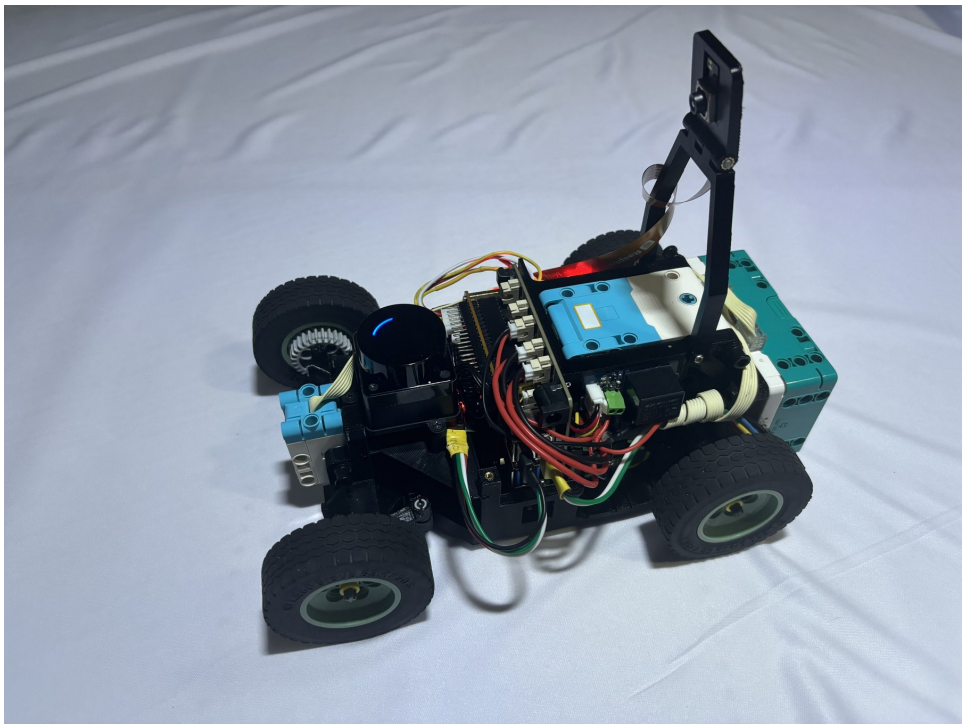
ENGINEERING JOURNAL

Team CSS

WRO Future Engineers 2026

Queretaro, Mexico



*The 2026 competition vehicle, **"The Maker"** (design v6)*

Vehicle	"The Maker"
Team members	Christian Gael Centeno Velez (Captain) · Sebastian Esquivel Mondragon
Coach	Manuel Alejandro Cardoso Duarte
Dimensions / Mass	240 × 160 × 250 mm · 1.006 kg
Drive / Steering	Rear-wheel drive, single axle · Front Ackermann, 45.44° max
Compute	Raspberry Pi 5 + Hailo-10H AI HAT+ · LEGO SPIKE Prime Hub
Repository	github.com/alex309-duarte/WRO_Future_Engineers_Queretaro_CSS

How to read this journal

This journal is organised around the five evaluation criteria in Appendix C of the 2026 Future Engineers rules, so that each criterion has an obvious home:

Rubric criterion	Where it is addressed
1. Mobility and Mechanical Design	Section 2 - steering, drivetrain, motor sizing, chassis, design evolution
2. Power and Sensor Architecture	Section 3 - power budget, failures, sensor selection, placement, calibration
3. Software Architecture and Obstacle Strategy	Section 4 - threading, perception, control, both challenge strategies, vision model
4. Systems Thinking and Engineering Decisions	Section 5 - tradeoff register, risk register, subsystem coupling
5. Reproducibility and GitHub Quality	Section 7 - repository map, interfaces, physical specification

Section 6 is the dated logbook that ties them together. **Every constant quoted in this journal is the value actually present in the source code in our public repository**, not a design target.

1. Design philosophy and season constraints

We did not start this season from a blank page. We finished the 2025 season with a working Future Engineers vehicle that placed 2nd nationally, and our starting decision was whether to iterate on it or rebuild. We chose to keep the *concept* - Ackermann-steered, LiDAR-led, LEGO SPIKE actuation - and rebuild everything else, because our post-season review identified three problems that were structural rather than tuning issues:

- **Latency.** The 2025 codebase was Python. Adding a higher-rate camera and LiDAR made the control loop the bottleneck, not the sensors.
- **Mass and packaging.** The 2025 vehicle came in *barely* under the 1.5 kg limit, with a high centre of gravity and loosely packaged electronics. Being that close to a hard rule limit is itself a design risk - it means any late addition is impossible.
- **Mechanical inconsistency.** The custom double-Ackermann steering we built in 2025 gave a tighter turning radius but did not behave the same way twice.

Those findings became the season's design goals: **cut latency, cut mass, and trade peak capability for repeatability wherever the two conflict.** That last principle explains most of the decisions in this journal, including several where we deliberately chose the *less* capable option.

Constraints we designed against

Constraint	Source	Limit	Our value	How it shaped the design
Maximum dimensions	Rule 11.1	300 × 200 × 300 mm	**240 × 160 × 250 mm**	Dense electronics packaging; ultrasonics removed
Maximum mass	Rule 11.2	1.5 kg	**1.006 kg** (67%)	Down from barely under 1.5 kg in 2025
One driving axle, one steering actuator	Rule 11.3	-	RWD + front Ackermann	Ruled out the 2025 AWD layout
No wireless during rounds	Rule 11.10	-	Radios disabled, Ethernet only	See Section 5.2
Track wall height	Game field	100 mm	LiDAR at 60-80 mm	**Hard constraint - see Section 3.4**
Round time limit	Rules 9.1 / 9.2	3 minutes	-	Sets minimum viable average speed for 3 laps
Parking lot length	Game field	1.5 × vehicle length	360 mm bay	Shorter vehicle means a proportionally shorter bay

Constraint	Source	Limit	Our value	How it shaped the design
Budget	Team	-	$33,018.73 MXN	Ruled out a second accelerator and a spare vehicle

The mass figure deserves emphasis. Going from *barely* under 1.5 kg to 1.006 kg did not just buy compliance margin - it is the root cause of half the changes in Section 2. A lighter vehicle needs less steering torque, which permitted a smaller steering motor, which returned mass and volume, which permitted denser packaging. We return to that chain in Section 5.3, including the part where it went wrong.

Note also the interaction in the parking row, which we did not anticipate at the start of the season: the parking bay is defined as 1.5× our *own* vehicle length, so making the robot shorter makes the bay proportionally shorter too. **Shrinking the chassis does not make parking easier.** It holds the difficulty constant while shrinking our margin for error in absolute millimetres.

2. Mobility and mechanical design

2.1 Steering: why we went backwards on purpose

In 2025 we designed and built a custom **double-Ackermann** linkage - both axles steering - to get a tighter turning radius. It worked, and on a good run it was better than a conventional layout. The problem was that it was not consistent: the additional linkage introduced play, and the steering did not return to the same neutral point reliably across runs.

For 2026 we replaced it with a **regular Ackermann** linkage steering the front axle only, reaching a maximum steer angle of **45.44°**.

> **We chose regular Ackermann instead of double-Ackermann because** the double-Ackermann's advantage was a smaller turning radius, but our failure mode was never "the corner was too tight to make" - it was "the robot did not end the corner pointing where we expected." The regular Ackermann completes every turn the WRO track requires with margin, and it removes a mechanical error source from every corner of a 12-corner run. We traded a capability we did not need for a consistency we did.

The Ackermann condition - inner and outer wheels tracing concentric arcs about a common centre - is what keeps the front tyres rolling instead of scrubbing. This matters twice over: scrubbing costs traction mid-corner, and it makes the effective wheelbase behave differently at different steer angles, which is exactly the non-repeatability we were trying to design out.

Actuator change. Steering is driven by a **LEGO SPIKE Small Angular Motor (45607)**, replacing the Medium Angular Motor used previously. This was a consequence of the mass reduction, not an independent choice: with a lighter chassis the required steering torque dropped enough that the smaller motor became sufficient, which then returned both mass and packaging volume.

2.2 Drivetrain: AWD to RWD

The 2025 vehicle was all-wheel drive. The 2026 vehicle is rear-wheel drive.

> **We chose RWD instead of AWD because** AWD required a front differential and driveshaft to carry torque past the steering axle. That hardware cost mass, volume in exactly the region where the Ackermann linkage lives, and one more set of mechanical tolerances. On a flat, high-grip competition mat the traction AWD buys is not the limiting factor - our lap times are limited by cornering confidence, not wheelspin. Paired with the LEGO 62.4 × 20 S rubber tyres, the single driven rear axle has not slipped in testing.

2.3 Motor sizing: the calculation we did before buying anything

We sized the traction motor against a dynamic torque budget rather than picking a motor and hoping. Target: reach cruising speed in **t = 0.5 s**, implying **a = 0.73 m/s^2**.

Parameter	Symbol	Value
Vehicle mass	m	1.006 kg
Wheel radius	r	0.0312 m
Target acceleration	a	0.73 m/s^2
Rolling resistance coefficient	C_{rr}	0.03
Gear reduction (20:28)	R	1.4
Gearbox efficiency	eta	0.85

Derivation.

– Acceleration force: $F_{acc} = m \cdot a = 0.7344$ N

- Rolling resistance: $F_{rr} = C_{rr} \cdot m \cdot g = 0.2961$ N
- Total: $F_{total} = 1.0305$ N
- Wheel torque: $T_w = F_{total} \cdot r = 0.03215$ N·m
- Motor torque: $T_m = T_w / (\text{eta} \cdot R) =$ **0.02702 N·m** (2.70 N·cm)
- With a 1.25× margin: $T_{m,req} =$ **0.03377 N·m** (3.38 N·cm)

Result against the LEGO Large Angular Motor (45602) - max-efficiency point 8.00 N·cm at 135 rpm, stall torque 25.00 N·cm:

Metric	Value	Interpretation
Load vs. max-efficiency torque	**33.8%**	Motor runs well inside its efficient band
Load vs. stall torque	**10.8%**	No realistic stall risk
Loaded top speed	**0.365 m/s**	Reached in 0.5 s at the design acceleration
Torque reserve	**66.2%**	Absorbs perturbations without dropping speed

Running at a third of the motor's efficient capacity is not conservatism for its own sake: a motor near its limit changes speed when the load changes, and a robot whose speed depends on how hard the last corner was is a robot whose distance-based manoeuvres drift.

2.4 Chassis and centre of gravity

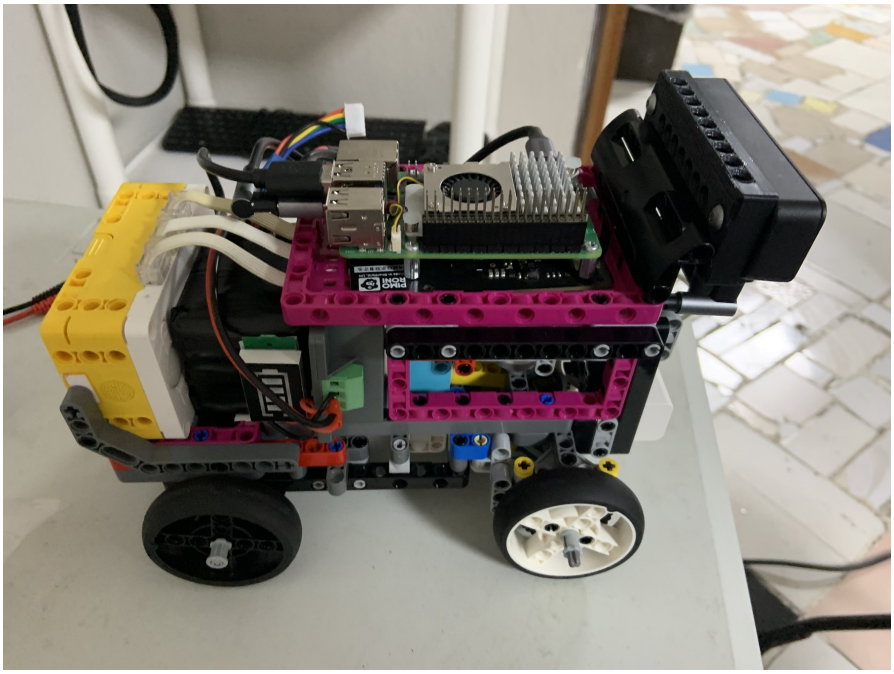
The chassis is a fully custom design, modelled in **SolidWorks** and printed on a **Creality K1C**. The 2026 revision repackages the electronics much more densely and moves the battery pack, the Raspberry Pi 5, the UPS board and the LiDAR **lower in the frame**.

The objective was lateral weight transfer. In a corner, transfer scales with CG height; less transfer keeps the inside front wheel loaded, which keeps the Ackermann geometry doing what it was designed to do instead of being defeated by an unloaded tyre. This was the mechanical half of the same "repeatability over peak capability" theme as Section 2.1.

2.5 Design evolution: six vehicles in two seasons

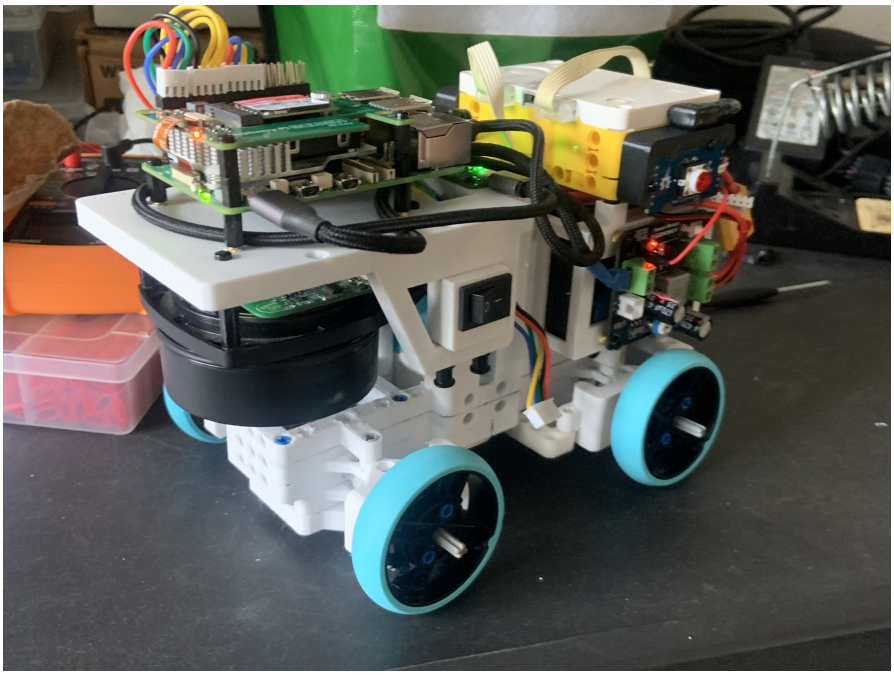
Our current chassis is the **sixth** distinct design the team has built across two seasons. Listing them matters because the 2026 vehicle is not an incremental descendant of the 2025 one - there is a deliberate discontinuity between v2 and v3.

2025 season - v1 and v2



***v1** - hybrid LEGO Technic chassis, regional competition (Queretaro). Yellow SPIKE Prime hub, Raspberry Pi on a Technic deck, OAK-D Lite camera at the front.*

Our first Future Engineers vehicle was built partly from LEGO Technic structural elements. The advantage was speed of iteration: we could change the geometry in an afternoon without waiting on a printer. The limitation is visible in the photograph - component placement is dictated by the Technic hole grid rather than by what the vehicle needs, the electronics sit high above the axles, and the assembly has play at every beam connection.

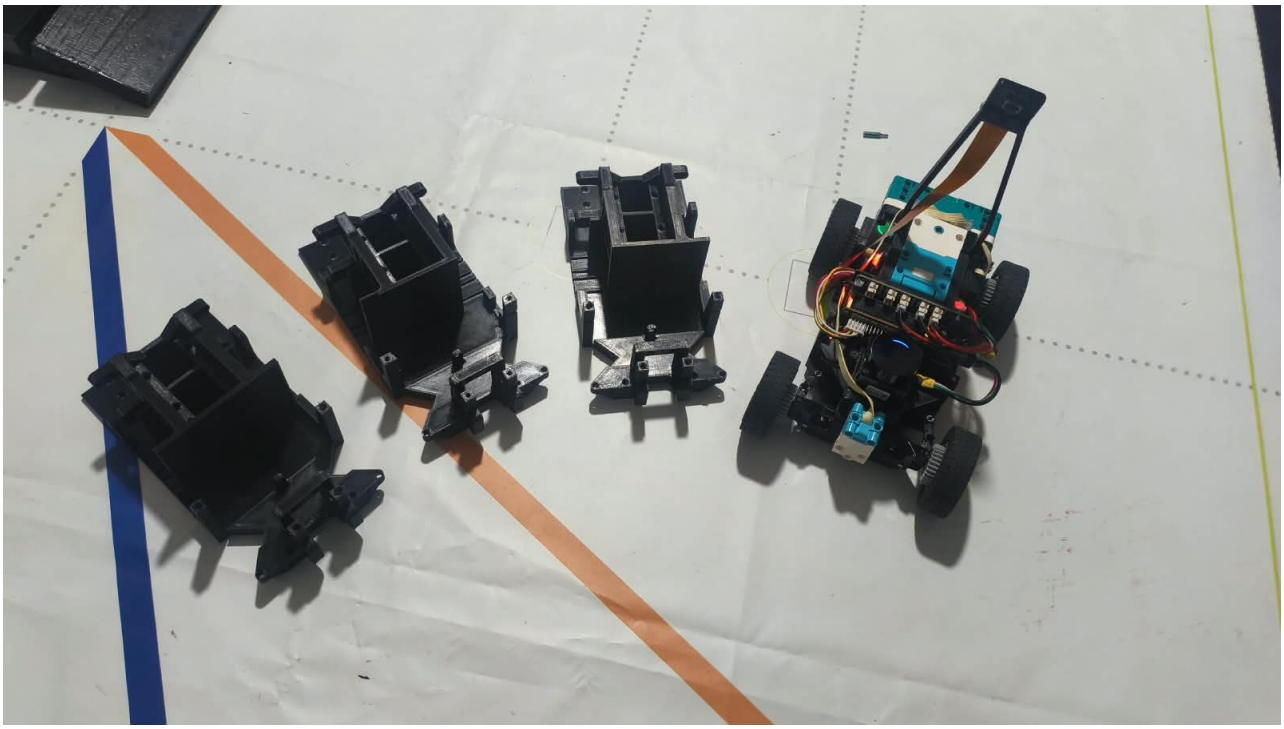


***v2** - fully custom 3D-printed chassis, national competition (2nd place nationally). RPLidar S2L on the front deck, double-Ackermann steering, AWD.*

For the national competition we abandoned the Technic structure and designed a fully custom, 3D-printed chassis. This removed the grid constraint and the accumulated joint play, and it is the design that carried the double-Ackermann steering and the AWD drivetrain to 2nd place nationally. Its weaknesses are the three findings listed in Section 1: heavy (barely under 1.5 kg), high centre of gravity, and mechanically inconsistent steering. Note in the photograph how much vertical stack there is above the wheels - that is the CG problem we set out to fix.

2026 season - a clean-sheet redesign, then four iterations (v3 to v6)

For 2026 we did **not** modify v2. We started a complete redesign, because the changes we wanted - lower CG, different steering geometry, RWD, denser packaging - were not reachable by editing the previous frame. That redesign then went through four iterations, each a focused improvement on the one before.



*The 2026 iterations side by side: **v3, v4 and v5** as bare printed frames, with **v6** assembled at the right. The progression from left to right is visible in the frame itself - material removed, mounting features consolidated, and the central bay opened up to drop the electronics lower.*

#	Iteration	Focus of this revision
v3	2026 clean sheet	New layout: RWD, regular Ackermann, battery and Pi dropped low in the frame
v4	Packaging	Component placement tightened around the new layout
v5	Mounting	Mounting and serviceability fixes, including the LiDAR mount revision of Section 3.4
v6	**Final**	Final geometry and mounting; the vehicle documented throughout this journal

> **We chose a clean-sheet redesign instead of iterating v2 because** the three problems we had were all *layout* problems. Lowering the CG means moving the battery and the Pi, which means moving everything around them; changing from double- to single-Ackermann changes the front axle entirely; dropping AWD removes the front driveshaft that the old frame was built around. Each change individually might have been retrofittable. Together they left almost nothing of the original frame, so retrofitting would have cost more than starting over and produced a worse result - a frame carrying mounting features for hardware that no longer existed.

The distinction between the two seasons is worth stating plainly, because it is the shape of our engineering process: **2025 to 2026 was a discontinuity driven by structural problems; within 2026, progress was incremental.** Once the layout was right, no iteration after v3 changed the concept - v4 through v6 are refinements of packaging, mounting and serviceability. Knowing which mode you are in matters: retrofitting a broken layout wastes time, and clean-sheeting a working one throws away tuning.

3. Power and sensor architecture

3.1 Power budget

Storage. 4 × Panasonic NCR18650B (3400 mAh, 3.6 V nominal) in a **2S2P** arrangement - two series stages of two parallel cells - giving **7.2 V nominal, 6800 mAh, 48.96 Wh**. The pack feeds a **Geekworm X1203** UPS/boost board that supplies the 5.1 V the Raspberry Pi 5 and its peripherals require.

This replaces the 2025 pack, which was the same 2-series topology with **three** cells per stage - **2S3P, six cells, 10200 mAh**. We reduced the pack by one third.

> **We chose 4 cells instead of 6 because** the 2025 pack was sized for a load we no longer have. Lighter, more efficient components (Section 3.5) cut the draw, and the endurance calculation below shows even the smaller pack runs for hours against a 3-minute round. Two cells of capacity we were never going to use cost mass we could not spare, on a vehicle that was already barely inside the 1.5 kg limit. This is the clearest case this season of removing capability because the requirement went away.

Component	State	Power
Raspberry Pi 5	High CPU / OS load	9.0 W
Hailo-10H AI HAT	Continuous neural network inference	4.0 W
Oradar MS200k LiDAR	Scanning, motor spinning	1.5 W
Grove modules	Relay pulses + LED button	0.1 W
Total at 5.1 V		**14.6 W**

Accounting for the X1203's ~90% conversion efficiency:

- P_{input} = 14.6 / 0.90 = **16.22 W**
- Pack current at 7.2 V = **2.25 A**
- Per parallel branch: **1.13 A per cell**

That per-cell figure is the one that matters. The NCR18650B is rated for 4.8 A continuous, so we are at roughly a quarter of it - which keeps the cells out of the thermal region where internal resistance climbs, and is precisely the failure we hit in Section 3.2.

Theoretical worst-case endurance: 48.96 Wh / 16.22 W = **3.0 hours**.

Measured reality. On the bench the integrated system draws 0.667 A per connector - each connector being one parallel pair - totalling 1.334 A at 5 V = 6.67 W, or 7.41 W at the pack. That gives **6.6 hours** of continuous running. The gap between 3.0 and 6.6 hours is simply that sustained simultaneous peak CPU *and* peak inference load does not occur in a real 3-minute round.

Known limitation. The cells charge in place through the X1203. Swapping in a charged set requires partially disassembling the chassis, so a depleted robot has to be plugged in rather than hot-swapped. Against a 6.6-hour endurance and a 3-minute round this is not a competition risk, but it does limit how much testing we can do away from mains power, and we would design a removable tray if we rebuilt this.

3.2 Failure: brownouts under acceleration

Symptom. With the Raspberry Pi 5 and the Hailo accelerator both active, the system would shut down abruptly during hard acceleration - not gradually, not with a warning, just off.

Investigation. The pack still showed plenty of charge, ruling out simple depletion. The correlation with acceleration pointed at a transient: the traction motor's current spike was pulling the rail down far enough to reset the Pi. We were using generic 18650 cells, and generic cells have high internal resistance - under a current spike the terminal voltage sags in proportion to that resistance.

Fix. Two changes together: **Panasonic NCR18650B cells**, which hold a much flatter discharge curve under load, and the **X1203 board**, which both regulates and buffers the rail. The brownouts stopped completely and have not recurred.

> **What we took from it.** A power system is not specified by capacity alone. Our original selection criterion was mAh, which is the wrong criterion for a load with sharp transients; internal resistance and discharge-curve flatness were what actually mattered. This is the clearest example this season of a component that met its stated spec and still failed the application.

3.3 LiDAR: RPLidar S2L to Oradar MS200k

Problem. During early testing the RPLidar S2L returned distorted and imprecise readings specifically when the vehicle passed very close to a track wall - exactly the condition our wall-following logic depends on most, and exactly when a bad reading is most expensive.

Decision. We replaced it with the **Orbbec Oradar MS200k**, whose optics and internal firmware handle ultra-short-range reflections much more cleanly. It is also lighter, serving the season's mass goal.

> **We chose the MS200k instead of keeping the S2L because** the S2L's noise was worst in the operating region we actually live in. A sensor that is accurate at 2 m and unreliable at 15 cm is the wrong sensor for a robot whose job is to hug a wall.

The driver switch is still visible in the codebase: `src/ondevice/` builds against either driver via a compile-time switch (`make LIDAR=oradar` selects the Oradar path; plain `make` still defaults to the legacy RPLidar driver).

3.4 Failure: the LiDAR could not see the walls

This is the most instructive failure of our season, because the mechanical design and the sensor requirement were developed separately and only met on the track.

Symptom. After finishing the new compact chassis, the robot behaved as though it were blind - it was not detecting the track walls at all.

Cause. The compact repackaging had placed the LiDAR high in the frame. The track walls are **100 mm** tall. The LiDAR's scanning plane was passing *over* them. Every wall reading was a miss, not a bad measurement - the mounting geometry had silently invalidated the entire perception stack.

Fix. We iterated the central CAD mount to bring the LiDAR down to roughly **60-80 mm** above the floor, safely inside the wall height. The non-obvious part was that lowering it put it level with the rear chassis structure, so the same revision had to keep the rear field of view clear - a 360° sensor with a structural blind spot behind it would have broken the rear-distance readings the parking manoeuvre depends on.

> **What we took from it.** Sensor placement is a constraint imposed by the *field*, not a packaging variable to be optimised freely. We now treat the 100 mm wall height as a hard design input alongside the 300 × 200 × 300 mm envelope, and any future chassis revision is checked against it before it is printed rather than after.

3.5 Vision: OAK-D Lite to Camera Module 3 Wide + Hailo-10H

The 2025 vehicle used an **OAK-D Lite**, an integrated stereo camera with onboard inference (visible on v1 and v2 in Section 2.5). For 2026 we split the job into a **Raspberry Pi Camera Module 3 Wide** for capture and a **Raspberry Pi AI HAT+ (Hailo-10H, 40 TOPS)** for inference.

> **We chose the split camera + accelerator instead of the integrated OAK-D Lite because** it was lighter, drew less power, and inferred faster - three of the season's goals at once. The cost is a more complex software stack: we now own the whole capture, preprocess, infer and postprocess pipeline rather than receiving finished detections from an integrated device. We judged that acceptable because owning the pipeline is also what let us train and deploy our own model (Section 4.8) instead of working within a vendor's toolchain.

3.6 What we removed: ultrasonic sensors

Previous seasons carried ultrasonic rangefinders. The 2026 vehicle has none. Once the MS200k proved reliable at short range (Section 3.3), the ultrasonics were measuring something the LiDAR already measured better, while costing mass, power, GPIO pins and mounting volume in a chassis we were actively trying to shrink. Removing a working sensor is uncomfortable, but a redundant sensor that nothing reads is just ballast.

3.7 Sensor placement and calibration

Item	Setting	Rationale
LiDAR height	60-80 mm above floor	Below the 100 mm wall height - see Section 3.4
ORADAR_ANGLE_OFFSET	270°	Empirically determined for this chassis's LiDAR mounting orientation; only needs re-checking if the mount rotates
Gyro zero	Automatic, at the start of every run	Rule 9.9 forbids pre-start calibration, so the reset happens *after* the start button is pressed
Steering neutral	45.44° max angle, fixed in hardware	No software calibration step
Chassis self-return rejection	Returns closer than 150 mm discarded	The robot's own body is inside the LiDAR's field of view

4. Software architecture and obstacle strategy

4.1 The migration to C/C++

Problem. The 2025 Python codebase became the bottleneck as soon as we attached a higher-rate camera and LiDAR. The sensors could supply data faster than the loop could consume it, so the robot was acting on stale information.

Decision. We rewrote the acquisition and control logic in **C/C++**, and spent the first weeks of the season doing nothing else. The justification was deterministic timing: when several high-rate sensor streams have to be combined at speed, garbage-collection pauses and interpreter overhead show up directly as steering jitter.

Result. The main control loop stabilised at **more than 50 Hz**.

> **We chose to rewrite instead of optimising the Python because** the problem was not a slow function we could profile and fix - it was the cost of the runtime itself under multi-threaded, high-rate sensor load. Optimising would have bought a constant factor; changing language changed the ceiling. The cost was several weeks of the season producing no new capability, which is why it had to happen first or not at all.

4.2 Threading model

The Obstacle Challenge binary (`object_detection`) runs cooperating threads:

- a **Hailo pipeline** thread - preprocess, inference and postprocess on camera frames, keeping a global `traffic_lights` struct (colour, x-position, area, confidence) current;
- a **LiDAR reader** thread - continuously filling a 360-slot distance buffer;
- `Obstacle_Challenge_Thread` - the navigation state machine, which reads both and issues motor commands over serial to the SPIKE hub.

The Open Challenge is a **separate, simpler binary** (`src/ondevice/`) that never starts the camera or Hailo pipeline at all - there are no coloured pillars to detect, so loading the vision stack would only add boot time and failure surface.

4.3 Wall detection: Select_Wall()

Each call runs a six-stage pipeline. The constants below are the values in the source.

#	Stage	What it does
1	Angular windowing	A 70°-wide window centred on the requested side, with the innermost ±3° excluded to avoid seam and corner noise
2	Chassis filtering	Returns closer than **150 mm** are the robot's own body
3	Radius Outlier Removal	A point is dropped with fewer than **3** neighbours within **300 mm**, with 15° of padding outside the window so edge points are not unfairly penalised
4	Segmentation	Split wherever consecutive points jump by more than **60 mm**, then split again at corners by Iterative End-Point Fit with a **40 mm** deviation threshold
5	Scoring	A segment qualifies only with at least **6 points**, **200 mm** end-to-end extent, and an RMS perpendicular residual under **15 mm** against its Total-Least-Squares best-fit line
6	Per-side tracking	Once a wall is chosen for a side, later frames prefer to keep following that same physical wall (matched within **300 mm** and **20°**) rather than re-snapping every frame

Total-Least-Squares (PCA) is used rather than ordinary least squares because it has no bias toward any particular wall orientation. Stage 5 earns its complexity: the 200 mm minimum extent is what rejects the 50 × 50 mm traffic-sign pillars. Without it, a pillar at the right distance reads as a wall and the robot follows an obstacle.

4.4 Heading estimation - and what it is not

Heading comes primarily from the **SPIKE Prime Hub's built-in gyroscope**, read and reset over serial. LiDAR wall-orientation readings (`Slope()`, `Correction_For_Triangles_Left/Right`) correct position and alignment at defined points in the manoeuvre logic.

We want to be precise, because it is easy to overclaim: **this is not sensor fusion.** There is no complementary filter, no Kalman filter, no algorithm combining the two signals into a single estimate. The honest description is **gyro-primary heading with situational LiDAR-based positional correction** - the gyro runs orientation continuously, and the LiDAR intervenes at known points to correct accumulated drift against a known wall.

We considered a complementary filter and did not implement one. The gyro's drift over a single 3-minute round is small, and re-squaring against a physical wall at every corner resets it to a ground truth that a filter would only have estimated. A filter would have added tuning parameters and a new failure mode to solve a problem the geometry already solved.

4.5 Closed-loop steering

Steering uses **PD** control - proportional to lateral position error, derivative to damp the oscillation pure P produces. Two separately tuned loops:

Controller	K_p	K_d
Camera-based cube following	150	100
Gyro-based wall following	2	300000

Both gain sets were obtained by **trial-and-error tuning on the physical track**, not derived analytically. We state that plainly rather than implying a model-based process. The very different magnitudes reflect the different error units the two loops work in - normalised pixel offset versus degrees of heading error.

Our tuning method was to change one gain at a time and judge the result by watching the robot's line through a section: too little D and the robot weaves down a straight, too much and it responds sluggishly to a real correction. This is a subjective process, and it is the main methodological weakness of our software work this season - see Section 8.

No integral term is used in either loop. Steady-state offset is not our failure mode, and an integrator accumulating during a cube-avoidance manoeuvre would fight the next correction.

4.6 Open Challenge strategy

The run is direction-agnostic: an initial LiDAR read determines which side is open (`sentido`), and the entire subsequent path is the sign-mirror of the other direction.

Constant	Value	Where it acts
Outside/inside start split	800 mm	A side reading above this means an "outside" starting box
First straight (outside / inside)	1350 / 300 mm	Distance driven after the first turn
Slope sampling gap	300 + 700 mm	Two Slope() reads, averaged into one heading correction
Corner turn angle	80°	Every corner turn
Corner loop	10 iterations	Plus the setup turn and the final turn = 12 corners = 3 laps
Final approach bands	400 / 600 mm	Selects the parking stop distance
Final front threshold	1500 mm	Chooses a 1400 or 1750 mm stop from the front wall

Averaging *two* slope measurements taken 700 mm apart, rather than trusting one, is a deliberate noise-rejection step: a single bad wall fit at the start of a straight would otherwise be baked into the gyro reference for that whole section.

4.7 Obstacle Challenge strategy

Direction is decided once from the initial side-wall reading (600 mm split) and held for the attempt. The unit of the run is one **corner cycle** - `Corner_Case` followed by `Desicion` - repeated **12 times** (3 laps × 4 corners), with the 4th corner of each lap run in parking mode.

`Corner_Case` advances to 1100 mm from the front wall, reads the pillar colour the camera currently sees, and advances a colour-dependent distance (green 600/400 mm, red 990 mm, none 650 mm) before turning 86° into the next section. `Desicion` then re-squares the gyro against the outer side wall and dispatches to the cube-avoidance routines, which compute an angle and hypotenuse to the detected pillar, turn toward it, pass it, and straighten.

Passing rule: green means pass on the left, red means pass on the right. This is fixed by rule 9.19 and does **not** mirror with the driving direction - worth stating explicitly because almost everything else in the run does mirror.

Parking. The vision model detects the magenta parking-lot markers directly, as a dedicated `x-parking` class alongside the two pillar classes. Detecting the markers rather than inferring the bay from wall geometry was a deliberate choice: the bay is defined relative to a dotted line and a movable magenta element, so its position is not derivable from the track walls alone. Once the markers are located, the LiDAR positions the robot at a set distance from the wall and the manoeuvre runs on combined LiDAR and gyro readings.

4.8 Vision model: training and measured performance

We trained our own detector rather than using a stock model, because the classes we care about - red and green traffic-sign pillars, and the magenta parking markers, under competition lighting - are not COCO classes.

Dataset. Hand-annotated by the team in YOLO format, held in `other/dataset/`. The pillar classes were collected first and deliberately balanced almost exactly; the `x-parking` class was added in a later collection pass once the parking strategy of Section 4.7 was settled.

Training configuration. YOLOv8n, 100 epochs, batch 16, image size 320, initial learning rate 0.01, on GPU. Augmentation: HSV jitter, horizontal flip (p = 0.5), mosaic with close_mosaic at 10, random erasing 0.4. Horizontal flip is safe here - a pillar is symmetric - and note it does *not* teach the model anything about which side to pass, since that is decided in code from the class, not from the image.

Measured results from the training run published in `other/Training_model_and_results/results.csv` (501 images, two pillar classes, 1,896 labelled instances balanced 947 / 949):

Metric	Epoch 1	Epoch 100
Precision	0.012	**0.993**
Recall	0.968	**0.984**
mAP@50	0.462	**0.986**
mAP@50-95	0.316	**0.902**

Precision at 0.993 with recall at 0.984 is the balance we wanted. A false positive sends the robot around a pillar that is not there; a brief miss is recoverable, because the LiDAR still prevents a collision with a real obstacle. **We would rather miss a frame than hallucinate one.**

> **Note for judges:** the metrics above are from the two-class training run whose full artefacts (curves, confusion matrix, weights) are published in the repository. The model deployed on the competition vehicle adds the `x-parking` class; its training artefacts are published alongside the updated weights.

Deployment. The `.pt` weights are compiled to Hailo `.hef` format with the Hailo Dataflow Compiler. The compiler and the on-device HailoRT runtime must both be **v5.x** for the Hailo-10H - the Hailo-8/8L v3.x/v4.x pairing is incompatible, and a version mismatch produces a `.hef` that compiles cleanly and then fails to load on the Pi. This cost us time and is written down here so it does not cost anyone else's.

5. Systems thinking: tradeoffs, risks and mitigations

5.1 Tradeoff register

#	Decision	Rejected alternative	Why	Cost accepted
1	Regular Ackermann	Double-Ackermann (v2)	Repeatability beat turning radius	Wider turning circle
2	Rear-wheel drive	All-wheel drive (v2)	Traction was never the limit; AWD cost mass and volume	Less grip in a hypothetical low-grip case
3	Clean-sheet 2026 frame	Retrofitting the v2 frame	All three problems were layout problems	Restarted the mechanical design
4	C/C++	Optimising Python	Needed a higher ceiling, not a constant factor	Weeks of season time, no new features
5	Oradar MS200k	RPLidar S2L	Accuracy where we operate: very close to walls	New driver integration
6	Camera + Hailo-10H	OAK-D Lite	Lighter, lower power, faster inference	We own the whole vision pipeline now
7	4-cell 2S2P pack	6-cell 2S3P (2025)	Load dropped; endurance still far exceeds round length	Less reserve if consumption grows
8	Small steering motor	Medium (2025)	Lower mass made it sufficient	Less torque headroom if mass grows
9	No ultrasonics	Keep as redundancy	LiDAR measured it better; mass, pins, volume	No independent backup range sensor
10	Gyro + situational correction	Kalman / complementary filter	Wall re-squaring gives ground truth, not an estimate	No continuous fused estimate
11	No integral term	Full PID	Steady-state offset is not our failure mode	Small persistent offsets uncorrected
12	Vision-detected parking markers	Geometric inference from walls	Bay position is not derivable from the walls alone	Depends on the detector seeing magenta reliably
13	Single vehicle, no spare	Build two	Budget; rule 11.26 forbids a spare on the day anyway	No fallback if it breaks in transit

5.2 Risk and failure-mode register

Risk	How it shows up	Mitigation	Status
Gyro drift over a run	Robot ends a corner mis-aimed	Re-square against the side wall at every corner (Slope, ±90°)	Mitigated in code
Pillar misread as wall	Robot follows an obstacle	200 mm minimum wall extent in Select_Wall()	Mitigated in code
LiDAR noise at very short range	Bad wall fit near a wall	Sensor change (3.3) + ROR filtering	Mitigated
Voltage sag under load	Sudden reset mid-run	Low-ESR cells + X1203 buffering (3.2)	Resolved
LiDAR blind to walls	Total navigation failure	Mount height fixed below 100 mm (3.4)	Resolved
.hef / runtime version mismatch	Model fails to load at the venue	Both pinned to v5.x; documented in the README setup steps	Documented
Wireless used during a round	Disqualification (rules 11.6 / 11.10)	See procedure below	Procedure defined

Risk	How it shows up	Mitigation	Status
Sensor failure mid-run	Robot continues on bad data	**None implemented.** No automatic fallback exists	Open, accepted
Emergency stop	-	**No hardware e-stop** separate from terminating the process	Open, accepted

Wireless compliance procedure. During development we work over Wi-Fi SSH, because it is far more convenient for iterating on the track. That is not permissible during a round, so at competition we switch to **SSH over a wired Ethernet connection**, which the team has done at previous competitions. Our documented pre-round procedure is:

- Connect to the vehicle over Ethernet for any final setup between rounds.
- Disable the Raspberry Pi's Wi-Fi and Bluetooth radios in software, and confirm the SPIKE Prime hub's Bluetooth is off.
- **Physically disconnect the Ethernet cable before the round begins** - rule 11.6 prohibits wired control systems while the vehicle is running, not merely wireless ones.
- Start the vehicle with the single power switch, then the single start button (rules 9.10, 9.11).

Steps 2 and 3 are the ones that are easy to forget under time pressure, which is exactly why they are written down here rather than left as team knowledge.

We list the final two rows of the table as unresolved rather than omitting them. Neither is required by the rules, and adding a second button would conflict with rule 9.11's single-start-button requirement, but a judge asking "what happens if the LiDAR dies on lap two?" deserves the honest answer: the robot keeps driving on stale data until the round ends.

5.3 How the subsystems constrain each other

The decisions above are not independent, and the season's most useful lesson was watching one change propagate:

> **Reduce mass** -> steering torque drops -> **smaller steering motor fits** -> mass and volume freed -> **denser packaging possible** -> CG lowered -> less weight transfer -> **Ackermann geometry stays effective** -> cornering more repeatable.
>
> In parallel: lighter and more efficient electronics -> **lower current draw** -> the 6-cell pack became two cells of unusable reserve -> **4-cell pack** -> less mass again, feeding back into the same loop.
>
> And simultaneously, the same denser packaging -> **LiDAR pushed upward** -> *above the 100 mm walls* -> **total perception failure** (Section 3.4).

The design move that produced most of our gains also produced our worst failure. That is the clearest argument we can make for why a chassis revision is now checked against the field geometry before it is printed.

6. Logbook

Date	Milestone	Section
2026-02-07	Repository created; season begins. Decision to rebuild rather than iterate the 2025 vehicle	1
2026-03-07 to 03-14	SPIKE and Raspberry Pi control functions ported to C; repository restructured to the WRO template layout	4.1
2026-03-21	KiCad schematics added; RPLidar S2L driver integrated	3.3
2026-04-18	Double-Ackermann reviewed against 2025 race experience; decision taken to revert to regular Ackermann for 2026	2.1
2026-05-09	**v3** - clean-sheet chassis: RWD, regular Ackermann, battery and Pi dropped low in the frame	2.5
2026-05-30	**v4** - packaging refinement around the new layout	2.5
2026-06-13	Brownouts under acceleration diagnosed; generic cells replaced with Panasonic NCR18650B, X1203 integrated	3.2
2026-06-27	Battery pack reduced from 2S3P (6 cells) to 2S2P (4 cells) after re-running the power budget	3.1
2026-07-11	RPLidar S2L short-range noise identified; Oradar MS200k selected and driver integrated behind a compile-time switch	3.3
2026-07-25	**LiDAR mounting-height failure found** - scan plane passing above the 100 mm walls	3.4
2026-08-01	**v5** - revised central LiDAR mount, sensor lowered to 60-80 mm, rear field of view cleared	2.5, 3.4
2026-08-08	OAK-D Lite replaced by Camera Module 3 Wide + Hailo-10H AI HAT	3.5
2026-08-15	Vision dataset labelled; YOLOv8n trained, 100 epochs, mAP@50 0.986	4.8
2026-08-22	Model compiled to .hef and deployed; HailoRT / DFC v5.x version pairing resolved	4.8
2026-08-29	**v6 (final)** - final geometry and mounting; vehicle photographed	2.5
2026-09-05	Ultrasonic sensors removed after the MS200k proved reliable at short range	3.6
2026-09-07	Documentation, BOM, flowcharts and final repository organisation completed	7

7. Reproducibility

Everything needed to rebuild this vehicle is in the public repository at `github.com/alex309-duarte/WRO_Future_Engineers_Queretaro_CSS`:

Content	Location
Control and vision source	src/
CAD - SolidWorks parts, assemblies and STEP	models/cad/New/
Printable STLs and print settings	models/cad/New/stl/
Design history photographs (v1 to v6)	models/designs/
Wiring diagram	schemes/hardware/wiringSchematics.jpeg
Training dataset and labels	other/dataset/
Training results and weights	other/Training_model_and_results/
Datasheets	other/datasheets/
Compiled models	src/HailoModels/
Build and run instructions	README.md section 8 and src/README.md

Hardware interfaces

Interface	Port	Rate
SPIKE Prime hub	/dev/ttyACM0	115200 baud
Oradar MS200k LiDAR	/dev/ttyUSB0	230400 baud
Start button	GPIO 4 (BCM)	-
Status LED	GPIO 3 (BCM)	-
SPIKE power relay	GPIO 2 (BCM)	-

Physical specification

Property	Value	Rule limit
Dimensions (L × W × H)	240 × 160 × 250 mm	300 × 200 × 300 mm
Mass	1.006 kg	1.5 kg
Drive	Rear-wheel, single axle	One driving axle
Steering	Front Ackermann, 45.44° max	One steering actuator

Bill of materials: 12 line items, **$33,018.73 MXN** total. Full table with sourcing links in `README.md` section 6.

8. What we would do differently

– **Check sensor geometry against the field before printing a chassis.** Section 3.4 cost us real time and was entirely predictable from a number printed in the rules.

– **Specify power by internal resistance, not capacity.** Section 3.2 was a selection-criterion error, not a component failure.

– **Design a removable battery tray.** The in-place charging constraint limits how much testing we can do in a session away from mains power.

– **Instrument the robot to log runs.** This is our biggest methodological gap. Our PD tuning (Section 4.5) and our mechanical iterations (Section 2.5) were judged by watching the robot, not by measurement. Logged per-run telemetry would let us say "this change improved corner exit angle by X degrees" instead of "it felt

better", and would have made this journal considerably stronger. It is the first thing we would build next season.

Team CSS - Christian Gael Centeno Velez, Sebastian Esquivel Mondragon
Coach: Manuel Alejandro Cardoso Duarte - Queretaro, Mexico - 2026 season