways at once: an on-screen banner naming the exact flag type and zone, and a spoken alert through the driver's own device — "Yellow Flag, Zone 5," "Double Yellow, Zone 5," "Debris, Zone 5," or, for a genuine event-wide call, "Full-Course Yellow" without a zone number, because it applies everywhere. A driver gets the same information without having to look away from the track to read it.


**Interface, illustrative.** The same call reaches the driver as an on-screen banner and a spoken alert, naming the exact zone, without the driver looking away from the track.

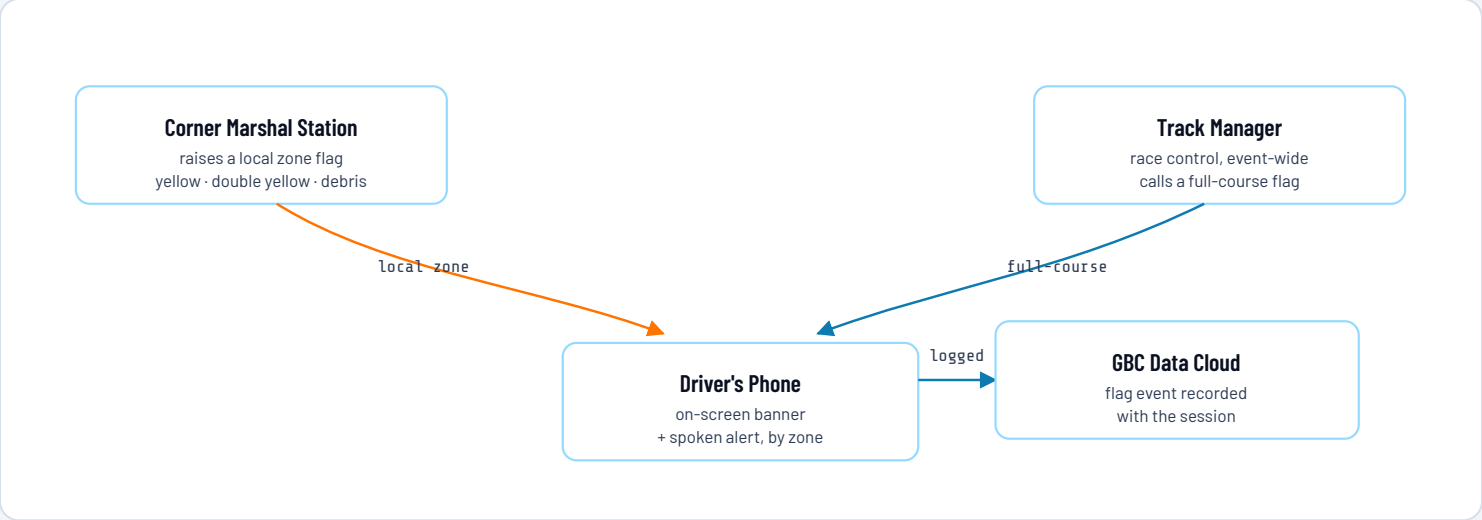



**Figure 3.** Local and full-course flags reach every driver the same two ways — banner and voice — and are logged to the cloud with the session. Flags are driver information only; they have no effect on the vehicle or its wing position.

## 7.4 Beyond flags: a direct line to the driver

A flag communicates a track condition to every affected driver at once — by design, it can't say anything more specific than that. For anything that needs to go to one driver specifically, Race Engineering (Section 10) gives a crew a separate, independent channel straight to their own car.

Driver - 08

# Advanced Mobile

The standalone-phone tier: a complete track session on the handset's own GNSS and inertial sensors, with no on-vehicle hardware in the loop.

## 8.1 What runs on the phone alone

Advanced Mobile is the entry tier of the platform: a native iOS and Android application that records and reviews a track session using only the handset's own hardware — its GNSS receiver for position and course, and its inertial measurement unit for cornering and braking G-force. No Black Box, no wiring, no bench setup. The application is a native shell around the same browser-based surfaces the rest of the platform uses, so a driver on the phone tier and a driver on the full peripheral stack work in the same session views, the same track library, and the same review tools.

## 8.2 Automatic venue detection and pre-loading

On launch, the app resolves the handset's position against GBC's maintained library of international circuits and loads the matching venue automatically — its start/finish line, sector-split geometry, and flag-zone layout — with no venue picker to work through. The circuit library is held server-side and expanded or corrected independently of app releases, so a newly mapped venue reaches every device without an update. A driver can also stage specific venues and custom tracks — track days, rallies, autocross, non-standard layouts — from the race-management control portal ahead of an event; staged tracks download to the device at login.

## 8.3 Session capture and crossing detection

During a run the app samples GNSS position and course at the receiver's fix rate and derives speed from it, alongside lateral and longitudinal G-force from the IMU. Lap and sector timing use the same two-radius geofence model as the on-vehicle controller — an outer arm radius and an inner trigger radius around each split point, with a minimum interval between crossings to reject GNSS noise (Section 15). Because the app knows its own polling rate, the on-device path gates re-triggering by sample count rather than by elapsed time. The completed session uploads to the cloud on pit-in and is available

for full review — position, speed, G-force, lap and sector splits — the moment the driver is back in the paddock, on any device, not only the one that recorded it (Section 11).

## 8.4 Flags on the phone tier

A phone-tier driver joined to a managed event receives the same flag layer as any other driver: local and full-course flags reach the handset as an on-screen banner and an on-device voice announcement, on the delivery path described in Section 7. Flags are driver information only and never act on a vehicle — on this tier there is no vehicle in the loop to act on. A paired Apple Watch uses haptics and text for flag alerts — the face turns the flag's colour and plays a per-flag haptic, so the call registers on the wrist without the driver looking at a screen. Section 9 covers that integration in full.

## 8.5 Where the Black Box changes things

Adding the Black Box peripheral does not change the event-level layer at all — the same track library, flag layer, and review tools apply. What changes is the capture path: the peripheral's dedicated 200 Hz IMU and GPS replace the phone's sensors, on-vehicle SD logging is added at 100 Hz, and wing control becomes available (Sections 3 and 5). Advanced Mobile is the tier that runs with none of that present, and nothing about a driver's setup, history, or event membership changes when a peripheral is later added.

## 8.6 Race Review and export

Every recorded session is listed, reviewable, renameable, and exportable from the Driver Race Review portal — standard and custom tracks alike, including DEs, rallies, and other event types. Sessions export in a widely used motorsport data format at hundredths-of-a-second precision, and GBC can supply a custom export layout for a specific downstream tool on request. This is the same review and export path used across the platform (Section 11), not a phone-tier-only feature.

Driver Data - 09

# Apple Watch

A paired Apple Watch as a second, glanceable surface for the one thing a driver cannot afford to miss on track: the flag that is out right now.

## 9.1 Purpose: a channel that needs no screen and no ears

Advanced Mobile (Section 8) already delivers every flag to the driver's phone as an on-screen banner and a spoken announcement. Both assume something: that the phone is visible on its mount, or that its audio is audible over the car. In a loud cockpit, with eyes on an apex, neither is guaranteed. A paired Apple Watch adds a third path that depends on neither — a haptic tap felt through the wrist and a full-face colour change read in well under a second. It is a mirror, never a source: the watch only ever displays a flag the platform has already called, and every decision about what flag is out, and where, stays with race control and the phone.

## 9.2 How the phone reaches the watch

The link is Apple's WatchConnectivity framework — a direct, peer-to-peer session between an iPhone and its paired Watch that carries no third-party server and needs no network once the two devices are paired. On the phone, a small bridge inside the native app shell receives each flag change from the running session and hands it straight to that session. On the watch, a companion app installed alongside the phone app receives it. There is no separate account, pairing step, or configuration for the driver: if the Watch is paired to the phone and the companion app is installed, the channel is live.

Each flag update is sent two ways at once. It is written as the session's **application context** — a single latest-value slot that the watch reads the moment it next wakes, so a watch whose screen was asleep when the flag was called still comes up showing the correct state rather than a stale one. And, whenever the watch is currently reachable, the same update is also pushed as a live **message** for immediate delivery. The redundancy is deliberate: messages are fast but only arrive if the watch is awake and in range; application context always survives, even across a backgrounded app or a dropped connection.

## 9.3 What a flag looks and feels like on the wrist

On a flag call the entire watch face fills with that flag's colour, and the watch plays a haptic pattern chosen so the flag can be told apart without looking at all. The driver's speed, run group or class, and the time of day stay on the face throughout, so a glance confirms both the flag and the basic context around it. When the condition clears, the face returns to the green (track-clear) state with no haptic — the absence of a buzz is itself the "all clear".

FLAG STATE ON THE WATCH

| FLAG | FACE COLOUR | HAPTIC |
|---|---|---|
| **Green / clear** | Green | None — colour change only |
| **Yellow, double yellow, full-course yellow** | Yellow | One pulse per second while the flag is out |
| **Blue** | Blue | A single short burst |
| **White** (last lap / crossing start–finish) | White | Two quick pulses, played as the car crosses the line |
| **Red** | Red | A firm repeating buzz until the car slows or stops |
| **Black** | Black | A firm repeating buzz until the car slows or stops |

## 9.4 Not nagging a driver who has already responded

Red and black are the two flags that demand an immediate change in what the driver is doing, so their haptic repeats rather than firing once. Left unchecked that would buzz indefinitely. The watch stops the repeating haptic as soon as the phone reports the car below walking pace, or after twenty seconds, whichever comes first. The face keeps the flag's colour until the flag is genuinely cleared — the visual state is never shortened — but the wrist stops pulsing once the driver has clearly acknowledged it by slowing.

## 9.5 Scope, dependencies, and failure behaviour

The Watch integration belongs to the Advanced Mobile tier and rides entirely on the phone's existing event connection. It adds nothing to the Black Box path and issues no command to any vehicle system — like every flag on this platform, it is driver information only. A driver without a Watch loses nothing: every flag still arrives on the phone as banner and voice (Section 7). If the Watch link drops mid-session the failure is silent and self-correcting — the phone keeps writing application context, and the

moment the Watch reconnects it reads the current flag state and catches up, with no gap to notice and nothing for the driver to restart.

Crew - 10

# Race engineering

A pit wall for one car — live session visibility, and a direct line to the driver that a flag can't provide.

Race Engineering gives a crew a pit-wall view of their own driver's live session — the same telemetry stream described in Section 5, watched in real time from outside the car. On top of that visibility, a race engineer can send a short message directly to that driver's in-car display: not a track-wide condition like a flag, but something addressed to one car specifically — pit this lap, a gap to the car ahead, a fuel or tire call, anything a flag has no way to express. Spoken delivery of these messages is planned on the same on-device voice pipeline already used for flag alerts. It's a genuinely separate channel from race control's flag layer, and a separate product tier from the core platform.

Review - 11

# Driver analysis

Everything captured live in Section 5 becomes reviewable data here, the moment a session ends.

Every recorded session is listed, reviewable, renameable, and exportable. Session review overlays telemetry — position, speed, G-force, wing state, lap and sector splits — against the session timeline, so a driver or coach can step through exactly what the car and the driver did, corner by corner. Sessions export in a format compatible with common third-party motorsport analysis tools, carrying the same millisecond-accurate timing through to hundredths-of-a-second precision in the exported file.

Data Aggregation - 12

# Data aggregation

How the platform's own record of a driver's settings and real results grows lap after lap — and what that record makes possible.

## 12.1 What gets paired, session after session

Every time a session starts, the driver's active Custom Mode thresholds — the same G-force and speed-curve settings described in Section 4 — are captured alongside it. That snapshot sits next to the real telemetry the vehicle records during that same session: real G-force achieved, real wing position, real lap and sector times. Neither half is useful alone — the settings without the results don't say whether they worked; the results without the settings don't say what produced them. Recording both together, every session, is what turns driving into a growing record instead of a one-off.

## 12.2 Comparing settings to reality, one session at a time

Driver Analysis (Section 11) already surfaces this pairing directly: a session's configured thresholds shown next to what was actually recorded against them — how often a braking or cornering threshold was actually crossed, how the real G-force range compares to what was configured. Stated plainly, in the driver's own review, not buried in a database. Same principle as everywhere else in this platform: informational, never automatic — a driver reads the comparison and decides for themselves whether to adjust anything.

## 12.3 Where this leads: trend data across sessions

A single session's comparison is useful on its own, but the same paired record, kept every session, is what makes a longer view possible: how results have moved as settings changed, session after session, at the same car and the same track. That's the natural next step this architecture is built toward — not a guess bolted on after the fact, but the direct extension of data that's already being captured today. It's also the one place this platform is deliberately careful: lap time moves for plenty of reasons that have nothing to do with a threshold — tire wear, fuel load, traffic, weather — so a trend

across sessions is a data point worth looking at, never proof that one setting change caused a given result.

Onboarding - 13

# Event login

How a driver — and everyone else at the event — actually gets connected to it.

A driver joins a specific track event with an event code or an invite issued by that event's Track Management. Joining does the setup automatically: the venue's flag-zone layout and sector splits come down with it, and, where the driver has a peripheral connected, that geometry is pushed straight to their vehicle — nothing about the specific track has to be configured by hand for each event a driver attends.

Everyone else at the event — race control staff, corner marshals, a race engineer's crew — joins through the same web-based views described in Section 6, in a browser, with no install step at all. The mobile app is only necessary for a driver who wants their own vehicle's live telemetry and peripheral control tied into that event; it is never a requirement for the event itself to run.

Data - 14

# Data architecture

Every telemetry point is a timestamped, session-scoped record — stored with enough precision to tell two GPS fixes half a second apart, apart.

## 14.1 Telemetry storage

Live telemetry lands in an Oracle MySQL database, one record per data point, scoped to a session and a driver — position, speed, G-force, wing target/actual position, and the onboard lap/sector state at that instant. Sessions themselves are a separate record: owner, track, start time, and the totals (lap count, best lap) rolled up from the telemetry underneath.

## 14.2 Timestamp precision

Every stored data point carries a millisecond-resolution timestamp, computed at the moment of capture rather than inherited from a single value shared across a whole batch. That distinction matters more than it sounds: a batch of telemetry written in one database operation can otherwise share one identical whole-second timestamp if the time is left to a default, which is enough to make same-second GPS fixes indistinguishable and corrupt lap-time math built on top of them. The platform captures a timestamp at the moment each data point is built and stores it explicitly.

## 14.3 Export & interoperability

Sessions export in a widely used motorsport data format, carrying the same millisecond-accurate timing through to hundredths-of-a-second precision in the exported file — so a session reviewed in Driver Analysis and the same session opened in an outside analysis tool agree on when a lap actually started and ended.

REPRESENTATIVE DATA CAPTURED, PER DATA POINT

| DATA CAPTURED | SOURCE | NOTES |
|---|---|---|
| **Vehicle position** | Onboard GPS | Independent of phone location services |
| **Cornering & braking G-force** | Onboard accelerometer | Measured directly, not estimated from chassis angle |
| **Wing target vs. actual position** | Peripheral controller | Confirms the commanded position was actually reached |
| **Lap count & last/best lap time** | Onboard lap timer | Computed on the vehicle itself |
| **Sector splits** | Onboard lap timer | Same standalone logic as full laps |
| **Capture time** | Recorded per data point | Millisecond precision, set at the moment of capture |

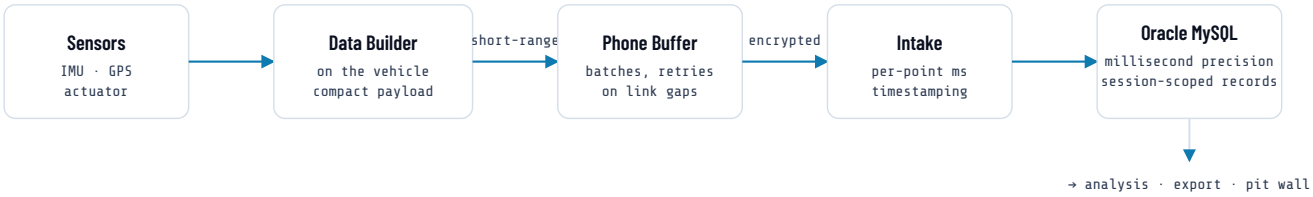



**Figure 4.** Every stage between the sensor and the stored record is either on-vehicle or explicitly buffered, so a stalled link delays delivery rather than losing data.

Methodology - 15

# Timing & measurement

How a lap becomes a number: the geofence model behind every crossing, on the vehicle and in the cloud alike.

## 15.1 GPS-based crossing detection

Both the vehicle's onboard timer and the cloud's session-review timer use the same two-radius geofence model around a target point — start/finish or a sector split. An outer radius arms the detector as the car approaches; an inner radius confirms the actual crossing. The gap between the two radii exists specifically to reject GPS jitter: a single noisy fix near the boundary can't fire a false crossing on its own, because it first has to arm from outside before it can trigger from inside.

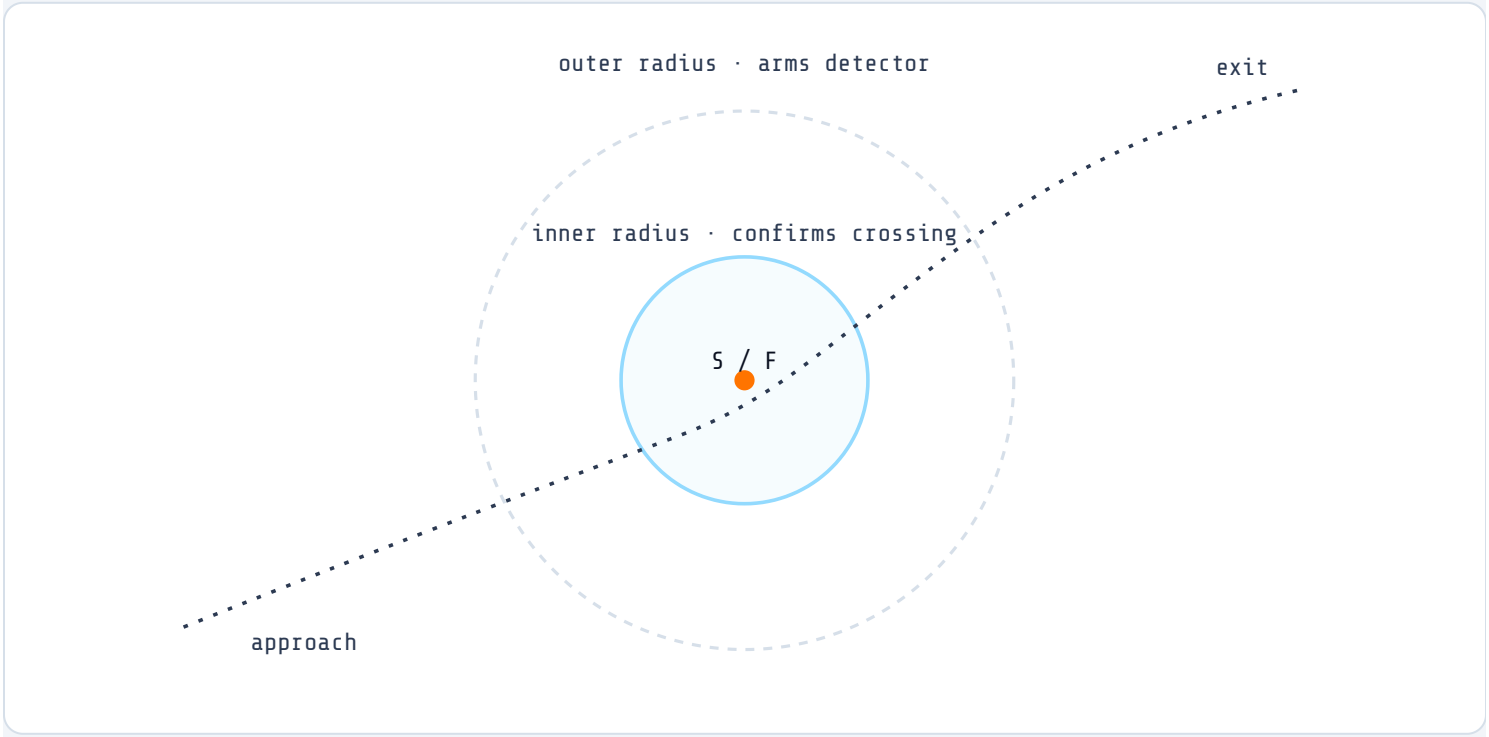



**Figure 5.** A crossing only registers once a fix arms from outside the outer radius and then confirms inside the inner one — plus a minimum time gate between crossings — so a single stray fix near the line can't split a lap in two.

## 15.2 Two timing paths, one model

The cloud's session-review timer runs this model against telemetry already stored from a phone-recorded session, gating repeat detections with a minimum sample count. The vehicle's onboard timer runs the identical model in real time against its own GPS fix, gating with a minimum elapsed time instead, since it can't assume a fixed sample rate the way the cloud-side calculation can. Both produce the same lap and sector splits; the onboard version is the one that works with no phone in range at all.

## 15.3 G-force: measured, not estimated

Cornering and braking G-force displayed to the driver and stored with the session comes from the vehicle's own accelerometer — the lateral axis read directly off the sensor — rather than being derived from chassis roll angle. It is a direct sensor reading, on the vehicle, at the moment it happened.

Configuration - 16

# Customization

Every layer of the platform is set up per driver, per venue, and per event — nothing is a fixed, one-size configuration.

## 16.1 Peripheral customization

Wing positions and G-force trigger thresholds are defined per driver, tuned to their own car and their own preference — not a shared factory setting.

## 16.2 Track & venue customization

Any track can be mapped once — GPS outline, start/finish, sector splits — and reused across every future session or event at that venue. A track owner or event operator sets this up a single time for everyone who uses their venue afterward.

## 16.3 Event customization

Track Management lets each event define its own flag-zone layout, staffing, and communication setup independent of any other event on the platform — a regional series running the same physical circuit on two different weekends can configure each one differently.

## 16.4 Communication customization

Spoken announcements can be turned on or off per driver. Pit-wall messages are written per situation by the crew sending them, not drawn from a fixed script.

## 16.5 Access customization

A driver, a race engineer, and a track manager or marshal each see a purpose-built view of the same event, scoped to what their role actually needs, rather than one generic screen shared by everyone.

Safety - 17

# Safety & reliability posture

What happens when a link drops or the unit is mounted wrong — stated plainly rather than assumed away.

## 17.1 Link loss

If the short-range link between phone and vehicle drops mid-session, the controller does not freeze or fail unsafe — it continues running whatever standalone behavior was already active (Track Mode, Custom Mode, onboard timing) using its own sensors, and simply stops receiving live relay signals until the link returns. Telemetry the phone can't currently deliver to the cloud is buffered and flushed once connectivity resumes, on both hops.

## 17.2 What can and can't be standalone — full list

FEATURE DEPENDENCY SUMMARY

| CAPABILITY | STANDALONE? | WHY |
|---|---|---|
| **Wing waypoint playback** | standalone | Fixed geometry, uploaded once, stored on the vehicle |
| **G-triggered actuation** | standalone | Driven by the controller's own live sensor reading |
| **Track-following (Track Mode)** | standalone | Stored track outline + on-vehicle GPS |
| **Lap / sector timing** | standalone | Stored target points, logged on the vehicle |
| **Mounting calibration** | standalone | A one-shot local operation against live sensor data |
| **Live cloud telemetry** | relay-dependent | Requires the phone's internet connection |
| **Pit-wall messaging** | relay-dependent | Originates from a race engineer off-vehicle |

Position - 18

# Where GBC fits

Neither a consumer lap-timing app nor a professional trackside system — a real option in the gap between them.

On one end of the market are simple, consumer-facing lap-timing and video-overlay apps: accessible and inexpensive, built entirely around a phone's own sensors, and offering nothing at the event or track-operations level — no flag control, no race control, no hardware-verified data.

On the other end are full professional trackside race-control and timing systems, built for series with dedicated technical staff and budgets well beyond a club program or a regional series' resources to operate or maintain.

GBC sits deliberately in between. It brings real, hardware-verified telemetry and an active on-vehicle peripheral together with genuine event-level tools — flag control, pit-wall communication, driver analysis — delivered as software a track or a series can adopt without the infrastructure a professional system assumes. That middle tier — club racing, regional series, track-day operators, and teams who have outgrown a consumer app but have no realistic path to a professional system — hasn't had a real option before. Opening it up is the specific opportunity here.

19

# Specification summary

| | | |
|---|---|---|
| **CONTROLLER**<br>Dual-chip design, short-range wireless only, on-board log storage, non-volatile configuration storage | **SENSING**<br>MPU6050 6-axis inertial measurement (3-axis accel + 3-axis gyro), NEO-6M GPS with course & fix-quality reporting | **ACTUATION**<br>Closed-loop linear actuator control, target vs. reported position |
| **ONBOARD TIMING**<br>Start/finish plus intermediate sector points, geofence crossing detection, logged on the vehicle | **WIRELESS LINK**<br>Short-range only, chunked & acknowledged transfers | **TIMESTAMP PRECISION**<br>Millisecond resolution, set per data point |
| **EXPORT FORMAT**<br>Widely used motorsport data format, hundredths-of-a-second precision | **CLOUD DATABASE**<br>Oracle MySQL | **PRODUCT SURFACES**<br>Driver Analysis · Race Engineering · Track Management |