

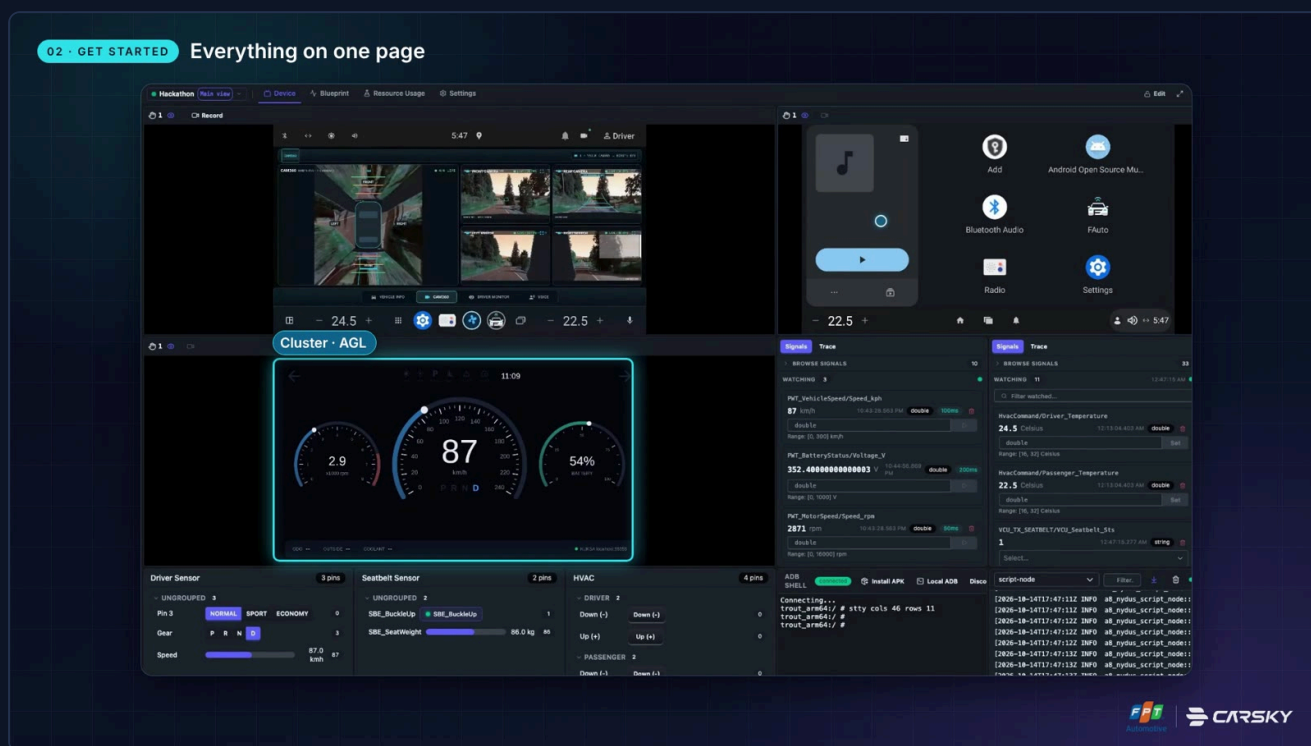


ハッカソン スターターキット

参加者ガイド

ハードウェアを待たずに、クラウドで車載ソフトウェアを開発・テスト。

このガイドは動画の章立てに沿っています。動画と一緒に読み、開発中もいつでも参照してください。



動画：CarSky Hackathon Starter Kit — Guide v2（16分55秒・全12章）

FPT Automotive・AWS Hackathon Tokyo 2026

はじめる前に

1. チームアカウントを取得。QRコードを読み取る（または support.carsky@fpt.com へメール）。チーム名・会社名・担当者名・担当者メールを記入してください。1チーム1アカウントです。
2. **Chrome** または **Edge** をノートPCで使用。ウィンドウは 1920×1080 以上を推奨。
3. ハッカソン用 **CarSky** (hackathon.carsky.io) を開き、受け取ったアカウントでサインイン。
4. **CarSky** → **Device** → **Connect** → **Focus Screen**。自分のデバイス（動画では *Hackathon*）を探して **Connect**、続けて **Focus Screen** を押します。
5. **モバイルアプリシミュレーター**：スマートフォンで mobile.live.carsky.io を開いてサインイン。右クリック（スマートフォンでは長押し）で車両を選び、**Connected** になるまで待ちます。
6. **詳細ガイドライン**：guide.hackathon.carsky.io（Luau、DBC と VSS、APK と AGL、MCP）。



読み取ってチームアカウントを申請
support.carsky@fpt.com

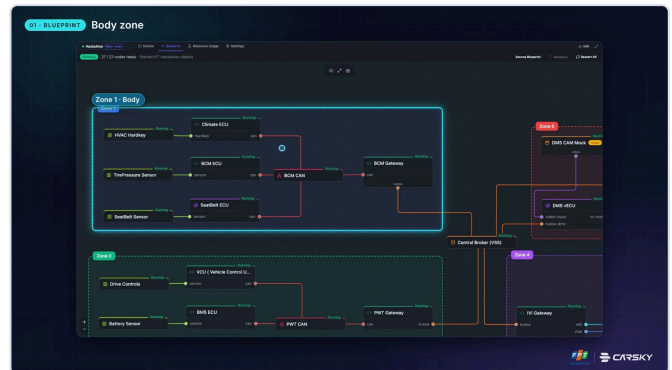
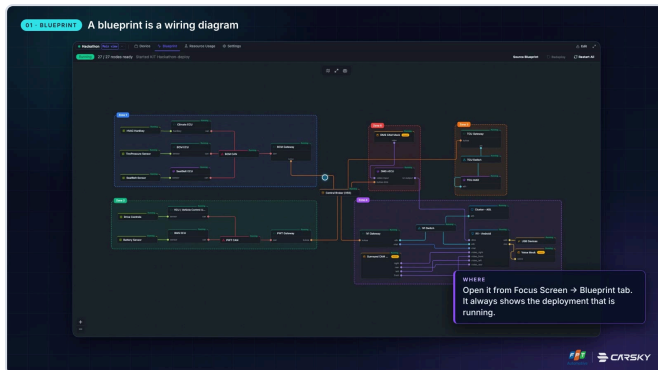
目次

01	車両：ブループリント	動画の位置 0:34
02	使い始める	動画の位置 2:38
03	デジタルコックピット	動画の位置 4:10
04	センサーとシグナルチェーン	動画の位置 6:20
05	モバイルコンパニオン	動画の位置 9:06
06	シグナルツール	動画の位置 10:51
07	Android アプリ	動画の位置 12:36
08	車両ロジックを変える	動画の位置 13:30
09	モニタリングと共有	動画の位置 15:12
10	MCP 経由の AI エージェント	動画の位置 15:52
11	アイデアとリソース	動画の位置 16:18
★	クイックリファレンス	

01 車両：ブループリント

▶ 動画の位置 0:34

ブループリントは車両全体の配線図で、27個のノードがゾーンごとにまとめられています。各ボックスはノード（仮想ECU、センサー、バス、ゲートウェイ、OS）、各線は実際の接続です。全ノードはまとめて1台のデバイス **Hackathon** としてデプロイされています。



ゾーン	ノード	バス／出力
ボディ	HVAC Hardkey、TirePressure Sensor、SeatBelt Sensor → Climate ECU、BCM ECU、SeatBelt ECU	BCM CAN → BCM Gateway
パワートレイン	Drive Controls、Battery Sensor → VCU、BMS ECU	PWT CAN → PWT Gateway
ドライバーモニタリング	DMS CAM Mock → DMS vECU	注意度・疲労度を VSS へ
テレマティクス	TCU Gateway、TCU Switch、TCU-NAD	コンパニオンアプリとの接続
コックピット	IVI - Android、Cluster - AGL、IVI Gateway、Surround CAM Mock、Voice Mock、USB Devices	各ディスプレイ
Central Broker	KUKSA databroker	VSS — 全信号の唯一の情報源

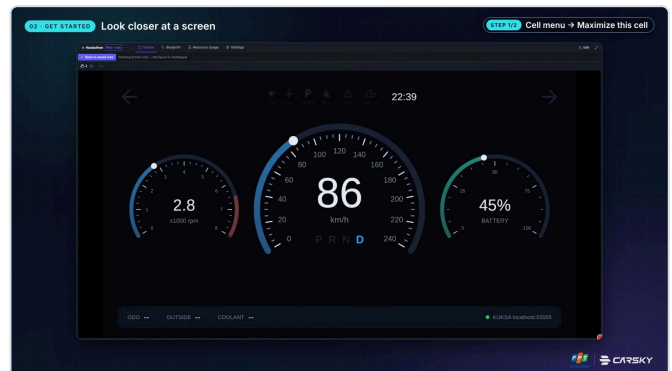
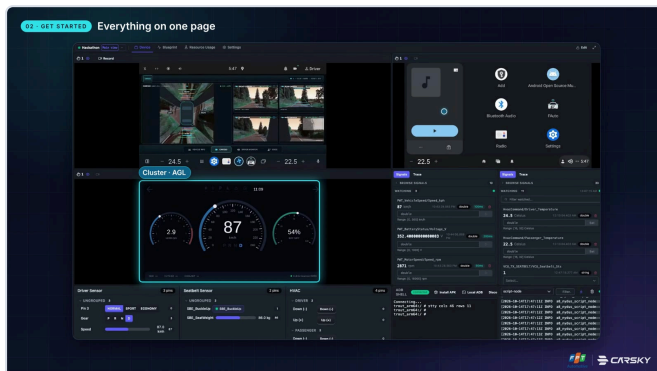
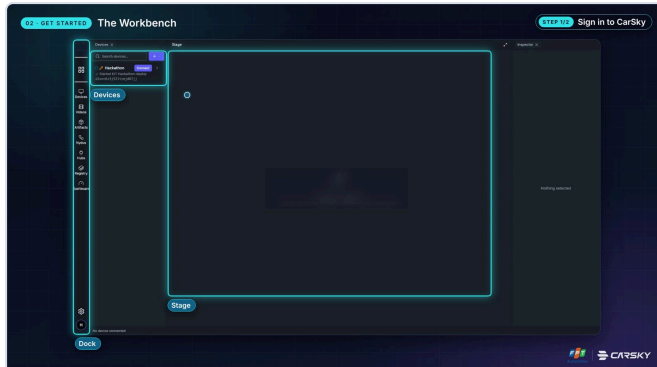
ヒント

- 経路を覚えておきましょう：**センサー → ECU → CANバス → ゲートウェイ → VSS → 画面**。すべての章でこの経路をたどります。
- Blueprint タブ：スクロールでズーム、背景ドラッグで移動、**Fit View** で全体表示。

02 使い始める

▶ 動画の位置 2:38

Workbench は、左端のドック、デバイス一覧、中央のステージ、右側のインスペクターで構成されます。**Focus Screen** では車両全体を1画面で確認できます。



やってみよう

1. hackathon.carsky.io にサインイン。
2. **Devices** で自分のデバイスを探して **Connect** を押すと、ウィジェットが表示され、ステージに画面が開きます。
3. **Focus Screen** を押す：上段に IVI と助手席画面、中段にメーターと PWT/BCM の CAN 信号パネル、下段にセンサーパネル、ADB シェル、スクリプトログ。
4. タブ： **Device** (このページ)・ **Blueprint** (稼働中のデプロイ)・ **Resource Usage**・ **Settings**。

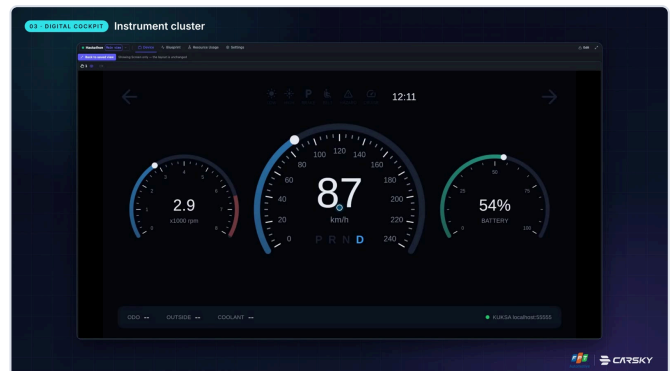
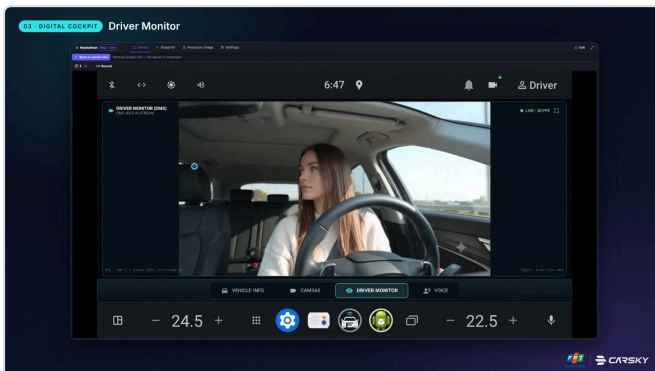
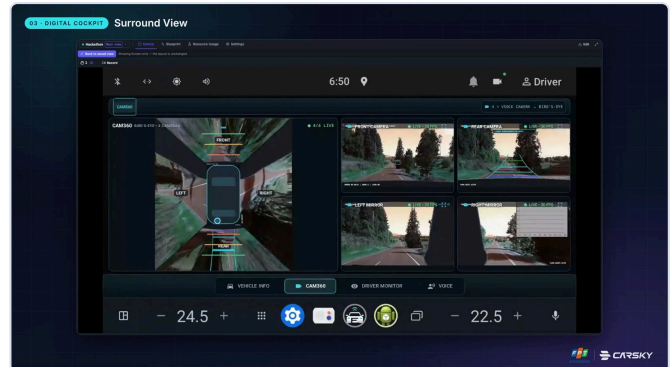
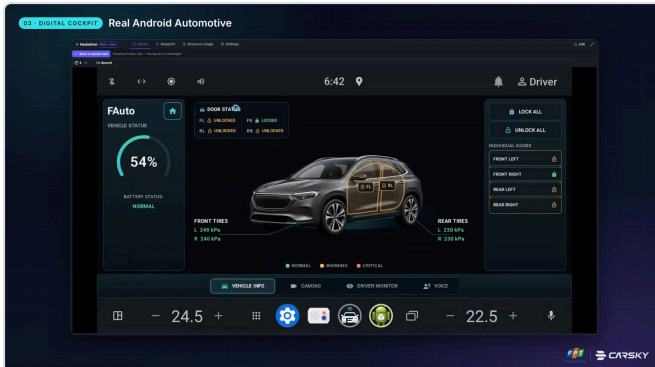
ヒント

- セルメニュー → 「**Maximize this cell**」で1画面を全画面表示、「**Back to saved view**」で戻ります。使うのは Maximize、Back to saved view、Toggle Fullscreen だけにしてください。他の項目は全員のレイアウトを変更します。
- Focus Screen は新しいウィンドウで開きます。開かない場合は CarSky のポップアップを許可してください。

03 デジタルコックピット

▶ 動画の位置 4:10

ヘッドユニットと **FAuto** アプリは実際の **Android Automotive** (arm64)、メーターは **Automotive Grade Linux** で動作しています。クリックでタップ、キーボードで文字入力できます。



画面	表示内容
IVI・VEHICLE INFO (FAuto)	バッテリー、タイヤ空気圧、各ドア（車両信号からリアルタイム）
IVI・CAM360／Surround View	前・後・左右ミラーの映像（モックノードが再生）。画面セルの Record で録画
IVI・DRIVER MONITOR	車内カメラ。DMS ECU が注意度・疲労度を VSS に配信（顔の枠が緑 → 赤）
IVI・VOICE	ブラウザのマイク、または Voice Mock ノードの音声ファイル。マイクをタップして聞き取り
助手席画面	独自アプリを持つ2台目の Android 画面
AGL メーター	速度・ギア・モーター回転数・バッテリー（KUKSA ブローカーから取得）

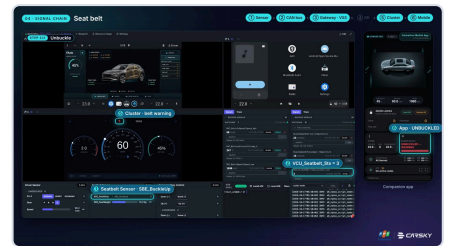
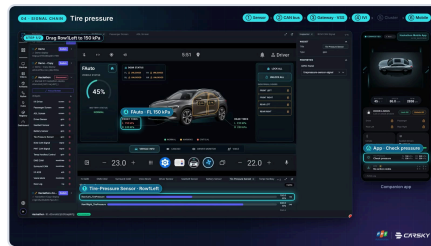
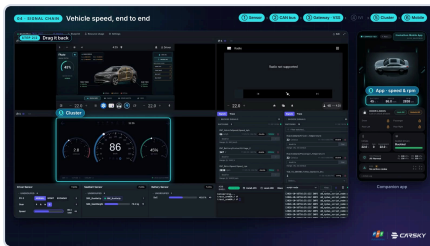
ヒント

- 録画済みカメラ映像で画像認識機能を開発できます。実車は不要です。

04 センサーとシグナルチェーン

▶ 動画の位置 6:20

Focus Screen 下段のパネルは、ブループリントに接続された **GPIO 入力** です。値を変えて ① **センサー** → ② **CANバス** → ③ **ゲートウェイ・VSS** → ④ **IVI** → ⑤ **メーター** → ⑥ **モバイル** の流れを追いましょ



操作	確認する場所
Driver Sensor : Speed を設定	PWT CAN PWT_VehicleSpeed → メーター → アプリの速度
Driver Sensor : P/R/N/D、NORMAL・SPORT・ECONOMY	ギアとドライブモードが PWT CAN → メーター
Battery Sensor : SoC を 20 % に	CAN のバッテリーフレーム → IVI に LOW → メーターとアプリ
IVI のエアコンバー : + をタップ	BCM CAN HvacCommand/Driver_Temperature → Climate ECU → アプリ
HVAC ハードキー : 運転席 Up	Climate ECU が温度を上げる → IVI とアプリ
FAuto : Lock All/Unlock All	BCM が各ドアを CAN で報告 → IVI とアプリに4ドアのロック
TirePressure Sensor : Row1Left を 150 kPa に	TPMS フレーム → FAuto の FRONT TIRES L 150 kPa → アプリ「Check pressure」
60 km/h で SBE_BuckleUp をオフ	VCU_Seatbelt_Sts = 3 → メーターにベルト警告 → アプリ UNBUCKLED

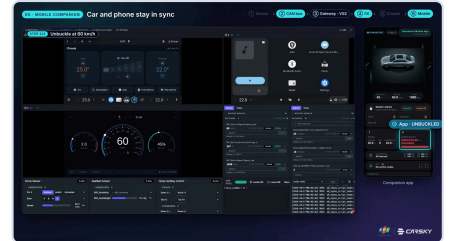
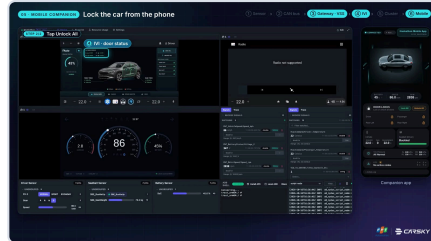
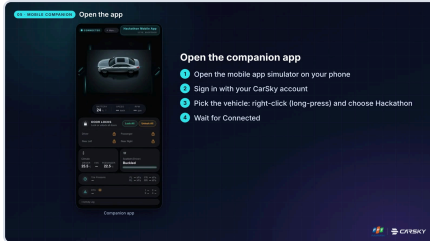
ヒント

- シートベルト ECU は実際の量産 C コードで動いています。
- 終わったら値を戻してください (NORMAL・D・86 km/h・SoC 45 %・ベルト着用・ドア解錠)。

05 モバイルコンパニオン

▶ 動画の位置 9:06

コンパニオンアプリはテレマティクスユニット（TCU）を通じて車両と通信します。コマンドはスマートフォン → 車両、状態は車両 → スマートフォンへ戻ります。



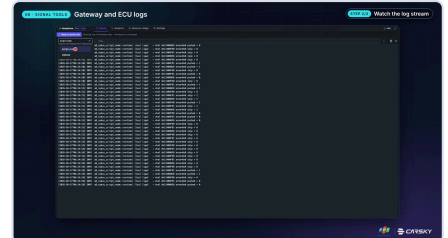
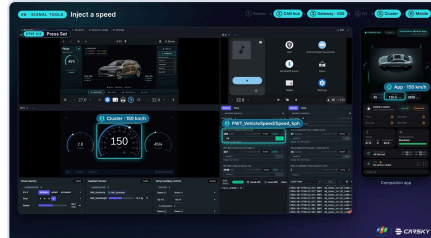
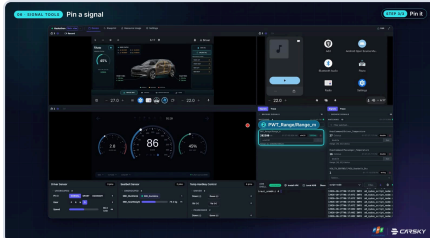
やってみよう

1. モバイルアプリシミュレーター **mobile.live.carsky.io** を開き、CarSky アカウントでサインイン。
2. 右クリック（スマートフォンでは長押し）で車両を選び、**CONNECTED** になるまで待ちます。
3. 上部のライブ表示：バッテリー、速度、モーター回転数（メーターと同じ値）。
4. **Lock All**：スマートフォン → MQTT → TCU → SOME/IP → ゲートウェイ → CAN。ヘッドユニットがドアのロックを確認します。
5. エアコンとシートベルトのカードは車両に合わせてリアルタイムに更新されます。
6. **Activity Log**：アプリが受信したすべてのメッセージ（例：DMS ECU からの注意度・疲労度）。

06 シグナルツール

▶ 動画の位置 10:51

一覧表示、ピン留め、値の注入、ログの追跡。信号をエンドツーエンドでデバッグするのに必要な道具がそろっています。



ツール	用途
Browse Signals	そのバスの DBC ファイルに定義された全信号を一覧表示し、名前で検索
ピン留め (Watching)	選んだ信号をライブ値とタイムスタンプ付きで表示（例：PWT_Range/Range_m）
Set	値を入力して Set：ECU が送ったのと同じフレームがバスに流れます（例：PWT_VehicleSpeed/Speed_kph = 150 → メーターとアプリが 150 km/h）
ゲートウェイ・ ECU ログ	ログウィジェットで Luau スクリプトノードを選択。セルを最大化して各信号を追跡

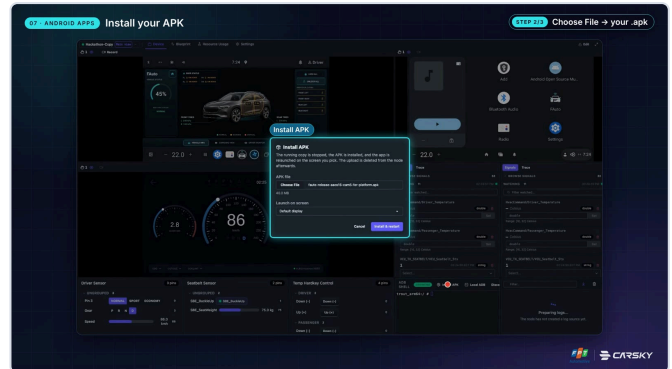
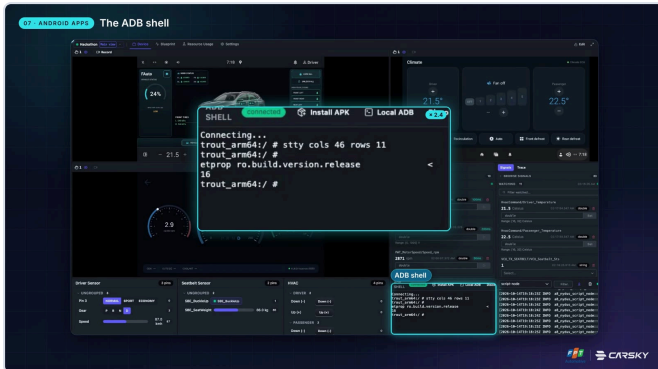
ヒント

- センサーパネルでは作りにくいエッジケースを注入してテストできます。

07 Android アプリ

▶ 動画の位置 12:36

ヘッドユニットは本物の Android Automotive です。シェルを使い、自作アプリをインストールできます。



やってみよう

1. **ADB シェル**：ヘッドユニットのターミナル（例：`getprop ro.build.version.release`）。
2. **Install APK** → **Choose File** → **arm64** の .apk を選択 → 起動する画面を選択 → **Install & restart**。
3. ヘッドユニットがアプリをインストールし、選んだ画面で起動し直します。

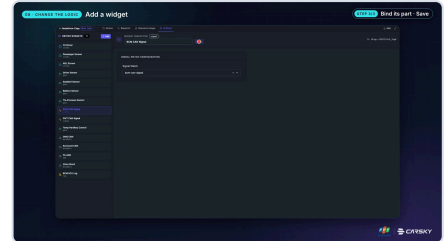
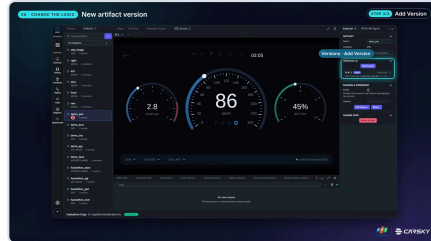
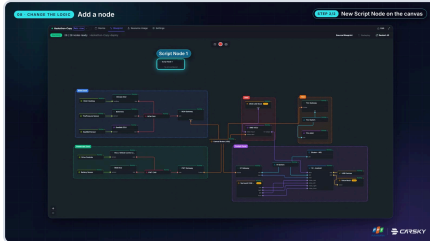
ヒント

- **arm64** 向けにビルドしてください。アプリは Android Car API を通じて、同じ VSS 信号に基づく車両データを読み取れます。

08 車両ロジックを変える

▶ 動画の位置 13:30

クルマそのものを変更します：スクリプト編集、ブループリントの拡張、データファイルのアップロード、ウィジェットの追加。**必ず自分用のデバイスのコピーで行ってください。**



やってみよう

1. **Luau スクリプトを編集**：Blueprint → BCM ECU → スクリプトを開いて1行変更 → Save → Redeploy → タイヤのスライダーを動かす。
2. **ノードを追加**：パレットから Script Node を追加し、KUKSA ブローカーに接続してデプロイ。ログウィジェットも追加します。
3. **アーティファクトの新バージョン**：ドック → Artifacts → DBC（または VSS、OS イメージ）を開く → **Add Version**。ノードに指定して再デプロイ。
4. **ウィジェットを追加**：Settings → ウィジェットを選択 → パーツ（例：BCM CAN Signal）に割り当て → Save。Focus Screen にすぐ表示されます。

例 `print("Tire Row1Left = "..p) → print("TPMS FL: "..p)`

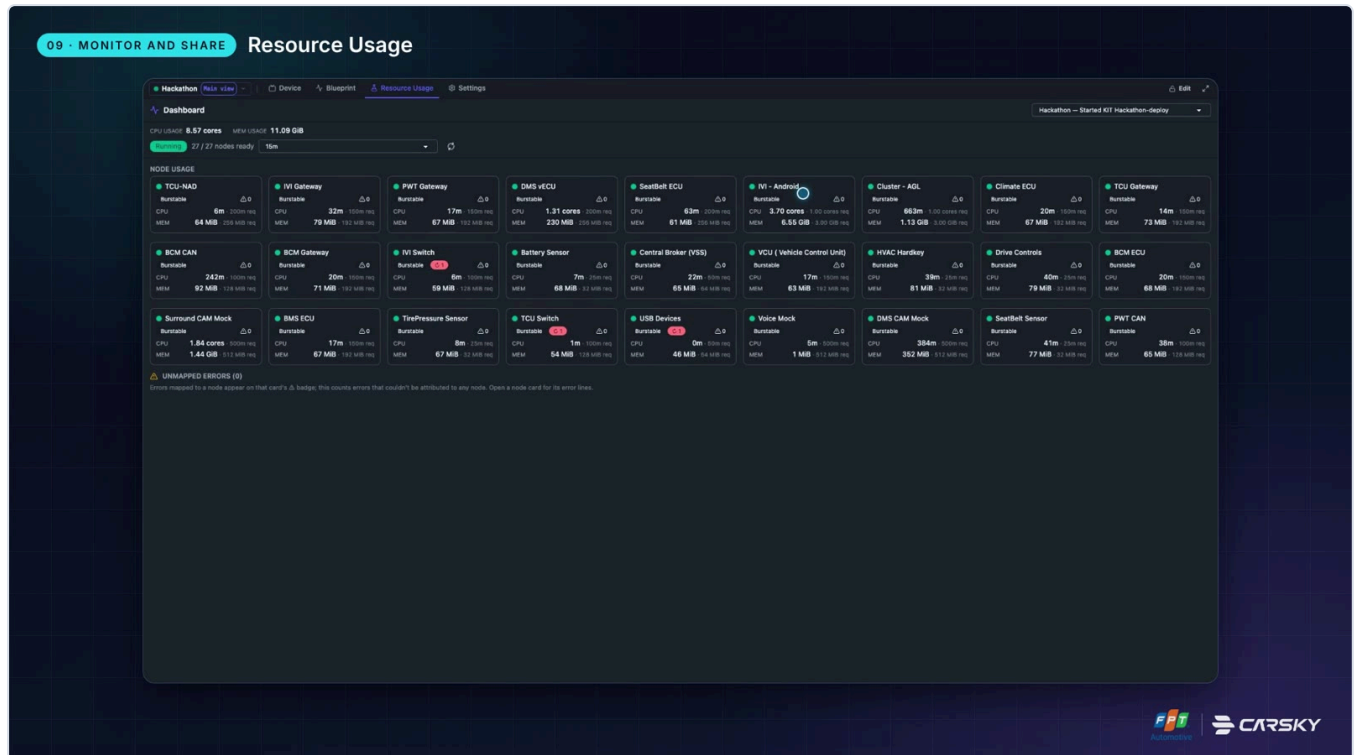
ヒント

- 編集 → 再デプロイ → 数秒で結果を確認。このループが CarSky での開発の中心です。

09 モニタリングと共有

▶ 動画の位置 15:12

Resource Usage ではノードごとの CPU とメモリを表示し、エラーのあるノードには警告バッジが付きます。期間（例：15 m）を選び、各ノードカードで CPU、メモリ、エラー、再起動を確認します。



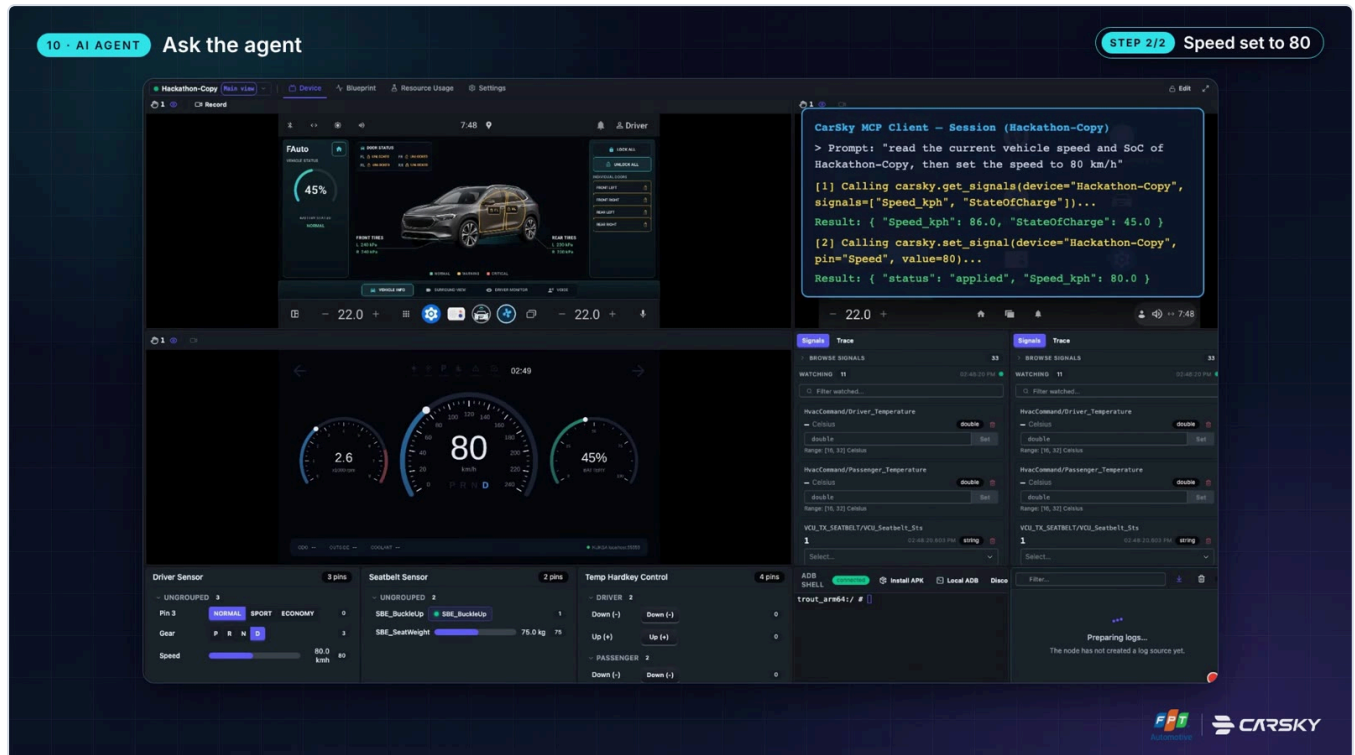
ヒント

- 機能をリリースする前に、そのコストを測りましょう。

10 MCP 経由の AI エージェント

▶ 動画の位置 15:52

CarSky は **MCP サーバー** を公開しているため、AI コーディングエージェント（Kiro、Claude Code など）があなたと同じ車両を操作できます。



やってみよう

1. **Settings** で API キーを作成。
2. エージェントを CarSky の MCP サーバーに接続。
3. 例えば「現在の車速と SoC を読み、速度を 80 km/h に設定して」と依頼します。エージェントが `carsky.get_signals` と `carsky.set_signal` を呼び出し、メーターが追従します。

ヒント

- API キーは公開しないでください。共有チャット、スクリーンショット、録画に貼り付けないこと。

11 アイデアとリソース

▶ 動画の位置 16:18

必要なものはすべてスターターキットにそろっています。

リソース	リンク
ハッカソン用 CarSky	hackathon.carsky.io
モバイルアプリシミュレーター	mobile.live.carsky.io
ガイドライン	guide.hackathon.carsky.io
サポートメール	support.carsky@fpt.com

1

ドライバーモニタリングと車速を組み合わせた疲労アラート

2

AGL メーター上のエコドライブコーチ

3

モバイルアプリからのリモート空調（プレコンディショニング）

4

独自の CAN メッセージを持つ新しい ECU（DBC + Lua スクリプトノード）

ヒント

- イメージとアプリは **arm64** 向けにしてください。
- チームのデータはイベント終了後に削除されます。

クイックリファレンス

ハッカソン用 CarSky	hackathon.carsky.io
モバイルアプリシミュレーター	mobile.live.carsky.io
ガイドライン	guide.hackathon.carsky.io (Luau、DBC と VSS、APK と AGL、MCP)
サポート	support.carsky@fpt.com
最初の手順	CarSky → Device → Connect → Focus Screen
初期値	NORMAL ・ D ・ 86 km/h ・ SoC 45 % ・ シートベルト着用 ・ ドア解錠 ・ タイヤ 240／230 kPa
シグナルチェーン	センサー (GPIO) → ECU → CAN → ゲートウェイ → KUKSA／VSS → IVI ・ メーター ・ モバイル
プロトコル	CAN (DBC)、GPIO、Ethernet、MQTT、SOME/IP、KUKSA／VSS、Android VHAL
ゲスト OS	Android Automotive (AAOS)、Automotive Grade Linux (AGL) — AWS Graviton 上の arm64
AI エージェント	MCP ツールで Kiro、Claude Code などがデプロイ ・ テスト ・ デバッグを代行

用語	意味
ECU	電子制御ユニット
CAN	車載ネットワーク (Controller Area Network)
DBC	CAN データベースファイル (メッセージと信号)
VSS	COVESA Vehicle Signal Specification
KUKSA	オープンソースの VSS データブローカー
IVI	車載インフォテインメント
AGL	Automotive Grade Linux
DMS	ドライバーモニタリングシステム
TCU	テレマティクス制御ユニット
TPMS	タイヤ空気圧監視システム
Luau	ゲートウェイ／ECU ノードのスクリプト言語
MCP	Model Context Protocol — AI エージェント用ツール