PRODUCT DOCUMENTATION

FieldScout

Operating manual & technical reference

MODULAR MOBILITY FOR AGRICULTURAL MONITORING

Know the platform. Operate with confidence.

Operate

Configuration-specific controls, startup and stopping

Integrate

ESP32 interface, wheel mapping and Modbus messages

Maintain

Inspection, fault diagnosis and delivery records

FS-UM-001 / Revision 1.0 / 21 September 2026

Reference configuration edition. Concept image shown; delivered equipment governs.

READ THIS FIRST

The manual follows the supplied configuration

This manual explains the FieldScout mobile platform, documented control interfaces and the FieldScout Navigation Studio. Keep it with the robot and make it available to every operator. Use it together with the signed delivery record, the installed controller's release notes and the component instructions supplied with the unit.

One platform, three distinct control contexts

Standalone demonstration: the supplied demonstration firmware executes a repeating motion sequence and accepts a small set of serial commands.
External controller: the supplied CoppeliaSim script describes a serial wheel-command interface. Its matching ESP32 receiver must be supplied and verified separately.
Navigation Studio: a browser simulation for learning, demonstrations and message inspection. It does not control the physical robot.

Who should read which sections?

Reader	Recommended route
Operator	Read the product, installation and operation sections; use the quick-start sheet only after training.
Qualified integrator	Also read the interface and handover sections before changing sensors, wiring, firmware or command software.
Service technician	Read care and troubleshooting; identify the exact hardware revision and isolate power before work.
Purchaser or site owner	Confirm supplied options, environment, approved limits, support arrangements and the delivery record.

How to interpret the information

Documented behavior is traceable to the supplied software or project material. **Reported specification** is a brochure value, not a new test result. **Configuration-dependent** means the delivered unit and its acceptance record determine the answer. Source tags such as [S4] point to the source register at the end.

Before first powered use

Complete the delivered-unit record in Section 7. Confirm the installed firmware and its power-on behavior, the physical stop arrangement, battery/charger identity and approved operating limits. If these cannot be established, keep the drive power isolated and contact Adaptive AgroTech. The demonstration firmware can start motion automatically after an eight-second countdown.

FS-UM-001 / Rev. 1.0 / 21 September 2026. Related documents: FS-DS-001 (datasheet) and FS-QS-001 (quick start). The quotation, order and supplied service/warranty agreement govern commercial terms.

Contents

1 Product overview

FieldScout is a modular four-wheel mobile platform developed by Adaptive AgroTech for agricultural monitoring and related research. Its intended applications include moving between observation points, carrying environmental or imaging instruments, and recording measurements with position and time. Greenhouses, livestock facilities and open-field research are application contexts; suitability for a particular site depends on the delivered configuration and the conditions in which it has been validated. [S1][S2][S3]

The existing drive platform uses an ESP32 controller and motor drivers to operate the wheels. A navigation computer, LiDAR and supporting sensors can form the higher-level navigation system. The navigation configuration retains this existing motor-control layer. Adding a computer or sensor does not, by itself, establish autonomous operation: the software interface, localisation, obstacle response, power supply and stopping behaviour must work together and be checked on the actual unit. [S2][S3][S4]

Read this manual with the unit's configuration record

Record the delivered model and unit identifier, installed equipment, released software and permitted operating conditions at handover. Keep brochure references, simulator parameters and verified unit data clearly distinguished. Complete any missing configuration or test information before relying on the affected function.

1.1 How the main subsystems work together

The higher-level computer handles sensing, localisation and mission decisions in a navigation-equipped configuration. It issues bounded motion requests to the ESP32. The ESP32 is the local motor gateway. Its approved navigation receiver must validate the agreed request format, manage the required local control behaviour and translate wheel targets into motor-driver commands. The existing controller example uses one RS485 bus with two addressed dual-channel drivers; the ESP32 is its only Modbus master. A new host must not independently compete for control of that bus. [S3][S4]

This division allows navigation software to be developed without replacing the established wheel-drive hardware. However, the reviewed standalone demonstration and serial-heartbeat sketches are not complete receivers for the four-wheel command stream expected by the CoppeliaSim example. Confirm the actual operating firmware and its compatible external-command receiver at handover before connecting a navigation host. [S4][S5][S6]

1.2 Capabilities and configuration status

Use the following table to complete the delivered-unit configuration record. For each “Confirm” entry, record the installed equipment, applicable documentation and demonstration result.

Function or subsystem	Configuration basis	Confirm for the delivered unit
Four-wheel drive	Existing ESP32 and motor drivers; the controller example records two driver addresses and a four-wheel polarity map.	Motor/driver models and revisions, wheel identity, wiring, operating limits and the approved control profile.
Navigation computer	A Jetson-based higher-level system is described. Project material names Orin Nano.	Exact board and carrier: original Jetson Nano and Jetson Orin Nano are different platforms. Record the validated software image.
LiDAR navigation	Target integration includes ranging, localisation, route planning and obstacle handling.	Sensor model, mounting, software release, measured footprint, feedback source and demonstrated operating area.
Position and payload sensors	IMU, GNSS, camera and environmental sensing are described as possible parts of the platform.	Which instruments are fitted, their interfaces, calibration, acquisition software and data-quality limits.
Operator interface	The web package provides an interactive 3D view, a driving simulation and command inspection.	Whether a separate physical operator interface and telemetry connection have actually been commissioned.
Power and stopping	Drive and computing/sensor loads require a compatible protected supply and a verified stop arrangement.	Battery, charger, BMS, supply rails, protection, physical stop procedure and restart behaviour.

[S2][S3][S4][S6][S7]

1.3 Brochure-reported reference specifications

These values reproduce the product brochure's reported reference figures. Use the delivered-unit rating information, as-built measurements and qualification evidence to establish the operating limits of the actual robot. Confirm that any enclosure classification covers the complete fitted assembly. [S1]

Parameter	Brochure reference	Interpretation for use
Overall dimensions	650 × 480 × 420 mm	Reported length, width and height. Measure the configured robot, including added sensors and projections.
Mass	Approximately 22 kg	Reported mass. Weigh the unit with its working battery, accessories and payload.
Maximum speed	1.5 m/s	Reported maximum, not an approved commissioning speed or a guaranteed speed in every environment.
Operating time	Up to 8 h	Reported endurance; terrain, payload and duty cycle matter. No matching measured mission-energy record is supplied with the reference.
Drive arrangement	Four-wheel drive	Identify the installed drive components and verify the physical wheel mapping.
Enclosure protection	IP54	Brochure-reported classification. Supporting evidence for the delivered complete assembly must be confirmed before exposure decisions.
Operating temperature	-10 to $+50\,^{\circ}\mathrm{C}$	Reported range. Confirm component and complete-system limitations for the installed configuration.

Different reference configurations

The Lua reference uses a nominal total mass of 26 kg, a body box of approximately 397.55 × 240 × 189.34 mm, wheel radius 0.0535 m, track 0.250 m and wheelbase 0.200 m. Its header also declares 24 V, 100 W motor information. These are source/model declarations: the body box is not the full robot envelope, the nominal mass is not a weighing result, and the motor voltage does not establish the installed battery voltage. Keep these model parameters separate from the brochure figures and from the verified delivered-unit specification. [S6]

1.4 What the browser package demonstrates

The V9 browser environment supports manual simulated driving, waypoint missions and a 3D representation of the robot. It also demonstrates a route changing when synthetic LiDAR detects an added obstacle: an A* planner uses the updated simulation map to find a detour, or reports a blocked route when none is available. The accompanying display exposes logical wheel commands and illustrative Modbus messages. [S7]

The model uses an ideal pose and exact geometry for a detected simulated obstacle. It does not implement a physical SLAM system, supply measured motor feedback, confirm a live driver acknowledgement or establish a robot connection. Use the browser environment to understand operation and interfaces, and the physical test record to establish the capabilities of the delivered navigation system. [S7]

2 Installation and operating boundaries

2.1 Prepare the site and the operator

Use a trained operator who can maintain sight of the robot, keep the route clear and operate the physical stop. Establish a controlled test area before the first drive. Keep hands, feet, clothing and cables away from wheels and pinch points. Do not carry people or use the platform as a step.

The first commissioned operating area should be level, dry and free of traffic, loose cables, edges, holes and overhead restrictions. Greenhouses, barns and outdoor fields introduce different surface, visibility and hygiene conditions. Approval for one site does not establish approval for all three.

Condition	Check before operation	Action if uncertain
People and animals	A controlled route and an exclusion area sized to the verified stopping behavior	Stop; do not rely on a browser model or a single LiDAR plane for protection.
Surface and slopes	Load, grip, cross-slope, edges and step clearance within the delivered limits	Select another route. Do not infer climbing ability from the wheel appearance.
Water and cleaning	Actual enclosure and connector protection, seals and permitted cleaning method	Keep dry; do not assume the reported IP54 claim authorizes rain or washdown.
Temperature and ventilation	Battery, motor-driver and compute limits; unobstructed cooling	Stop on alarms, abnormal heating or blocked ventilation.
Payload and attachments	Approved mass, mounting points, clearance and center of gravity	Obtain a mounting/loading specification before installation.

2.2 Receive and identify the equipment

1. Compare the supplied robot, battery, charger, cables, keys/tools, sensors and documentation with the order and packing list. Record missing or damaged items before use.
2. Record model/SKU, serial number, hardware revision and installed firmware build. Photographs in this manual show reference versions and are not a connector-location guarantee.
3. Check for loose wheels, cracked parts, exposed conductors, shipping damage, damaged battery casing and insecure payloads. Keep a damaged unit out of service.
4. Identify the main power control, physical stop, charging point and intended computer/USB connection from the unit-specific labels and wiring sheet. The source material does not establish their positions for every unit.
5. Have the supplier demonstrate normal stop, emergency stop and restart behavior on the delivered configuration. Retain the test record.

2.3 Power and signal architecture

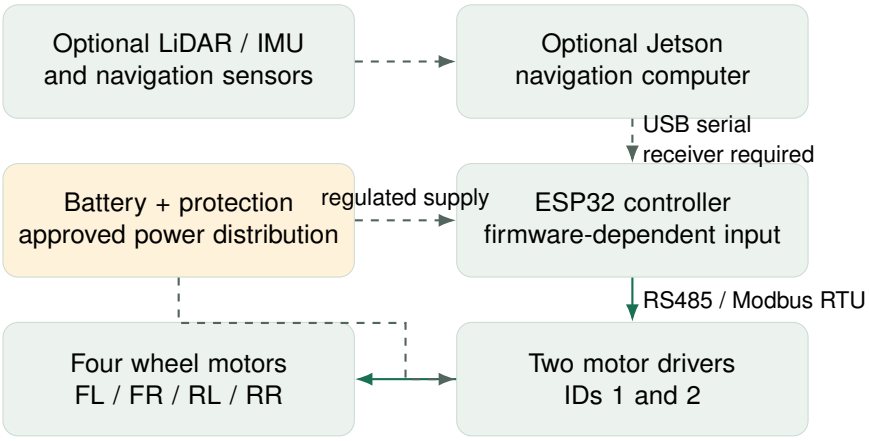

Figure 1. Functional architecture. Solid green arrows show the documented drive-command path. Dashed links require confirmation against the delivered unit. Power connectors, protection ratings and stop-circuit implementation belong on its electrical drawing.

The ESP32 should remain the sole Modbus master on the existing drive bus. A navigation computer normally sends high-level or wheel commands to a verified ESP32 receiver. Direct driver access is an integration alternative requiring controlled bus ownership; a second active master must not be added in parallel.

Power isolation is separate from a serial command

A `STOP`, zero-speed frame or software `ESTOP` is a communication-dependent request. It is not evidence that motor power is physically isolated. Before handling wheels, opening covers, moving wiring or servicing the unit, stop the controller, use the approved physical isolation procedure and verify that the equipment cannot restart. Keep any battery terminals protected against accidental contact.

2.4 Connect power and peripherals

Only qualified personnel should alter electrical connections. Use the supplied wiring sheet and component ratings to confirm voltage, polarity, pin numbering, fuse/protection ratings, wire size and strain relief. A motor's declared voltage does not establish the complete robot's battery voltage or the acceptable voltage of a sensor port.

1. Isolate power before fitting or removing power wiring. Inspect connectors and route cables away from wheel travel, pinch points and sharp edges.
2. Confirm that each branch supplies the voltage and peak current required by its actual device. Do not connect a Jetson or LiDAR directly to an unspecified battery rail.
3. Use the identified USB interface for computer communication. GPIO labels in this manual are firmware references, not a harness connector pinout.
4. Confirm RS485 A/B naming, common/reference arrangement, isolation, termination and transceiver direction control from the delivered electronics documentation. Do not connect RS485 A/B directly to ESP32 UART pins.
5. Secure sensor mounts and preserve the sensor field of view. Recheck clearance with the payload and all cables fitted.

Charging, cleaning, transport and storage procedures are covered in the operation and care chapters. Match them to the actual battery, charger, lifting points and component instructions supplied with the unit.

3 Operating the delivered robot

3.1 Identify the installed operating profile

The *delivered-unit configuration record* determines which instructions apply to your robot. Check its controller firmware version, approved operating conditions, power controls, physical stop arrangement, battery and charger before applying power. A computer, LiDAR or web page being present does not establish that autonomous navigation or external motion commands are enabled.

Profile	What it supports
A: standalone demonstration	The reviewed demonstration firmware controls the wheel drivers and repeats a timed movement sequence. It can start automatically after its startup countdown. The serial commands in this chapter apply to this profile.[S4]
B: external command receiver	The Lua controller expects a receiver for four-wheel motion commands. That receiver is absent from the reviewed ESP32 sketches. Use this profile only when the delivered-unit record identifies an installed, validated receiver and its operating instructions.[S5][S6]
Diagnostic image	The reviewed serial heartbeat sketch only prints a message and uptime. It does not accept motion commands or control the wheel drivers. A heartbeat confirms that this sketch is running; it does not confirm robot readiness.[S5]

Automatic movement is possible with Profile A

After successful driver initialization, the reviewed firmware counts down for eight seconds and begins driving. Treat power-on, a controller reset and a USB connection that resets the board as possible automatic-start events. The countdown is not a substitute for the unit's physical stop and operating procedure.

3.2 Before every powered session

1. **Confirm the configuration.** Match the unit identifier and firmware profile to the delivery record. If either is unknown, have the responsible technical contact identify it before a movement test.
2. **Prepare the area.** Use a supervised, clear test area within the unit's approved conditions. Allow for forward and reverse travel and a turning footprint. Keep people, animals, cables and loose objects outside the movement area.
3. **Inspect the robot.** Check that wheels, body panels, sensor mounts and payloads are secure; cables are clear of moving parts; and there is no visible damage, leakage or loose power connection. Resolve defects before operation.
4. **Confirm stopping arrangements.** Locate the physical stop and the documented motor-power isolation control. The operator must know their function and have immediate access. Complete the unit-specific stop check before allowing floor movement.
5. **Check power and supervision.** Use the battery condition checks and power-on sequence specified for this unit. Assign one operator to command motion and remain with the robot throughout the session.

3.3 Starting a Profile A demonstration

Profile A is an open-loop demonstration, not a waypoint navigator. The reviewed settings request 90 RPM for straight motion and pivots. They alternate forward for five seconds, reverse for five seconds, left and right timed pivots of three seconds each, then left and right timed pivots of six seconds each. A one-second zero-command interval separates movements, and the sequence repeats. The nominal “360” and “720” names describe intended turns; actual angles depend on the surface, load and slip. These settings are source-code values, not an approved speed or travel envelope for every delivered unit.[S4]

1. Complete the pre-operation checks. Keep the robot in the unit’s documented condition that prevents unintended movement while preparing the connection; do not rely on sending a serial command quickly enough to catch the countdown.
2. Connect the computer to the ESP32’s documented USB interface. Select the port assigned by this computer, **115200 baud**, and a line ending of **Newline** or **Both NL & CR**. Only one program should own that serial port.
3. Follow the unit’s power-on procedure. Expect startup messages and driver initialization. If initialization fails, keep the robot secured and investigate the reported fault.
4. To cancel an active countdown, send `PAUSE` or `STOP`; `ESTOP` also cancels it and requests driver disable. During the countdown, the reviewed program handles only those commands. Send `STATUS` after the countdown has been cancelled or normal command processing has begun.
5. Verify the robot is physically stationary and read its status. A paused state is the appropriate place to complete the session briefing. Confirm the area is clear before releasing the unit’s physical restraint or stop according to its instructions.
6. When ready for the timed demonstration, send `RESUME`. This starts again from the forward segment **without another eight-second countdown**. Watch the robot continuously and be ready to stop it.

Opening a serial monitor can reset the controller

Some USB/board configurations reset the ESP32 when the port opens. The reviewed demonstration firmware has one startup path; it does not implement the power-on versus USB-reset “fast-start” distinction described in the Lua commentary. If a startup banner or countdown reappears, treat it as a new startup.[S4][S6]

To pause normally, send `PAUSE` and verify that motion stops. Closing the serial monitor, unplugging USB or closing a web page does **not** stop the standalone demonstration. It runs on the ESP32 independently of the computer.

3.4 Profile A command reference

Send one command per line. The normal parser ignores surrounding whitespace and accepts either letter case. An unknown command prints help; it does not automatically stop a moving robot.[S4]

Command	Effect in the reviewed firmware
`PAUSE` or `STOP`	Pauses the sequence and requests zero speed at all wheels. Drivers remain enabled; this is not motor-power isolation.
`RESUME` or `START`	If drivers are ready, starts a new sequence from forward. It does not continue the interrupted segment or repeat the startup countdown. If drivers are not ready, the command is rejected.
`ESTOP`	Pauses the sequence, requests zero wheel speeds, then requests disable on both motor drivers. The program marks drivers not ready. Read each reported result and verify the actual stop.
`INIT`	Stops the sequence and attempts driver initialization: identity checks, disable, alarm clear, operating mode and ramp settings, then enable and zero-speed requests. On success, the normal command path remains paused until `RESUME`.
`STATUS`	Reports the firmware's ready/running flags, sequence state, cycle count and wheel-command acknowledgement/failure counters.
`HELP`	Displays the command list. It does not change the motion state.

INIT enables the motor drivers

`INIT` is an active initialization operation, not a read-only check. It requests alarm clearing and enables driver outputs. Use it only after addressing the cause of a stop or fault and preparing the robot for possible actuation. A failed initialization can leave one driver configured before the other fails; "NOT READY" does not establish that both outputs are electrically disabled.

3.5 Reading status correctly

A successful Modbus write acknowledges register acceptance; it is not a measurement of wheel speed or proof of a physical stop. `STATUS` does not report encoder readings, pose, battery charge or current. The text "All wheels stopped" printed after `PAUSE` is a program message, not a sensor-verified observation.

The cycle counter increases whenever the program enters the forward segment, including a `RESUME` restart. A successful pair of wheel-speed writes adds two to the acknowledgement counter; each failed driver write adds to the failure counter. Stop commands can change these counts too. `PAUSE`, `ESTOP` and `INIT` do not clear the accumulated totals. Clearing a driver alarm is different from clearing these software counters. A controller reboot resets the RAM totals and may also trigger automatic startup, so it is not a routine "clear counts" operation.[S4]

3.6 Stopping, shutdown and recovery

For a normal Profile A stop, send `PAUSE` or `STOP`, watch the wheels and confirm the robot has come to rest. If motion is unexpected or a stop request is ineffective, use the unit's physical stop immediately. Serial `ESTOP` uses the same communications path as motion commands and cannot replace an independent physical stop.

To finish a session, stop and verify motion first. Apply the documented motor-power isolation or secure-stop procedure, then shut down any installed navigation computer normally before following the unit's final power-off sequence. Disconnect charging or external cables as required for moving or storing the unit. Leave it secured against rolling and unintended restart.

Observation	Operator response
Unexpected movement or no response to a stop	Use the physical stop. Keep the movement area clear, isolate as instructed for the unit, and report the event. Do not continue by repeatedly sending motion commands.
`MODBUS ERROR`, failed disable, or `NOT READY`	Keep motion inhibited. Record the displayed driver ID and error. Have power, cabling and controller/driver configuration checked before an initialization attempt.
Only periodic "ESP32 LIVE" messages	This matches the diagnostic heartbeat image, not the motion controller. Confirm the installed firmware with the technical contact.
No serial text or unreadable text	Check the assigned port, 115200 baud, USB cable and that another application has not opened the port. Treat reconnecting or reopening as a possible controller reset.
Startup countdown appears unexpectedly	Follow the automatic-start precautions, cancel with `PAUSE`/`STOP` when possible, and determine why the controller restarted before proceeding.

After the cause is resolved, recovery requires a deliberate operator decision. Recheck the area, power state and physical stop. For Profile A, use `INIT` only when initialization is required and safe, inspect its results, and verify stationary operation before sending `RESUME`. Recovery should not be attempted solely because an error message has disappeared.

Communication failure is not a confirmed stop

The reviewed firmware pauses its demonstration after repeated failed wheel-command attempts and tries to send zero speeds. A broken bus can prevent that zero command from reaching the drivers. Missing acknowledgements therefore mean uncertain actuation, not stopped wheels.[S4]

3.7 Using a future external-controller profile

The Lua design sends newline-terminated `HELLO` and `CMD,FL,FR,RL,RR` messages over a 115200-baud serial link. Wheel values are in radians per second, ordered front left, front right, rear left, rear right; positive values mean vehicle-forward. A matching receiver, rather than Profile A, is needed to interpret them. Neither reviewed ESP32 sketch implements that receiver.[S4][S5][S6]

Before using Profile B, the delivery record must identify the installed receiver version and its approved host application, connection, speed limits, start/stop/rearm procedure and link-loss behaviour. The proposed split keeps the ESP32 responsible for the motor-driver bus. Do not connect two independent Modbus masters to the same bus.

The Lua comments refer to a 400 ms receiver watchdog; this is not implemented by the supplied ESP32 files. An installed receiver must be tested against its own documented timeout using its local clock. Loss of the host, USB link or command stream must be handled by that receiver, with deliberate rearming after recovery. Do not infer real-robot protection from a successful browser exercise or a Lua comment.

3.8 Charging and battery care

Use only the charger, charging connection and procedure specified for the delivered battery and unit. Check their identifiers before connection. The source package does not establish one battery chemistry, charge voltage, capacity or charger suitable for all units; motor information is not a battery-charging specification.

Park and secure the robot, inhibit motion and follow the approved charging location, ventilation, temperature and connection sequence. Inspect accessible connectors and cables for damage before use. Do not charge a damaged, leaking, unusually hot or otherwise abnormal battery; isolate the unit as its instructions require and contact the responsible service provider. Use the installed battery/BMS indicators according to their documentation, rather than estimating charge from demonstration runtime or the serial `STATUS` command.

3.9 Cleaning, storage and transport

Clean only by the methods permitted for the delivered enclosure, electronics and sensors. Keep charging connectors dry and protect optical surfaces from scratching. Do not assume that brochure protection ratings authorize rain exposure, pressure washing or chemical cleaning for this particular prototype. Establish the actual unit's limits before those conditions are encountered.

For storage, power down and secure the robot, protect it from moisture and accidental activation, and follow the battery supplier's storage charge, temperature and inspection intervals. For transport, remove or secure loose payloads, protect protruding sensors, restrain the chassis using approved points and prevent wheel movement. Use the measured unit mass and documented lifting/handling method; body panels and sensor mounts are not assumed lifting points. Obtain battery-specific transport instructions for the intended journey. Record damage or unusual behaviour and resolve it before returning the robot to service.

4 Using the Navigation Studio

The supplied V9 website is an interactive way to explore the robot, practise routes and inspect example controller messages. Open the package's `index.html` using its supplied launch instructions. Select **3D explorer** to inspect the original assembly on that same page at `index.html##explorer`; select **Drive Lab** for the simulator at `simulator.html`.[S7]

The Studio operates entirely in simulation

No real robot is connected. The controls move a browser model, and the displayed serial and Modbus messages are examples. JSON downloads are local specimens; they are not published commands that an ESP32 can retrieve.

4.1 Your first mission

1. Choose an arena if desired: **Indoor lab**, **Greenhouse aisles** or **Open test area**.
2. Press **Start example mission**. This loads the example route and starts the simulated robot at 0.50 m/s. No separate manual-start action is needed.
3. Watch the current goal, distance, route and navigation messages. The view follows the robot at a closer zoom; **Fit arena** reveals the complete test area.
4. Press **Stop simulation**, or Space outside a text input, to stop. When offered, **Resume mission** continues the retained mission progress. Starting an example again loads a new example run.

Control	Use
Start manual driving	Enable manual simulation, then hold W/A/S/D, arrow keys or an on-screen direction button. Release to stop.
Stop & disable	Stop motion and disable simulated driving.
Emergency stop [X]	Latch the simulation's stop state. **Reset emergency stop** clears the latch but does not resume motion; explicitly start driving or a mission again.
Run my waypoints	Click clear ground in Map view, or enter waypoint coordinates, then start the route. While a mission runs, this control becomes **Restart my route**.
Map view / **Isometric**	Change presentation. Zoom and **Follow robot** enlarge the view without changing the robot's size relative to the arena.

Load route only prepares an example without starting. **Clear route** removes the goals. If a fault or invalid route prevents movement, follow the message above the arena rather than repeatedly pressing Start. A browser pause or background-tab interruption can also require an explicit restart or resume.

4.2 See a new obstacle change the route

During an example mission, press **Add obstacle ahead**. The added box is initially unknown to the planner. When the synthetic LiDAR detects it, a blocked remaining route causes the model to pause, calculate a detour and continue towards the same goal when a route exists. Compare the amber dashed original route with the cyan current route and read the navigation event messages. The dashed route is the initial path for the current goal, not a complete history of every mission segment.[S7]

To demonstrate the different outcome when no passage remains:

1. Press **Test blocked route**. This restarts in the open arena and inserts a full barrier after two simulation seconds.
2. Observe obstacle detection, the failure to find a valid route and stopped motion. Mission progress is retained.
3. Press **Remove added obstacles**, then **Resume mission**. The robot continues to the remaining goal after your explicit restart decision.

This comparison demonstrates a new obstacle causing a revised path, and a blocked route causing a stop. **Add obstacle ahead** alone will not necessarily close every passage. **Test blocked route** provides the repeatable blocked-arena example. Stop, reset or starting a different activity cancels a queued test barrier.

4.3 Read and save the example messages

Below the arena, the first panel shows body speed, yaw rate, four logical wheel speeds and the serial line. The second shows corresponding Modbus RTU byte examples and expected acknowledgement frames. Positive yaw turns counter-clockwise; wheel order is FL, FR, RL, RR. Values in the serial line are rad/s, while the driver table also shows signed RPM and channel mapping.

Copy serial line copies a newline-terminated specimen. **Export JSON specimen**, **Export command log** and **Export waypoints** save local files for inspection. The command log retains the latest 300 generated commands, not the complete session. Waypoints use the local map's metres, not GPS coordinates. Expected acknowledgements are illustrations; the simulator receives no hardware acknowledgement or measured wheel telemetry.[S7]

4.4 What the demonstration does and does not model

The Studio uses a known-map A* planner, an ideal position estimate, synthetic scans and a waypoint follower. A scan hitting a new box reveals that box's entire ideal rectangle to the planner; real LiDAR interpretation is a separate engineering task. Online A* replanning is demonstrated, but the browser does not run SLAM, Nav2 or a DWA local planner.

The drawn robot and wheels occupy 0.46 by 0.34 m inside a 0.30 m circular collision boundary. These are simulation settings; the separate detail inset is a close-up. Wheel radius and track are taken from the Lua model, and the simulation omits real braking, inertia, slip and sensor uncertainty. Its speed settings and stopping behaviour are not physical operating limits or evidence of a validated robot safety system.

Under **Fault injection & reset**, command-stream loss exercises a simulated watchdog and LiDAR loss interrupts autonomy. Clearing either fault does not authorize motion. The **LiDAR on/off** view button only hides or shows the scan overlay; it does not remove the sensor from the model. Use the **Build plan** and **Messages** pages for the separate hardware integration work.

5 Care, troubleshooting and support

Maintain the robot according to its fitted battery, drive components, sensors, enclosure and operating conditions. Follow the component manufacturers' instructions and the delivered-unit maintenance record. Confirm the applicable service intervals, lubricants, battery storage settings and cleaning materials at handover.

Prepare the robot before inspection or service

Stop the mission, secure the platform against unintended movement and follow the verified isolation procedure for the delivered unit before touching moving parts or working on wiring. A browser button, USB disconnect or software zero-speed request is not a substitute for the physical stop/isolation arrangement. Do not perform work that requires access to energised drive electronics unless it is part of an approved procedure carried out by an appropriately qualified person. [S4][S6]

5.1 Routine checks

Carry out the checks below at the indicated events. Arrange periodic servicing according to the installed components' schedules and the company instructions recorded for the unit.

When	Check or action	Record or response
Before each use	Inspect wheels, accessible fasteners, sensor mounts and visible cables for damage, looseness, obstruction or strain. Check that payloads and covers are secure.	Resolve damage or loose mounting before motion. Record any change affecting footprint, mass or sensor alignment.
Before each use	Confirm battery readiness using the approved method, correct operating profile, sensor/controller status, clear test area and access to the physical stop.	Use the configuration-specific readiness and stop-check procedure; do not infer readiness only from a powered display.
After use	Stop and isolate as instructed. Inspect for debris, moisture, cable damage and abnormal heat; preserve mission/fault logs.	Record defects and operating conditions. Use only the approved charging, cleaning and storage procedures.
After impact or unusual behaviour	Remove the unit from operation. Inspect structure, wheels, mounts, sensors, cabling and payload retention.	Recheck footprint, sensor alignment and any affected calibration before return to service. Escalate hidden damage or repeated faults.
After hardware or software change	Update the configuration record and repeat the affected commissioning checks.	Record revision, reason, responsible person and test results; retain a recoverable approved software version.
Periodically	Follow the schedules for the installed battery, charger, drive and sensor components.	Have the company define any complete-system service requirements that are not covered by component instructions.

5.2 Cleaning, charging and storage

Remove loose contamination using the cleaning method approved for the configured robot. Keep sensor windows and scan paths unobstructed, and follow the sensor manufacturer's instructions for cleaning optical surfaces. Do not assume the brochure's IP54 reference permits pressure washing, immersion or exposure

of opened connectors. A new sensor opening, cable entry or removed cover can change the protection of the assembly. [S1]

Identify the installed battery chemistry, operating range, BMS and matched charger before charging or storage. Use the approved charger and connection procedure; keep the robot unable to move while charging. Charge level, temperature limits, long-term storage preparation and battery replacement criteria must come from the verified battery/charger documentation. The Lua motor-voltage declaration is not a battery specification. [S6]

If damage, leakage, an unusual smell or abnormal heating is observed, stop using the affected equipment and follow its manufacturer/company incident procedure. Do not continue a mission to reproduce a battery fault. Keep a record of the observed condition without opening a damaged pack or bypassing its protection.

5.3 Troubleshooting by symptom

Start by recording the exact message, time and operating profile. Apply only the checks within the operator's approved scope. If movement is unexpected or a stop request is ineffective, use the delivered unit's physical stop procedure and investigate before restarting.

Symptom	Likely distinction to check	Action and escalation
No computer/controller startup	Supply, connector, protection or startup issue; actual power architecture is configuration-specific.	Check the documented isolation/startup sequence and visible connections. Have a qualified person verify rails and protection; do not bridge a fuse or guess a supply voltage.
Heartbeat text but no motion	The serial-test profile prints uptime and has no motor-command receiver.	Confirm the installed firmware identifier. Obtain the approved operating profile from the company rather than sending more movement commands. [S5]
No `ACK,HELLO` response	Profile mismatch, wrong port/settings or unavailable receiver. The reviewed heartbeat and demo profiles do not implement this handshake.	Verify the intended interface, port ownership, cable, baud setting and released firmware. Do not assume a file named "CoppeliaSim" contains the receiver. [S4][S5][S6]
Motion starts after restart	The standalone demo can start its sequence after its initialization countdown.	Use the physical stop procedure. Confirm startup behaviour and replace the demonstration profile only through the approved software-release process. Disconnecting USB does not stop the standalone sequence. [S4]
Commands or ACKs displayed, but no verified motion	The display may show simulation targets or expected ACK examples; a real protocol acknowledgement still does not measure wheel movement.	Identify the data source. Check the actual controller/driver state and feedback through the approved diagnostic procedure; do not raise speed to force a response. [S4][S6][S7]

Symptom	Likely distinction to check	Action and escalation
A wheel turns in the wrong direction	Logical wheel identity, channel order or polarity does not match the unit's record.	Stop. Have the responsible engineer repeat the supported-wheel mapping test and check the approved configuration before floor operation. [S4]
Turn angle or route drifts	Timed turns are open loop; slip, effective track, wheel radius or sensor alignment can affect navigation.	Compare measured motion with command and pose logs. Recalibrate using the approved method; a timed "360-degree" label is not a heading measurement. [S4][S6]
LiDAR or pose unavailable	Missing/stale data, obscured scan view, displaced mounting or software/-configuration mismatch.	Keep navigation stopped. Check visible obstruction and recorded sensor status, then use the approved sensor/-transform diagnostic procedure. Do not bypass the missing-data condition to continue.
Mission reports a blocked route	There may be no feasible passage for the configured footprint and clearance. In the browser lab this is an intentional demonstration outcome.	Inspect the route and data. In the browser, remove added obstacles and explicitly resume. On hardware, follow its validated recovery procedure; do not reduce clearances merely to force a path. [S7]
Command link lost	Cable, host process, interface or receiver fault; a successful reconnect does not justify restoring an old movement request.	Confirm a stopped state, inspect logs/-connections and recover through the approved deliberate-restart procedure. If motion continues, use the physical stop. [S4][S6]
Runtime is short or restarts occur under load	Battery state/condition, load, converter behaviour or connection quality may differ from the validated baseline.	Record payload, route, temperatures and available voltage/current evidence. Use the matched charger and arrange approved power diagnostics. Do not treat the eight-hour brochure reference as a fault threshold. [S1]

5.4 Returning to operation

1. Record the fault and identify what was inspected, repaired or changed.
2. Confirm that covers, mounts, connectors, payloads and the stop arrangement are restored to the approved configuration.
3. Repeat the affected functional checks in the appropriate supported-wheel or controlled test setup. Check sensor alignment and calibration if disturbed.
4. Clear the fault using the approved procedure, with the robot remaining stopped until a deliberate new start. Record the verification result and any revised operating restriction.

Do not reuse a previous command or reload an old web message as a recovery shortcut. The browser package is an offline simulation and interface-inspection environment, and its generated command specimens are not an approved live-control service. [S7]

5.5 Requesting support

Contact Adaptive AgroTech through `www.AdaptiveAgroTech.com`. Record the agreed support route, service arrangements and any response-time commitment in the unit's handover or supply documents.

Provide the unit model/identifier, installed hardware and firmware/software revisions, operating profile, symptom, timestamp and steps that reproduce the issue. Attach relevant logs, photographs of visible damage or connections, the last successful operating state, recent changes and the results of approved basic checks. Identify whether each observation comes from the browser simulator, a real controller response or a physical measurement. Do not send passwords or access tokens with diagnostic material.

Service and commercial terms

At handover, obtain the service responsibilities, permitted operator maintenance, warranty terms and repair/return procedure from Adaptive AgroTech. Confirm the applicable commercial and jurisdiction terms, any required conformity documentation and service availability in the supply record. These items remain unconfirmed in the reference material; record an owner and closure date for each outstanding field.

6 External controller and motor-driver interface

6.1 Integration boundary and control ownership

The intended integration retains the ESP32 as the owner of wheel actuation. The Jetson computes navigation requests from localisation, the mission and obstacle observations; a bridge converts these requests into logical wheel velocities. The ESP32 must validate their units, limits and freshness before writing the two motor drivers. This allocation is a design basis for the integrator, not evidence that the supplied firmware already implements the complete navigation interface.

A matching external-command receiver is not supplied

The September 10 motor demonstration implements Modbus wheel writes but does not parse `HELLO` or `CMD,FL,FR,RL,RR`. The September 9 sketch is only a serial heartbeat. The Lua example implements the host side of a different ESP32 interface. A compatible, identified receiver must be obtained or implemented and verified before using these messages on the physical robot. [S4][S5][S6]

Boundary	Message and role	Evidence or required work
Jetson to ESP32	Proposed newline-delimited wheel requests over USB serial; 115200 baud.	Lua provides the host format. Receiver parsing, arming, freshness and feedback remain integration work.
ESP32 to drivers	Binary Modbus RTU requests to addresses 1 and 2; 115200 baud, 8N1.	Present in the motor demonstration; confirm the installed driver settings and electrical interface.
Driver to wheel	Driver-channel RPM targets with mounting polarity applied.	Source records the channel mapping; verify it on the delivered unit.

6.2 Vehicle coordinates and velocity conversion

Use a right-handed vehicle frame: $+X$ forward, $+Y$ left and $+Z$ up. Positive yaw rate is counterclockwise when viewed from above. Positive *logical* wheel speed always propels that wheel toward robot-forward; it is independent of the driver's electrical polarity.

For forward velocity v in m/s, yaw rate ω_z in rad/s, effective track width b in m and loaded wheel radius r in m:

$$v_L = v - \frac{b\omega_z}{2}, \qquad v_R = v + \frac{b\omega_z}{2},$$

$$\omega_{\mathrm{FL}} = \omega_{\mathrm{RL}} = \frac{v_L}{r}, \qquad \omega_{\mathrm{FR}} = \omega_{\mathrm{RR}} = \frac{v_R}{r}, \qquad n_i = \omega_i \frac{60}{2\pi}.$$

Here n_i is logical RPM. At zero forward velocity, a positive yaw request therefore reverses the left wheels and advances the right wheels.

The Lua profile uses $r = 0.0535$ m and $b = 0.250$ m. These are model inputs, not an as-built geometry certificate. Measure the delivered robot and calibrate effective track width for skid-steer slip. A timed pivot does not establish measured heading change. [S6]

6.3 Host-to-ESP32 serial contract

The following contract is reconstructed from the Lua host implementation. It specifies newline-delimited ASCII, with wheel values in **rad/s** and the fixed order **FL, FR, RL, RR**. Its configured serial rate is 115200 baud. The host formats three decimal places and ends each transmitted line with LF, hexadecimal `0A`. Its receive parser also strips CR characters, allowing CRLF replies. [S6]

Line or prefix	Meaning in the Lua host
`ESP32_READY` or `ARDUINO_READY`	Recognised readiness announcements; trigger a fresh hello while the handshake is incomplete.
`HELLO`	Host handshake request. It must not itself enable motion in a future receiver.
`ACK,HELLO`	Completes the host handshake and permits its command stream.
`CMD,FL,FR,RL,RR`	Four signed decimal logical wheel velocities. These field names are replaced by numbers on the wire.
`ACK,` and `ERR,`	Recognised reply prefixes. The host does not define a complete command-acknowledgement payload or error-code catalogue.
`STOP`	Sent after an all-zero command during host timeout handling and cleanup. Its receiver behaviour must be defined.

Illustrative exchange. The right-hand comments below describe the intended participants; they are not transmitted. `\n` denotes one LF byte, not two printable characters.

```
ESP32_READY\n                    [expected receiver announcement]
HELLO\n                          [host request]
ACK,HELLO\n                      [expected receiver reply]
CMD,9.425,9.425,9.425,9.425\n     [host: forward example]
CMD,0.000,0.000,0.000,0.000\n     [host: zero targets]
STOP\n                           [host: stop request]
```

This is an explanatory transcript, not captured traffic from the delivered robot. Neither supplied ESP32 sketch can perform the complete exchange. In particular, the motor demonstration's support for `STOP` does not make it a streaming `CMD` receiver. [S4][S5][S6]

Timing and acknowledgement limits. The Lua host probes after 0.5 seconds even if no readiness line arrives, retries hello every 1 second, and attempts commands every 0.05 seconds after handshake. It attempts reconnection every 2 seconds; an acknowledgement gap greater than 2 seconds triggers a zero/stop attempt and link closure. These intervals use *simulation time*. Every recognised `ACK,` line refreshes the host timer, so this timer does not establish that the latest individual command reached both drivers. [S6]

Define the production receiver explicitly

Specify bounds, decimal parsing, rounding, line-length limits, incomplete line handling, command ownership, arming and fault replies. The legacy line has no sequence number, session identifier or expiry field. A versioned replacement may add them, but must then document its compatibility with the Lua host.

6.4 Logical wheel mapping and the RS485 boundary

The motor demonstration records the following physical mapping. Driver channel names LEFT and RIGHT are connector/channel labels; they do not consistently identify the robot's left and right sides. Apply the polarity transformation exactly once, after logical velocity conversion. [S4]

Wheel	Driver	Channel	Register	Driver target
Front left (FL)	1	LEFT	`0x2088`	$+n_{\mathrm{FL}}$
Front right (FR)	1	RIGHT	`0x2089`	$-n_{\mathrm{FR}}$
Rear left (RL)	2	RIGHT	`0x2089`	$+n_{\mathrm{RL}}$
Rear right (RR)	2	LEFT	`0x2088`	$-n_{\mathrm{RR}}$

Thus the two consecutive channel targets are

$$\text{Driver } 1 = (n_{\mathrm{FL}}, -n_{\mathrm{FR}}), \qquad \text{Driver } 2 = (-n_{\mathrm{RR}}, n_{\mathrm{RL}}).$$

The second register, `0x2089`, follows from the source's write of two consecutive registers starting at `0x2088`; it is not declared as a separate named constant. Preserve packet order when converting rear-wheel values: driver 2 receives RR first and RL second.

Electrical connection and bus ownership. The demonstration creates UART1 with RX on GPIO 18 and TX on GPIO 17, configured as 115200 baud, eight data bits, no parity and one stop bit. These are the *source configuration*; verify that both physical drivers match. The USB host interface is a different connection from this motor UART. Do not send binary driver frames to the ESP32 USB port and assume they will be forwarded. [S4]

For the intended two-wire, half-duplex RS485 installation, use one active Modbus master: the ESP32. A direct Jetson-to-driver alternative requires explicit transfer of bus ownership and reassignment of local validation and stop responsibilities. The supplied ESP32 code does not expose a Modbus slave register map for an upstream controller. [S4][S9]

An appropriate RS485 transceiver is required between logic-level UART pins and the differential bus. No DE/RE direction-control pin appears in the sketch; automatic direction control is therefore *not proven*. Verify the transceiver, direction timing, voltage compatibility, bus reference or isolation, cable routing, termination and biasing against the actual wiring and driver documentation. The source cannot establish these physical details.

Limits and conversion policy. The Lua limits outgoing physical wheel commands to ± 150 RPM, equivalent to approximately ± 15.708 rad/s. The demonstration's straight and turning setpoints are 90 RPM. These source values are not an approved navigation speed envelope. [S4][S6]

Choose and record the conversion from decimal rad/s to integer RPM, including a tie-breaking rule. The demonstration accepts integer RPM and therefore defines no floating-point rounding policy. A new bridge should also document saturation behaviour: uniformly scaling all wheel requests preserves requested curvature, whereas the Lua independently clips each wheel. Match the chosen policy across host, receiver and test vectors.

Web messages remain a separate future interface

The web laboratory generates simulation message previews. The supplied ESP32 sketches contain no HTTP or JSON receiver. A later web interface needs authentication, command expiry and local stop authority; a saved JSON value must never act as an indefinitely valid velocity command. [S4][S5][S7]

6.5 Modbus framing and a worked 90 RPM transaction

The demonstration uses function `0x10` to write both speed targets of one driver in one request. It uses literal protocol register addresses; do not add a `40001` display offset. Multi-byte register addresses, counts and data are transmitted high byte first. Signed RPM is encoded as 16-bit two's complement. The CRC is transmitted low byte first. [S4][S8]

Byte positions	Example	Interpretation
0	`01`	Addressed driver
1	`10`	Write multiple holding registers
2–3	`20 88`	Starting register
4–5	`00 02`	Two consecutive registers
6	`04`	Four data bytes
7–8	`00 5A`	First channel: $+90$ RPM
9–10	`FF A6`	Second channel: -90 RPM
11–12	`82 31`	CRC over bytes 0–10, low byte first

Forward motion example. All logical wheels receive $+90$ RPM:

$$\omega_i = 90\frac{2\pi}{60} = 9.42477796\ \mathrm{rad/s}.$$

The Lua-formatted value is `9.425`; a nearest-integer conversion returns 90 RPM. Driver 1 then receives $(+90, -90)$, and driver 2 receives $(-90, +90)$:

```
Driver 1 request:
01 10 20 88 00 02 04 00 5A FF A6 82 31
Driver 1 expected successful reply:
01 10 20 88 00 02 CA 22

Driver 2 request:
02 10 20 88 00 02 04 FF A6 00 5A 3C E0
Driver 2 expected successful reply:
02 10 20 88 00 02 CA 11
```

These bytes were independently calculated and CRC-checked for this manual; they are not hardware captures or an instruction to initiate motion. CRC-16/Modbus starts at `0xFFFF`, uses reflected polynomial `0xA001`, and includes every byte preceding the CRC. For example, the first request has CRC value `0x3182`, transmitted as `82 31`. The full frame has a zero CRC residual.

Zero and left-pivot cross-checks. All-zero targets produce the following requests. Their successful replies remain the same eight-byte replies shown above.

```
Zero, driver 1: 01 10 20 88 00 02 04 00 00 00 00 63 A8
Zero, driver 2: 02 10 20 88 00 02 04 00 00 00 00 6C EC
Left pivot at logical (-90,+90,-90,+90) RPM:
Driver 1: 01 10 20 88 00 02 04 FF A6 FF A6 72 15
Driver 2: 02 10 20 88 00 02 04 FF A6 FF A6 7D 51
```

A successful function-`0x10` reply echoes the start address and quantity, not the speed data. It acknowledges the register transaction; it does not measure RPM, prove that all four wheels moved, or confirm that the chassis has stopped.

6.6 Driver initialisation and response handling

The sequence below describes what the supplied motor demonstration *actually does*. It configures driver 1 and then driver 2. This is an implementation record, not an approved production startup procedure. [S4]

Step	Function	Register/value	Operation
1	03	0x2001, read one	Require the node-ID readback to equal the addressed driver ID.
2	06	0x200E = 7	Disable driver.
3	06	0x200E = 6	Clear alarm.
4	06	0x200D = 3	Select velocity mode.
5	06	0x2080 = 1200	Left acceleration time, interpreted as milliseconds by the source.
6	06	0x2081 = 1200	Right acceleration time.
7	06	0x2082 = 1000	Left deceleration time.
8	06	0x2083 = 1000	Right deceleration time.
9	06	0x200E = 8	Enable driver.

An 80 ms delay follows each successful setting write. Once both drivers have configured successfully, the demonstration writes zero speed to both. It then performs an eight-second countdown and automatically starts its repeating motion sequence unless a recognised stop input cancels the countdown. The supplied sketch contains no separate fast-start behaviour for USB reset versus electrical power-on. [S4]

Initialisation does not establish a safe arm state

Each driver is enabled before the final common zero write. If the second driver fails to configure, the source does not explicitly roll back and disable the first. A production bridge needs a reviewed startup sequence that clears stale targets, remains stopped or disabled until explicitly armed, and handles partial initialisation failure on both drivers.

What responses are checked. The common receive routine first verifies a minimum frame length and CRC. The function-03 reader then requires a seven-byte response with the correct driver ID, function and two-byte data count. The function-06 writer requires an exact eight-byte echo of its request. The function-10 writer requires eight bytes with matching ID, function, start register and register quantity. A timeout, CRC failure, wrong field or exception reply causes failure; exception codes are not decoded into detailed diagnoses. [S4]

Paired commands and feedback. Driver 1's transaction completes before driver 2's transaction begins. Updating all four wheels is therefore neither simultaneous nor atomic. One driver can accept a request while the other fails. A revised bridge must report the two outcomes separately and apply its defined fault response to the whole vehicle.

The supplied source reads the node-ID register but does not poll encoder position or actual wheel velocity. Establish the correct feedback register map, signedness, scale, rollover behaviour and sampling times from the identified driver revision before designing odometry. Keep three statuses distinct: host command accepted, driver write acknowledged, and measured motion achieved.

6.7 Watchdogs, stop behaviour and integration acceptance

The host and motor demonstration have different timing mechanisms. Their values must not be combined into an unsupported physical stop claim.

Mechanism	Actual source value or limitation
Motor-demo command refresh	100 ms nominal interval. The printed 250 ms message and four-times-per-second comment are stale.
One Modbus transaction	Blocking timeout of 250 ms, plus surrounding work. Reception may finish after at least five bytes and 5 ms of silence.
Consecutive-failure threshold	Three failed paired sends; a pair succeeds only when both drivers acknowledge.
Motor-demo software `ESTOP`	Attempts zero targets, then a disable write to each driver. All depend on the same communication path.
External-command watchdog	Absent from both supplied ESP32 sketches. The 400 ms value is an expectation stated in Lua comments.

[S4][S5][S6]

Two serially executed 250 ms timeouts already occupy approximately 500 ms for one failed pair. Three failures consequently do not imply a 300 ms stop, and the supplied code cannot substantiate a 400 ms external-command stop. Subsequent zero or disable writes may also fail on a broken bus. Blocking serial-line reads and other work can add delay. The source's software stop does not establish an independent motor-power-removal circuit.

Required receiver behaviour. Use an onboard monotonic clock for the age of the last *valid* motion request, independent of Jetson, browser or simulation time. Specify a bounded parser and reject non-finite, malformed, overlong or out-of-range input. Invalid input must not renew command freshness. Define how queued, duplicate and out-of-order commands are rejected in a versioned production protocol. A reconnect, hello or restored sensor must not silently restore a previous nonzero target.

Document distinct disabled, armed, active and fault states, an explicit control-authority policy, and deliberate recovery after reboot, E-stop or communication loss. Define the whole-vehicle response to one-driver failure. A 400 ms receive timeout can be evaluated as a design target derived from the Lua expectation; actual stop latency and distance must be measured with the configured ramps, payload and surface.

Evidence required before physical navigation. Complete the following staged checks and retain their results:

1. Identify the installed ESP32 firmware, driver revisions, wiring, geometry and independent physical stop provisions.
2. Verify byte order, CRC, wheel order, polarity and limits offline; test receiver parsing and freshness without motor power.
3. With an appropriate commissioning setup, verify wheel identity and direction individually before coordinated low-speed motion.
4. Inject host disconnect, stale input, malformed frames, partial driver failure and reboot. Measure detection-to-stop time and confirm that recovery requires deliberate arming.
5. Verify odometry and localisation feedback, then conduct supervised navigation and obstacle-stop trials within an agreed operating envelope.

7 Delivery, commissioning and service records

7.1 Delivered-unit identification

The supplier and customer complete this record together. Attach the unit-specific wiring drawing, component instructions, firmware release note and acceptance results. Record “not fitted” for absent options. A blank entry does not mean a function is available or a limit is unrestricted.

Identification / configuration	Delivered value or attached record
Customer / site	
Robot model / SKU / serial number	
Hardware revision / delivery date	
ESP32 board / firmware build ID	
Power-on behavior / automatic motion	
Control profile: demo / external / other	
Driver make, model and firmware	
Driver IDs and wheel polarity verified	
High-level computer / OS / software	
LiDAR / IMU / GNSS / other payload	
Measured loaded dimensions / mass	
Approved payload / mounting drawing	
Approved speed / route / surface	
Approved slope / step / environment	
Measured stopping test / report ID	

Information needed to operate this unit

Record the confirmed control profile and power-on behavior before the operator connects a terminal or powers the drive. The brochure limits and Lua scene settings must not be substituted for the approved values above.

7.2 Power, safety and support record

Item	Delivered value or attached record
Battery make / model / chemistry	
Battery nominal / full-charge voltage	
Battery capacity / BMS / protection	
Approved charger model / settings	
Charging connector / polarity sheet	
Fuse / branch-protection schedule	
Main isolation procedure / location	
Physical stop circuit / test result	
Remote stop / watchdog test result	
Grounding / RS485 wiring drawing	
Controller connector/pinout sheet	
Cleaning / ingress / temperature limits	
Lifting points / transport instructions	
Warranty / service agreement reference	
Supplier address / service contact	
Required conformity documents supplied	

7.3 Release to the operator

Demonstrate startup, the supplied control mode, normal and physical emergency stopping, fault recovery and shutdown. For delivered external/autonomous control, also demonstrate communication loss, sensor loss and obstacle response. Record the test area and load; a bench test alone does not establish safe floor operation.

Supplier / technician _____

Customer / trained operator _____

Acceptance record ID / date _____

Open items and restrictions _____

7.4 Commissioning evidence checklist

	Check	Record as evidence
☐	Identity and inventory	Serial number, firmware build, supplied options and component instructions.
☐	Wiring and protection	As-built drawing, correct voltages/polarities, protection and isolation checks by qualified personnel.
☐	Supported wheel test	Correct FL/FR/RL/RR directions, both driver IDs, zero command and motor disable; setup prevents uncontrolled motion.
☐	Startup and restart	Actual behavior after power-on, USB connection, reset and brownout; no unplanned restart in the delivered operating mode.
☐	Normal and emergency stop	Actual stopping behavior under approved load/surface; independent physical stop path verified.
☐	Communication failure	Cable loss, missing/corrupt replies and stale commands; bounded response and deliberate restart.
☐	Motion repeatability	Straight drive and turning, measured wheel geometry and slip effects documented.
☐	Navigation option, if fitted	Measured localization, footprint and sensor coverage; route execution, detour/no-route behavior and sensor-loss response.
☐	Power and thermal behavior	Startup/peak load, low battery behavior and temperatures within the installed components' limits.
☐	Operator handover	Operator demonstrates stop/recovery; remaining restrictions and service arrangements accepted.

7.5 Service log

Record work that can change behavior: replacement wheels, motors or drivers; battery/charger changes; payload mounts; sensor calibration; controller or navigation-software updates. Repeat affected commissioning checks before returning the robot to operation.

Date / hours	Inspection, work, parts and test result	Technician

8 Illustrated platform and image reference

All distinct robot visuals supplied with the project are retained here. Captions separate physical prototype photographs from CAD screenshots and concept artwork. Illustrations show design intent; use the unit-specific assembly and wiring drawings for installation or service. The figure inventory in the LaTeX package records original sources and image dimensions.

8.1 Exterior concept and physical prototype

Figure 2. Front concept visualization. The enclosure and wheel appearance are illustrative.

Figure 3. Rear concept visualization of the modular body and cover arrangement.

Figure 4. Actual photograph of the supplied physical prototype, front three-quarter view.

Figure 5. Actual photograph from the opposite three-quarter angle. These are reference build photographs.

Use photographs for identification, not hidden specifications

The two photographs document the physical base. They do not establish internal wiring, battery chemistry, payload rating, certified protection or autonomous capability. Match the delivered serial number and revision to its handover photographs and drawings.

8.2 CAD and modular construction

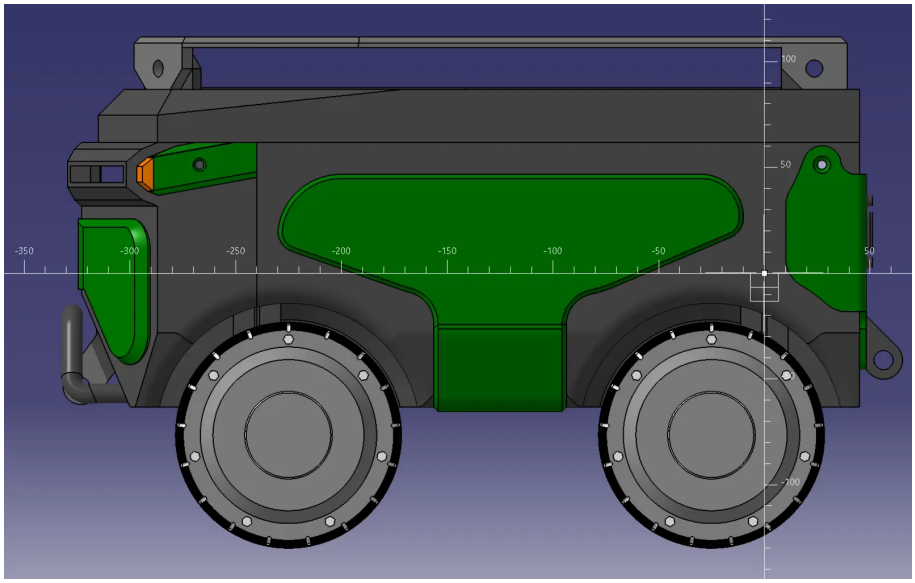

Figure 6. Side elevation of the supplied CAD model. It is a design screenshot, not a dimensioned fabrication drawing.

Figure 7. Exploded concept visualization showing top covers, body structure and four hub-wheel assemblies. The separated parts illustrate modularity; the image is not a service disassembly sequence or an as-built electrical drawing.

8.3 Sensing and field-application concepts

Figure 8. Concept with a raised sensor mast and roof equipment. Exact sensor models, mounts and fields of view must be specified for the delivered unit.

Figure 9. Field-application concept. This is supplied artwork, not a record of a successful physical field trial.

Figure 10. Crop-row application concept showing a sensor-equipped robot. Row clearance, localization and terrain suitability require validation on the actual configuration and site.

8.4 Design development and historical view

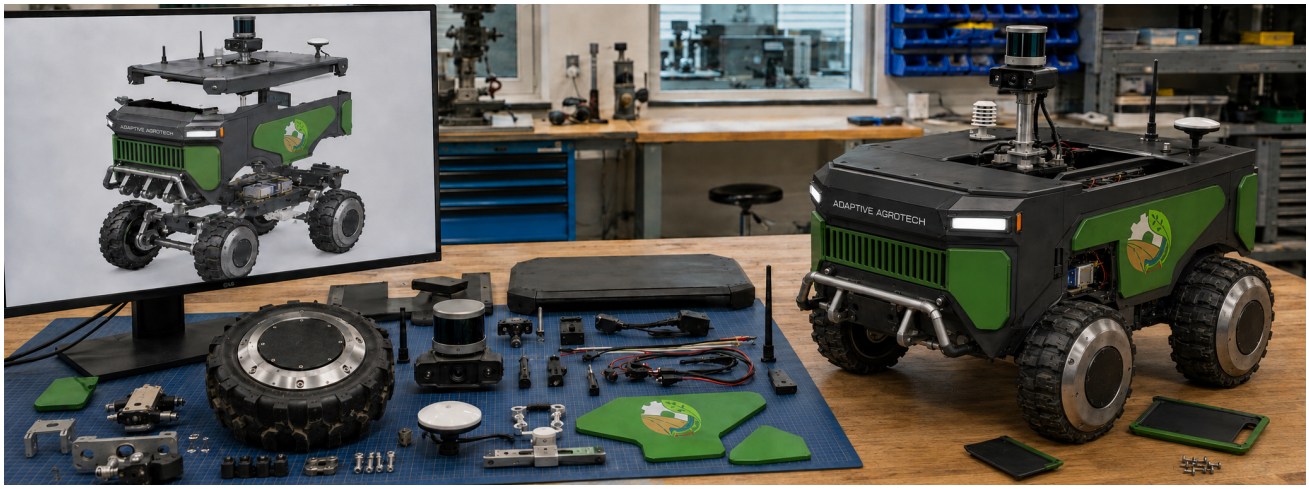

Figure 11. Conceptual workshop montage showing design, assembly and integration. The source presentation explicitly identifies this scene as a conceptual static poster; it is not a photograph of manufacturing facilities.

Figure 12. Earlier brochure concept. The lower-resolution source is reproduced at a smaller size.

Why the images differ

The project includes earlier body concepts, later exterior artwork, a CAD reference and photographs of a printed base. They document design development rather than one invariant commercial specification.

The earlier visual has a different body and sensor arrangement. Record the delivered hardware photograph, revision and optional equipment alongside the signed configuration sheet.

Attachments require their own operating documentation

The next plate includes proposed mowing attachments and a labeled assembly concept. It does not establish that cutters, two battery packs or the depicted sensor arrangement are fitted. This manual covers the mobile monitoring platform; operating a cutting attachment requires its own released instructions, safeguards and acceptance record.

CONCEPT ASSEMBLY PLATE

Illustrated assembly and proposed attachments

Figure 13. Supplied labeled concept showing mounting/cover plates, proposed battery packs, controller, motor driver and hub wheels, alongside two mowing concepts. The 25 cm dimension refers to the illustrated front cutter, not robot width. Layout and labels are design references, not as-built instructions.

8.5 Source architecture and mission concept

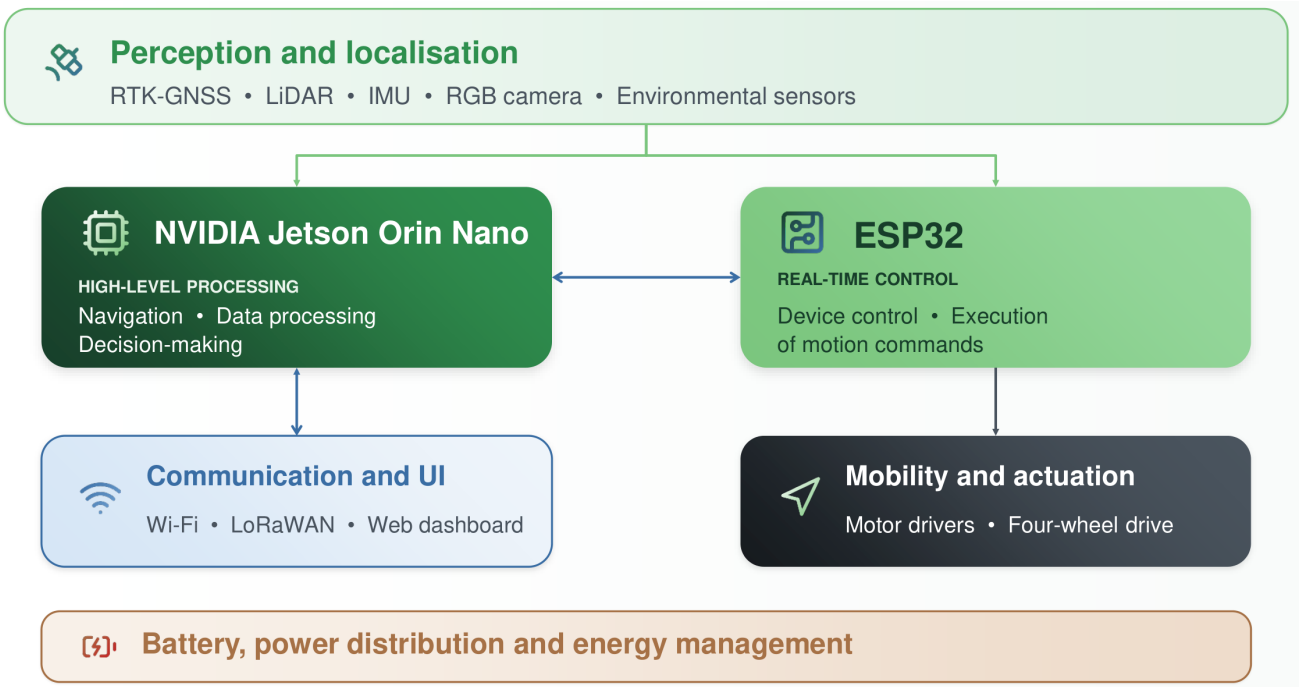

Figure 14. Original target system architecture: sensing, Jetson processing, ESP32, communication and power. This functional source diagram does not specify connector pins or prove that each subsystem is installed. See Sections 2 and 6 for the documented serial/Modbus responsibilities.

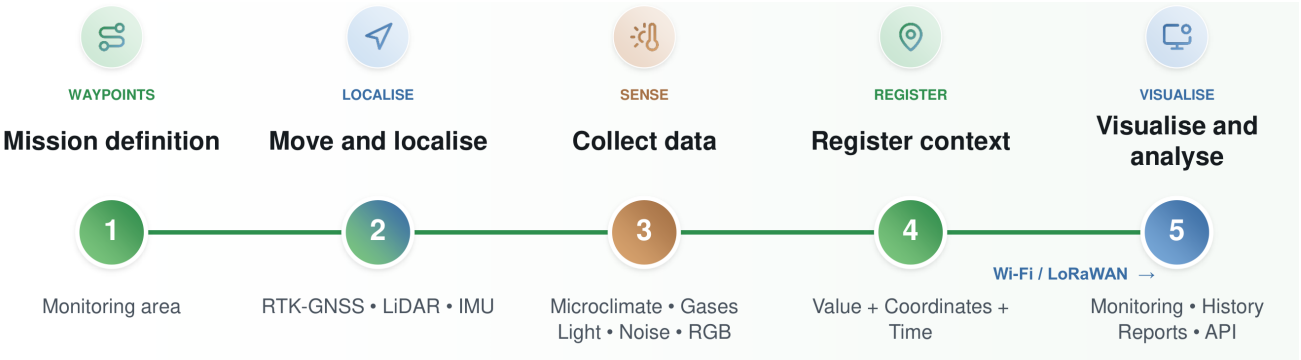

Figure 15. Original target mission workflow: define a route, localize, sense, register location/time and visualize results. Actual data channels and autonomy functions depend on the delivered software and sensors.

Positioned measurements need measured position

A useful monitoring record combines value, unit, timestamp, position, coordinate frame and quality/status. A commanded wheel speed is not a measured position. Calibrate installed sensors and verify the localization source before relying on a spatial data product.

9 Source register and terminology

9.1 Technical basis of this edition

Source tags identify the evidence behind configuration claims. The sources below are the basis of this reference edition, not an indication that raw source files are distributed to customers. The unit's release documentation takes precedence when it records a different verified configuration.

S1 Adaptive AgroTech, supplied FieldScout brochure. Original project package: *FieldScout Brochure*. Source of reported envelope, approximate mass, speed, duration, temperature and IP54 claims. These values are not new measurements in this manual.

S2 *ProjectFieldScoutAdaptiveAgroTech.pdf*, Experts in Teams 2026 project description. Platform intent, modular architecture and target sensor/compute options. Earlier proposed tasks do not override the current instruction to retain the existing controller and drivers.

S3 *FieldScout_Redmond_AdaptiveAgroTech_Sep_23_2026_.pptx*, supplied presentation. Target system architecture and reference specifications. Its filename contains a presentation date; it is not a hardware acceptance date.

S4 *2026_Sep_10_FieldScout_ESP32_RobotDemo.ino*. Source-inspected standalone motion demonstration. Establishes actual command parser, automatic startup, wheel polarity, register writes, timing constants and transaction checks described here.

S5 *2026_Sep_09_FieldScout_ESP32_CoppeliaSim.ino*. Despite the filename, supplied contents are a serial heartbeat test; they do not implement a wheel-command receiver.

S6 *RobotLuaScript.txt*, controller version 12.1 for CoppeliaSim 4.10.x. Establishes the expected serial contract and scene geometry. A declared receiver watchdog or motor rating in this script is not evidence that the delivered ESP32 firmware or hardware implements it.

S7 FieldScout Navigation Studio, version 9.0.0. Browser simulation and integrated original 3D explorer: https://adaptiveagrotech.com/FieldScout/index.html. Navigation and message displays remain simulated.

S8 Modbus Organization, *Modbus Application Protocol Specification V1.1b3*, 26 April 2012. Function-code and data-encoding reference. https://www.modbus.org/file/secure/modbusprotocolspecification.pdf.

S9 Modbus Organization, *Modbus over Serial Line: Specification and Implementation Guide V1.02*, 20 December 2006. Serial framing, CRC and bus implementation reference. https://www.modbus.org/file/secure/modbusoverserial.pdf. S8–S9 links checked on 21 September 2026.

9.2 Terms used in this manual

Term	Meaning
FL / FR / RL / RR	Front left / front right / rear left / rear right, relative to the robot's forward direction.
Logical wheel speed	Positive means that wheel drives the robot forward; driver polarity is applied separately.
RPM / rad/s	Revolutions per minute / radians per second; $1\,\mathrm{rev} = 2\pi\,\mathrm{rad}$.
ACK	Protocol acknowledgement; it is not measured proof of wheel movement.
Watchdog	Timeout supervision that must force a defined response when fresh commands cease.
Footprint	The occupied or conservatively protected area of the robot and its payload.
Odometry	Estimated motion from measured feedback; commanded speed alone is not measured odometry.
BMS	Battery management system; exact protective functions depend on the installed battery.

Keep the documentation with the robot

File the signed delivery record and later service/firmware updates with this manual. For support, start at adaptiveagrotech.com or use the service contact entered at delivery. Product, company and component names remain the property of their respective owners.