

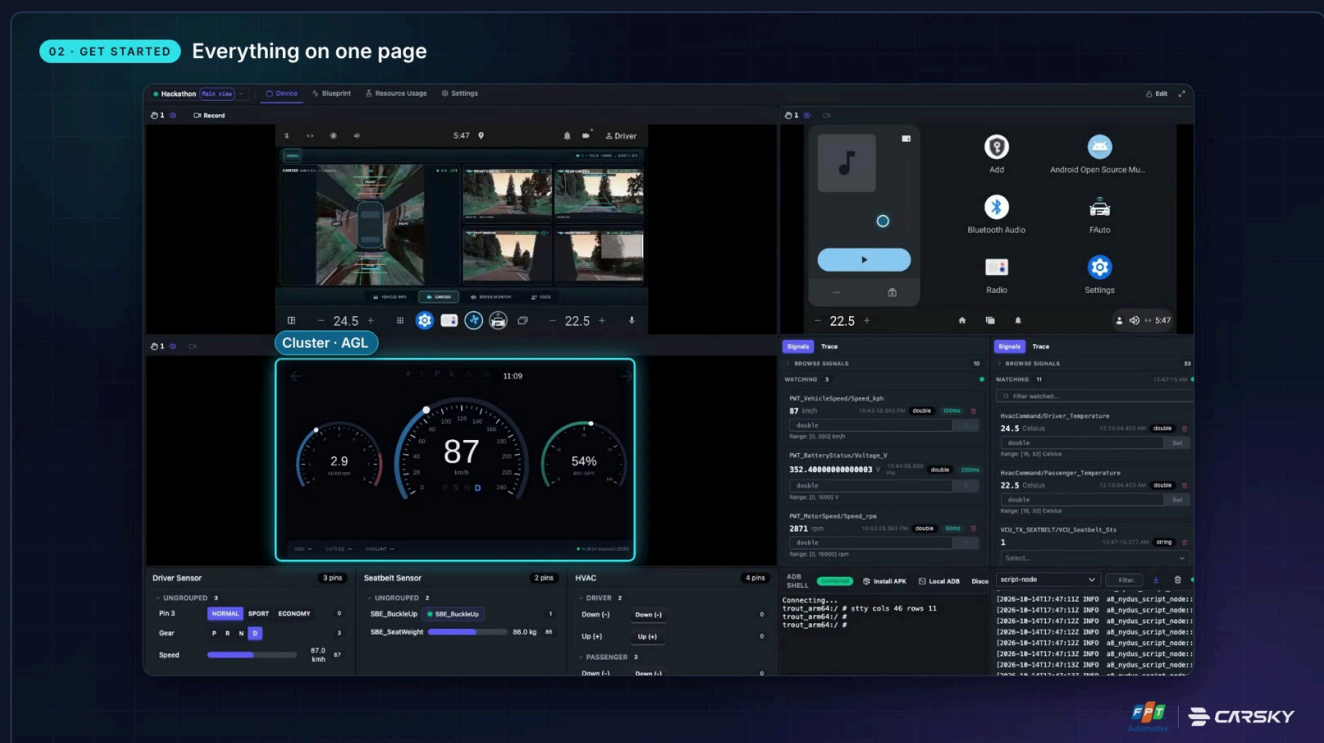


Hackathon Starter Kit

Participant Guide

Build and test in-vehicle software in the cloud — before hardware is ready.

This guide follows the video, chapter by chapter. Read along, then keep it open while you build.



Video: CarSky Hackathon Starter Kit — Guide v2 (16:55, 12 chapters)
FPT Automotive · AWS Hackathon Tokyo 2026

Before you start

1. **Get your team account.** Scan the QR code (or e-mail support.carsky@fpt.com) with your team name, company, contact name and contact e-mail. One account per team.
2. **Use Chrome or Edge** on a laptop, window at least 1920×1080.
3. **Open CarSky for the hackathon** at hackathon.carsky.io and sign in with the account you received.
4. **CarSky → Device → Connect → Focus Screen.** Find your device (called *Hackathon* in the video), press **Connect**, then **Focus Screen**.
5. **Mobile app simulator:** open mobile.live.carsky.io on your phone and sign in. Right-click (long-press on a phone) to pick your vehicle; wait for **Connected**.
6. **Full guideline:** guide.hackathon.carsky.io (Luau, DBC & VSS, APK & AGL, MCP).



Scan to request your team
account

support.carsky@fpt.com

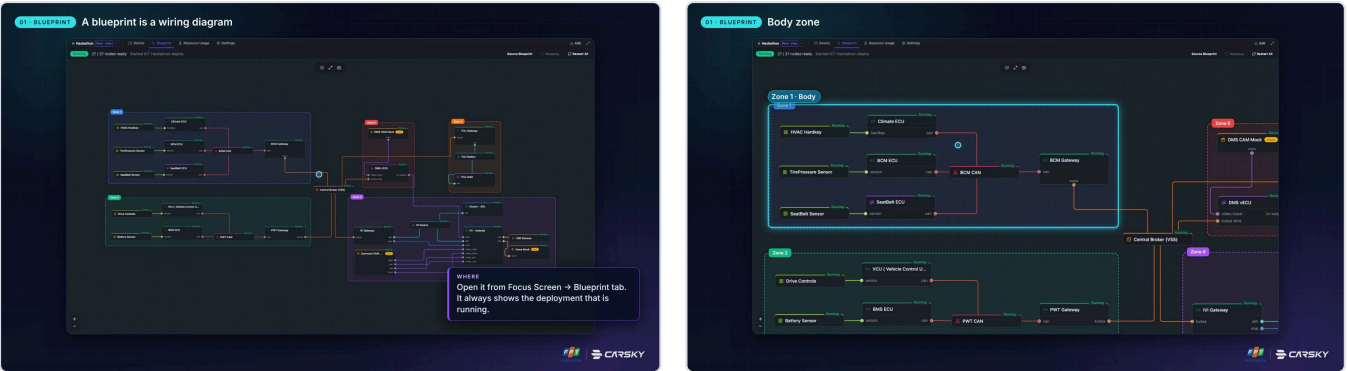
Contents

01	Your vehicle: the blueprint	Video at 0:34
02	Get started	Video at 2:38
03	The digital cockpit	Video at 4:10
04	Sensors and the signal chain	Video at 6:20
05	Mobile companion	Video at 9:06
06	Signal tools	Video at 10:51
07	Android apps	Video at 12:36
08	Change the vehicle logic	Video at 13:30
09	Monitor and share	Video at 15:12
10	AI agent via MCP	Video at 15:52
11	Ideas and resources	Video at 16:18
★	Quick reference	

01 Your vehicle: the blueprint

▶ Video at 0:34

A **blueprint** is the wiring diagram of a complete vehicle: 27 nodes grouped into zones. Each box is a node (a virtual ECU, sensor, bus, gateway or operating system); each line is a real connection. All nodes are deployed together as one device, **Hackathon**.



Zone	Nodes	Bus / output
Body	HVAC Hardkey, TirePressure Sensor, SeatBelt Sensor → Climate ECU, BCM ECU, SeatBelt ECU	BCM CAN → BCM Gateway
Powertrain	Drive Controls, Battery Sensor → VCU, BMS ECU	PWT CAN → PWT Gateway
Driver monitoring	DMS CAM Mock → DMS vECU	Attention / fatigue to VSS
Telematics	TCU Gateway, TCU Switch, TCU-NAD	Link to the companion app
Cockpit	IVI - Android, Cluster - AGL, IVI Gateway, Surround CAM Mock, Voice Mock, USB Devices	Displays
Central Broker	KUKSA databroker	VSS — single source of truth for every signal

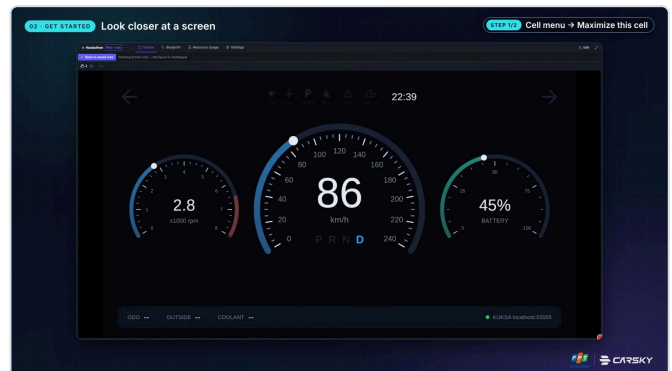
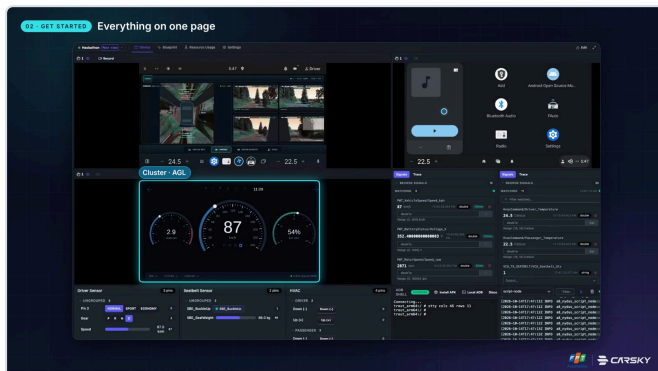
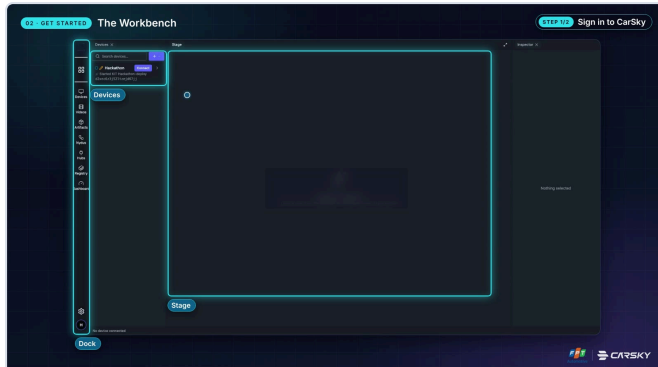
Tip

- Remember the path: **sensor → ECU → CAN bus → gateway → VSS → screens**. Every chapter follows it.
- Blueprint tab: scroll to zoom, drag the background to pan, **Fit View** to reset.

02 Get started

▶ Video at 2:38

The **Workbench** has the dock (far left), your devices, the stage (centre) and the inspector (right). **Focus Screen** puts the whole car on one page.



Try it

1. Sign in at **hackathon.carsky.io**.
2. Find your device in **Devices** and press **Connect**. Its widgets appear and a screen opens on the stage.
3. Press **Focus Screen**: IVI and passenger screen at the top; cluster and the PWT / BCM CAN signal panels in the middle; sensor panels, ADB shell and script logs at the bottom.
4. Tabs: **Device** (this page) · **Blueprint** (running deployment) · **Resource Usage** · **Settings**.

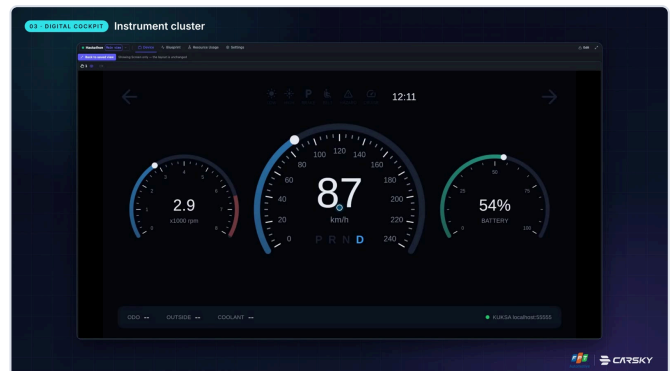
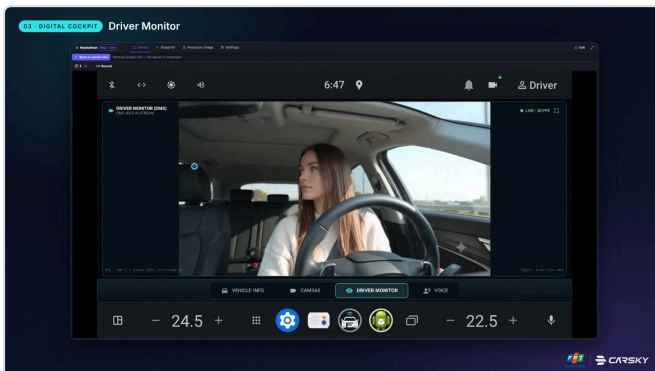
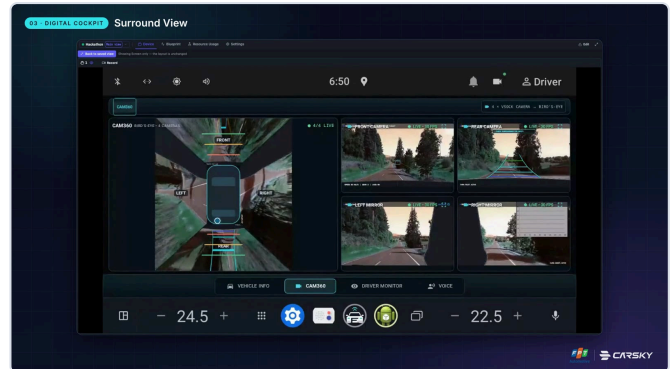
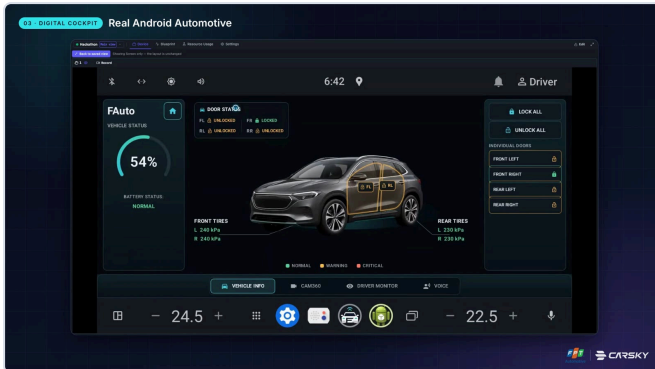
Tip

- Cell menu → **Maximize this cell** to see one screen full size; **Back to saved view** returns. Only use Maximize, Back to saved view and Toggle Fullscreen — the other items change the layout for everyone.
- Focus Screen opens a new window. If nothing happens, allow pop-ups for CarSky.

03 The digital cockpit

▶ Video at 4:10

A real **Android Automotive** build (arm64) runs the head unit and its **FAuto** app; the instrument cluster runs **Automotive Grade Linux**. Click to tap, type with your keyboard.



Screen	What it shows
IVI · VEHICLE INFO (FAuto)	Battery, tire pressures and every door — live from vehicle signals
IVI · CAM360 / Surround View	Front, rear and both mirrors, replayed by mock nodes. Record on a screen cell captures the feeds
IVI · DRIVER MONITOR	In-cabin camera; the DMS ECU publishes attention and fatigue to VSS (face box green → red)
IVI · VOICE	Browser microphone, or an audio file from the Voice Mock node. Tap the mic to listen
Passenger screen	Second Android display with its own apps
AGL cluster	Speed, gear, motor RPM and battery from the KUKSA broker

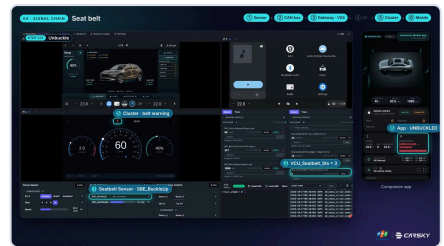
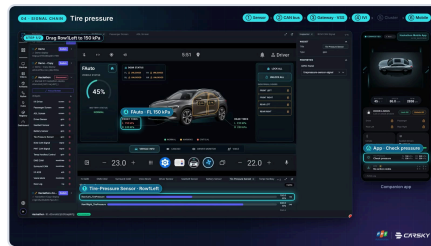
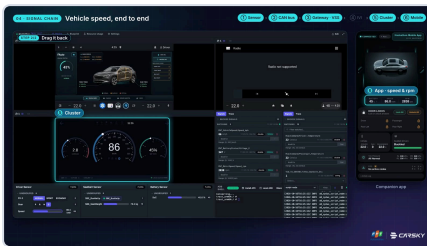
Tip

- Build vision features on recorded camera footage — no car needed.

04 Sensors and the signal chain

▶ Video at 6:20

The panels at the bottom of Focus Screen are **GPIO inputs** wired into the blueprint. Change one and follow it through ① **Sensor** → ② **CAN bus** → ③ **Gateway · VSS** → ④ **IVI** → ⑤ **Cluster** → ⑥ **Mobile**.



Do this	Watch here
Driver Sensor: set Speed	PWT CAN PWT_VehicleSpeed → cluster → app speed
Driver Sensor: P / R / N / D, NORMAL · SPORT · ECONOMY	Gear and drive mode on PWT CAN → cluster
Battery Sensor: SoC to 20 %	Battery frame on CAN → IVI shows LOW → cluster and app follow
IVI climate bar: tap +	BCM CAN HvacCommand/Driver_Temperature → Climate ECU → app
HVAC hard keys: Up (driver)	Climate ECU raises the temperature → IVI and app
FAuto: Lock All / Unlock All	BCM reports each door on CAN → IVI and app show 4 locked doors
TirePressure Sensor: Row1Left to 150 kPa	TPMS frame → FAuto FRONT TIRES L 150 kPa → app "Check pressure"
At 60 km/h, switch SBE_BuckleUp off	VCU_Seatbelt_Sts = 3 → cluster belt warning → app UNBUCKLED

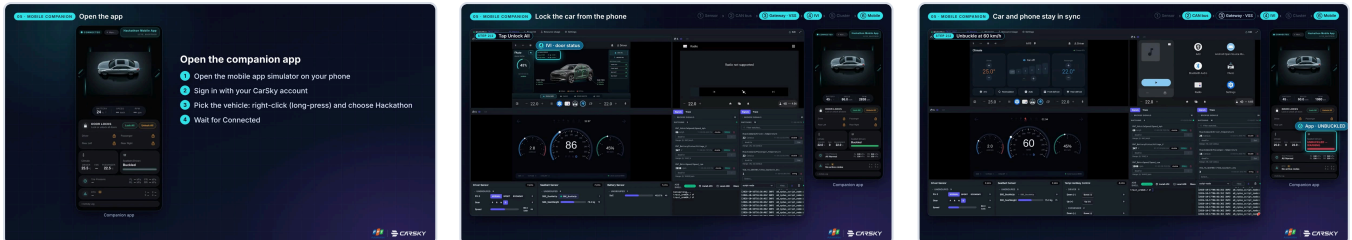
Tip

- The seat belt ECU runs real production C code.
- Put values back when you finish (NORMAL · D · 86 km/h · SoC 45 % · belt buckled · doors unlocked).

05 Mobile companion

▶ Video at 9:06

The companion app talks to the car through the telematics unit (TCU). Commands go phone → car; status comes back car → phone.



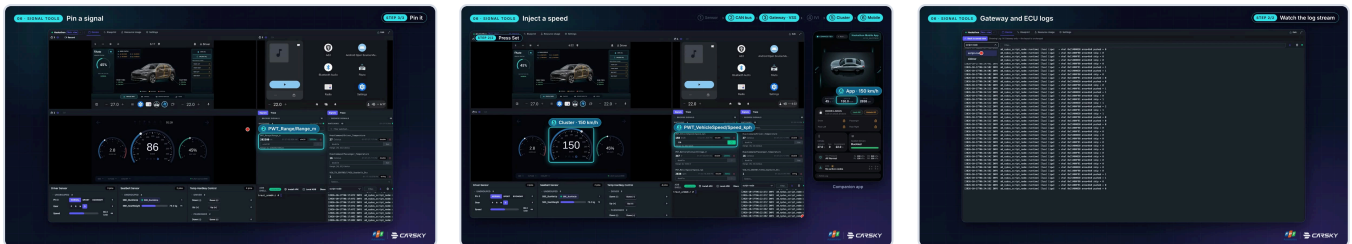
Try it

1. Open the mobile app simulator **mobile.live.carsky.io** and sign in with your CarSky account.
2. Right-click (long-press on a phone) and choose your vehicle. Wait for **CONNECTED**.
3. The live strip shows battery, speed and motor RPM — the same values as the cluster.
4. **Lock All**: phone → MQTT → TCU → SOME/IP → gateway → CAN; the head unit confirms the doors are locked.
5. Climate and seat-belt cards follow the car in real time.
6. **Activity Log**: every message the app receives, e.g. attention and fatigue from the DMS ECU.

06 Signal tools

▶ Video at 10:51

Browse, pin, inject and follow the logs — everything you need to debug a signal end to end.



Tool	Use it to
Browse Signals	List every signal from the DBC file of the bus; search by name
Pin (Watching)	Keep chosen signals with live value and timestamp, e.g. <code>PWT_Range/Range_m</code>
Set	Type a value and press Set: the frame goes on the bus as if an ECU sent it, e.g. <code>PWT_VehicleSpeed/Speed_kph = 150</code> → cluster and app 150 km/h
Gateway and ECU logs	Choose a Luau script node in the log widget; Maximize the cell to follow each signal

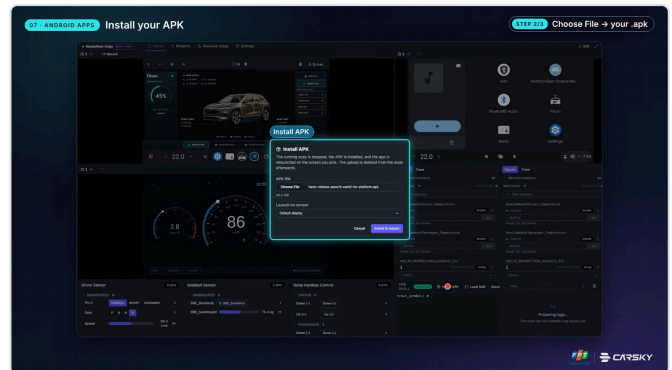
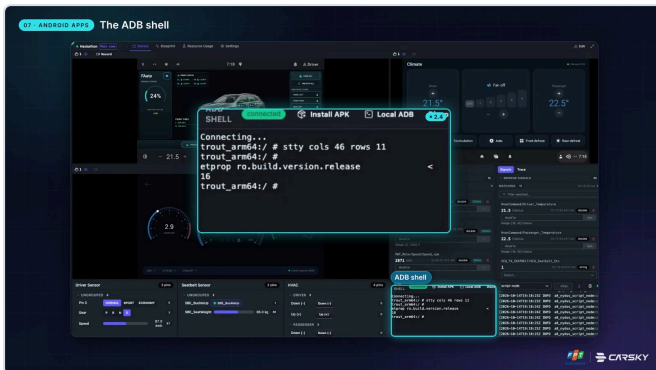
Tip

- Inject edge cases you cannot easily produce with the sensor panels.

07 Android apps

▶ Video at 12:36

The head unit is a real Android Automotive system: use its shell and install your own app.



Try it

1. **ADB shell:** a terminal on the head unit, e.g. `getprop ro.build.version.release`.
2. **Install APK** → **Choose File** → your **arm64** .apk → pick the launch screen → **Install & restart**.
3. The head unit installs the app and relaunches it on the screen you picked.

Tip

- Build for **arm64**. Your app reads vehicle data through the Android Car API, backed by the same VSS signals.

08 Change the vehicle logic

▶ Video at 13:30

Change the car itself: edit a script, extend the blueprint, upload new data files and add widgets.
Do this on your own copy of the device.



Try it

1. **Edit a Luau script:** Blueprint → BCM ECU → open its script, change one line, Save → Redeploy, move the tire slider.
2. **Add a node:** add a Script Node from the palette, wire it to the KUKSA broker, deploy, then add a log widget for it.
3. **New artifact version:** Dock → Artifacts → open a DBC (or VSS, OS image) → **Add Version**, point a node to it and redeploy.
4. **Add a widget:** Settings → select a widget → bind it to a part (e.g. BCM CAN Signal) → Save. It appears in Focus Screen.

Example `print("Tire RowLeft = "..p) → print("TPMS FL: "..p)`

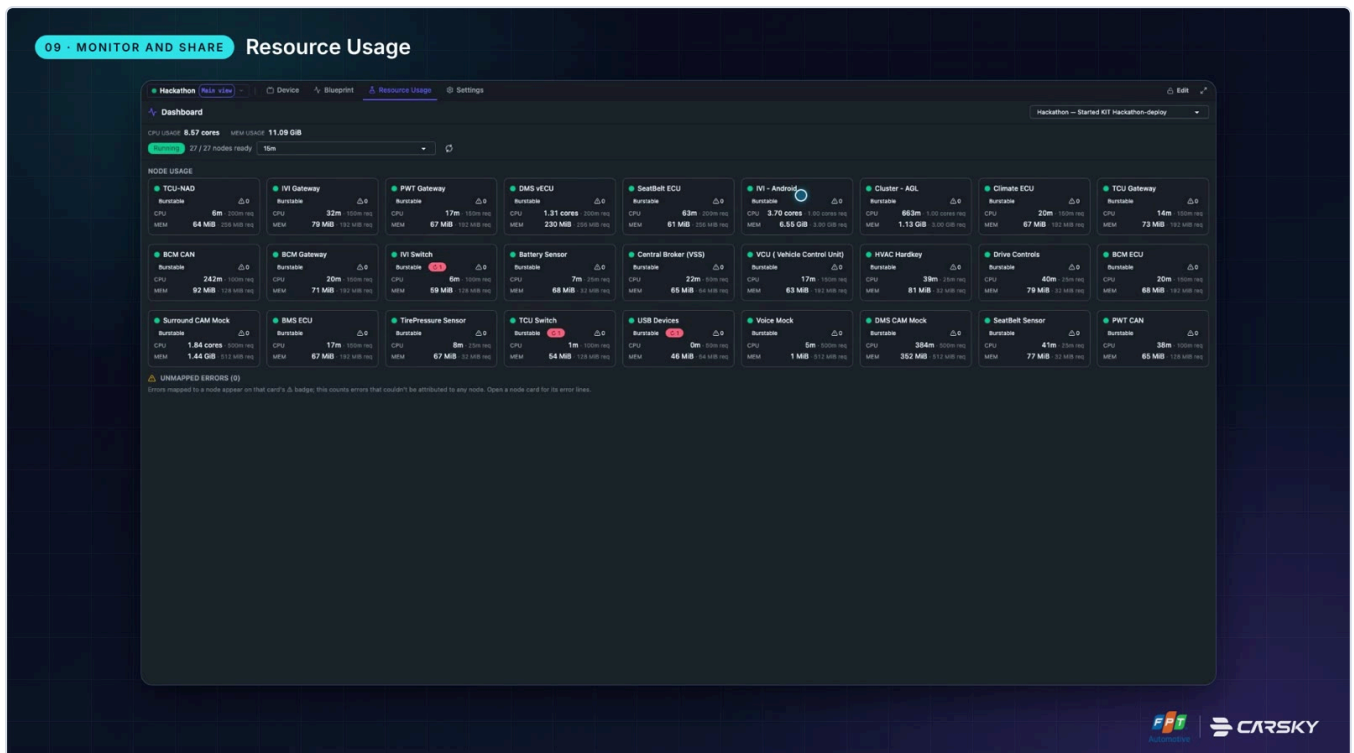
Tip

- Edit → redeploy → watch the result in seconds. That loop is the heart of building on CarSky.

09 Monitor and share

▶ Video at 15:12

Resource Usage shows CPU and memory for every node, with a warning badge on any node with errors. Pick a time range (e.g. 15 m) and check each node card for CPU, memory, errors and restarts.



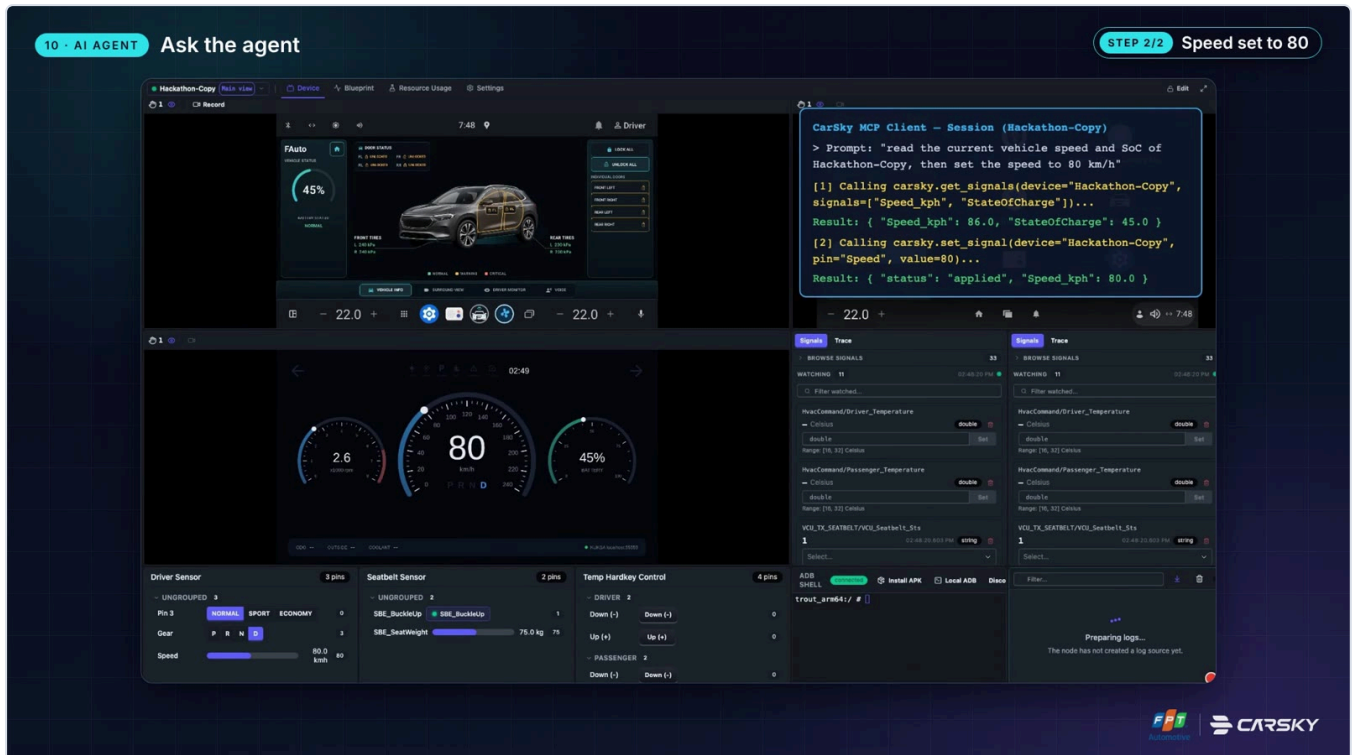
Tip

- Measure what your feature costs before you ship it.

10 AI agent via MCP

▶ Video at 15:52

CarSky exposes an **MCP server**, so AI coding agents (Kiro, Claude Code and others) can work with the same vehicle you do.



Try it

1. Create an API key in **Settings**.
2. Connect your agent to the CarSky MCP server.
3. Ask, for example: *"read the current vehicle speed and SoC, then set the speed to 80 km/h"*. The agent calls `carsky.get_signals` and `carsky.set_signal`, and the cluster follows.

Tip

- Keep your API key private. Never paste it into shared chats, screenshots or recordings.

11 Ideas and resources

▶ Video at 16:18

Everything you need is already in the Starter Kit.

Resource	Link
CarSky for the hackathon	hackathon.carsky.io
Mobile app simulator	mobile.live.carsky.io
Guideline	guide.hackathon.carsky.io
Support e-mail	support.carsky@fpt.com

1

Fatigue alert combining driver monitoring with vehicle speed

2

Eco-driving coach on the AGL cluster

3

Remote climate pre-conditioning from the mobile app

4

A brand-new ECU with its own CAN messages (DBC + Luau script node)

Tip

- Images and apps must target **arm64**.
- All team data is deleted after the event.

Quick reference

CarSky for the hackathon	hackathon.carsky.io
Mobile app simulator	mobile.live.carsky.io
Guideline	guide.hackathon.carsky.io (Luau, DBC & VSS, APK & AGL, MCP)
Help	support.carsky@fpt.com
First steps	CarSky → Device → Connect → Focus Screen
Baseline values	NORMAL · D · 86 km/h · SoC 45 % · seat belt buckled · doors unlocked · tires 240 / 230 kPa
Signal chain	Sensor (GPIO) → ECU → CAN → Gateway → KUKSA / VSS → IVI · cluster · mobile
Protocols	CAN (DBC), GPIO, Ethernet, MQTT, SOME/IP, KUKSA / VSS, Android VHAL
Guest OS	Android Automotive (AAOS), Automotive Grade Linux (AGL) — arm64 on AWS Graviton
AI agents	MCP tools let Kiro, Claude Code and others deploy, test and debug for you

Term	Meaning
ECU	Electronic Control Unit
CAN	Controller Area Network — in-vehicle bus
DBC	CAN database file (messages & signals)
VSS	COVESA Vehicle Signal Specification
KUKSA	Open-source VSS databroker
IVI	In-Vehicle Infotainment
AGL	Automotive Grade Linux
DMS	Driver Monitoring System
TCU	Telematics Control Unit
TPMS	Tire Pressure Monitoring System
Luau	Scripting language of the gateway / ECU nodes
MCP	Model Context Protocol — tools for AI agents