# First MEC Hackathon – Exemplary case study

This case study, based on a PoC realized by Vodafone, Amazon Web Services and Saguna, is written to inspire developers to work with the hackathon environment. The PoC was about a use case in automotive called "distracted driver monitoring", and made use of 4G radio, Saguna MEC platform and an AI/ML application.

The intention of this use case is to provide an end-to-end example (not binding for the competition) of how a MEC system as the one from Saguna (available for the hackathon) can serve as an edge computing environment and which other components may round off such an application. Developers are free to choose any other specific use case related to the general topic contained in the Call for Developers document.

Thus, please have a look at the present case study, and at the same time feel free to reuse your past or existing projects that may be related to advanced mobile applications for automotive infotainment services, e.g. by proposing your own Entertainment and VR/AR applications (called “EVA apps”), as in-car mobile solutions using ETSI MEC technologies for passengers.

Enjoy the reading!

## Case study: “Distracted driver monitoring”

### The proof of concept challenge

As part of the broader connected cars trend that is transforming the automotive industry, various accident prevention solutions are being developed. Among these are driver monitoring solutions, which are designed to alert distracted drivers who text, talk on the phone, become drowsy or shift their focus away from the road. These solutions require a camera in the vehicle focused on the driver and the artificial intelligence that track movements and effectively identify when the driver becomes distracted. However, car manufacturers have to deal with a large amount of in-vehicle data that exceeds the ability to process all of that data locally. Installing additional computing power in the vehicle is expensive and may quickly become obsolete during a typical car’s 20-year lifespan. Furthermore, sending the data to the cloud for processing has performance issues due to latency. As a result, current in-vehicle solutions are expensive, hard to upgrade and out of reach for the mass market.

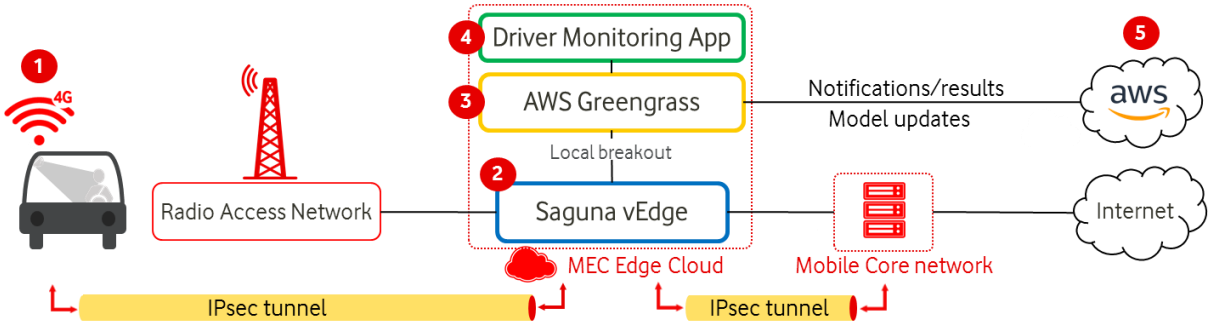
### Overall story

Amazon Web Services, Vodafone and Saguna worked together to explore edge-cloud-computing for smart-camera services, like driver monitoring, and offloading of computing resources from the car. Combining Vodafone’s 4G LTE network, Saguna’s ETSI MEC compliant platform and AWS Greengrass’s machine learning inference capabilities, the idea is that drivers can install a simple camera in their vehicles that watches the driver and relays video frames to the nearest processing point. In the cellular network, Saguna’s MEC platform hosts AWS Greengrass to quickly collect and intelligently process data, and send an alert back to the vehicle if the driver is distracted. This approach can deliver an economical and reliable solution that can enable car manufacturers and suppliers to ensure the safety of cars and their drivers. By operating inside the network yet close to the connected car, this low-latency solution can prevent accidents with real-time alerts while lowering the in-vehicle bill-of-materials, reducing cost through shared resources, continuing to benefit from technology improvements and scaling of the service. This can make new services economical and relevant to the mass market, opening a wide range of possibilities across a broad consumer market (e.g. fleet management, insurance). See a video explanation here: https://youtu.be/oDNCGIur778

## Solution architecture

The proof-of-concept solution targets after-market in-car devices, whose video feeds are streamed to the edge of a 4G LTE network, where Saguna's Multi-access Edge Computing (MEC) solution directs the stream to AWS Greengrass for machine learning analysis.

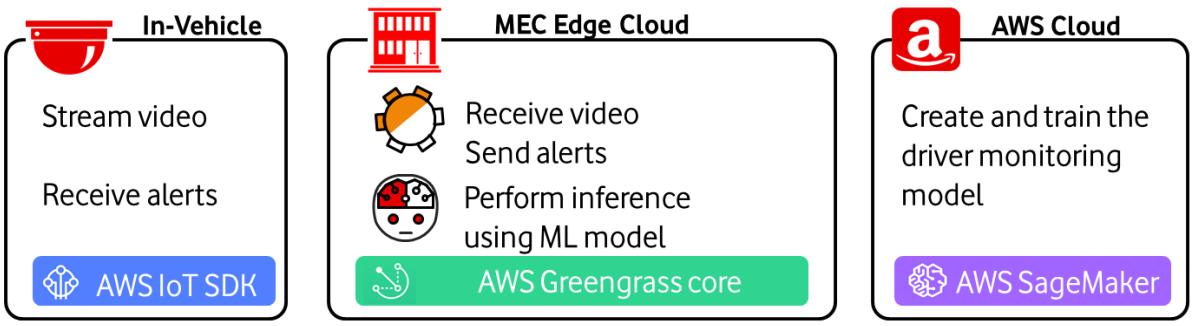
### The network infrastructure side




1. The in-car device (the **UE**) is a Raspberry Pi equipped with Pi Camera. It is connected via cellular radio and continuously streams video frames to the network.
2. Saguna vEdge, the **ETSI MEC platform**, filters traffic according to various traffic rules (e.g. matching the typical 5-tuple flow identifier, matching IMSI/APN) and redirects video streams to the co-located AWS Greengrass instance. It implements a mechanism called "local breakout" terminating the session at the MEC Edge Cloud without involving the mobile core network.
3. AWS Greengrass Core is hosted in the MEC Edge Cloud and the camera device is registered in the same AWS Greengrass Group. This is implemented as a **MEC application** that receives traffic from the radio access network (as per configured traffic rules in the MEC platform) and enables (through AWS Lambda functions) the driver monitoring application on top of it.
4. The driver monitoring application processes video frames using a neural network to detect distracting behaviours (talking and texting on the phone in this proof-of-concept).
5. AWS cloud is used to train the machine learning model and send updated models to the AWS Greengrass instance. It can also receive notifications and results from the edge if they need to be dispatched to other applications (examples might be IT system of an insurance company, fleet management system).

### The developer side

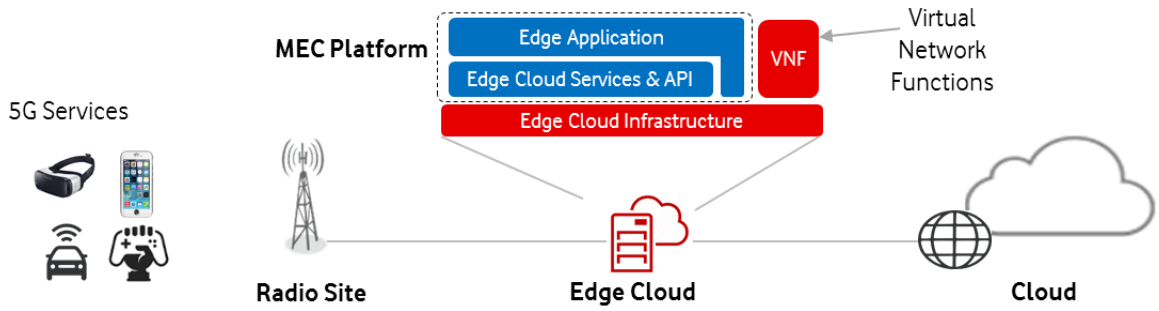
The driver monitoring application does not follow a traditional client-server approach, but a 3-tier model, with the MEC Edge Cloud as an intermediate element (see picture below). The machine learning process is then split into 3 locations: the data source at the in-vehicle device; the inference and fast reply at the network edge close to the device; the training in the cloud, where large computation power is available.



The in-vehicle Raspberry Pi video stream application has been built using the AWS IoT SDK, which establishes a secure communication between the device and the AWS Greengrass instance at the edge. Video streams are sent in MQTT messages between the Rapsberry Pi and AWS Greengrass over the wireless network.
At the edge cloud, the MEC platform is configured with the correct traffic rules to redirect video frames to AWS Greengrass and trigger the driver monitoring application, which uses a convolutional neural network (MobileNet) to detect driver behaviours.
In the cloud, AWS SageMaker trains the machine learning model using a public dataset of images of drivers in different situations. The training phase benefits from the heavy compute power in the cloud and can happen offline. The produced/updated model is sent to the edge, where the fast inference runs continuously without needing an always-on connection to the cloud.

## Working with MEC

Multi-access edge computing brings cloud-computing capabilities into the edge of the telecoms network.




In above use case, an automotive application has been deployed in a three-tiered way using i) a simple device platform (Raspberry Pi), ii) a cloud computing environment at the network edge that is also available in a public cloud (AWS's Lambda function environment) and iii) the public cloud itself (which supports the same Lambda function environment).
Thus, working with MEC was easy for the application developers as they faced:

- A common environment both at the network edge and the public cloud
- A MEC platform that takes care of routing traffic between end user devices and the edge application over a 4G cellular network
- A simple way to deploy the edge application to the MEC platform

Other new, innovative use cases may leverage a number of additional features that will be supported by an ETSI-compliant MEC platform. To recap:

ETSI MEC defines a platform framework that interfaces with MEC applications and other MEC platforms. The MEC platform is responsible for multiple functions, including: receiving DNS records and configuring a DNS proxy/server; hosting MEC Services, such as: Radio Network Information, Location and Bandwidth Manager. Each application instantiated at the edge can use any subset of the totality of functions provided by the MEC platform through simple, modern and well defined REST APIs[1]. To learn and play with those APIs, visit the ETSI live demo: [www.etsi.org/mec-hackathon-1-berlin/mec-live-demo](www.etsi.org/mec-hackathon-1-berlin/mec-live-demo)

[1] The complete specification is available here: [https://forge.etsi.org/](https://forge.etsi.org/)