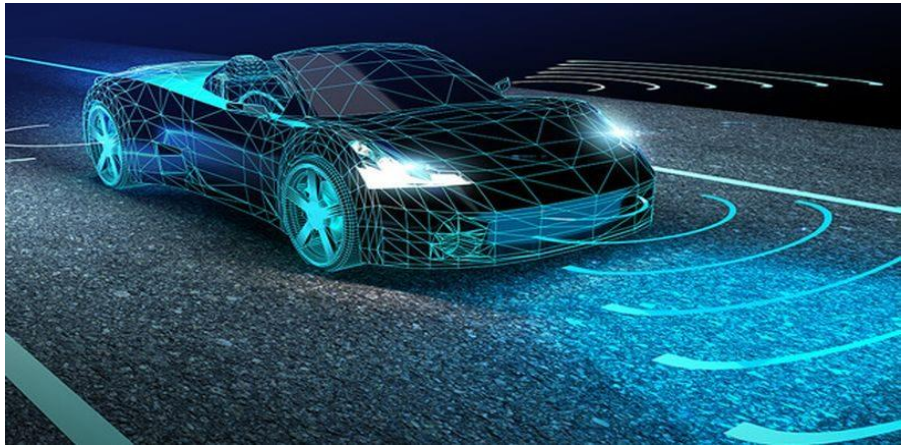
# Project Report

By Jyolsna Joji Isac,
Software Intern (ADAS Team),
ACTEVIA Technology Services Limited



# ADAS

Advanced Driver Assistance Systems (ADAS) refer to a collection of safety and convenience technologies designed to assist drivers in the driving process and improve overall vehicle safety.

These systems use various sensors, cameras, radar, lidar, and software algorithms to detect the vehicle's surroundings, interpret the data, and provide appropriate responses or alerts to the driver.

## Level 0

- No automation

## Level 1

- Emergency braking, collision warning

## Level 2

- Level 1 + ACC, Parking line detection, autonomous emergency braking

## Level 3

- Level 2 + Complete control except under certain traffic and weather conditions. Tells driver when it cannot.

## Level 4

- Level 3 + No assistance from driver but require assistance time to time. Geographical area is limited.

## Level 5

- Level 4 + Complete automation. No control elements like steering wheel or pedal.

# Objective of the Project

- The Project's objective is to develop a sophisticated emulator capable of generating approximately 60% of the essential data required for ADAS applications.
- This emulation will complement real-world testing by offering a practical and efficient tool for validation and optimization.

**Constraints**:-

- Currently Image data is not considered for processing of the input signals.
- SOME/IP is used as the transport mechanism between Emulator and EGO vehicle. Other protocols like DDS can be used but not considered in this project due to dependencies on free open source.
- SOME/IP protocol in this project is currently using UDP and doesn’t support TCP as it was leading to multiple packet exchanges
- The Emulation is designed to run on a host PC. No support in running the Emulation on board (Technically possible but has not been tried)
- Host Machine is an Ubuntu machine. Windows option is not explored
- Customization or Configuration of the SOME/IP protocol is done manually without using any tools.

# Problem Statement

- Testing ADAS technology in real life scenarios is challenging due to:

### Safety Concerns:

- Real-world tests involve inherent risks like emergency maneuvers. Safety for drivers, pedestrians, and other road users is crucial.

### Variability of Conditions:

- Diverse weather, road surfaces, traffic, and lighting conditions require extensive logistical planning and resources for comprehensive testing.

### Cost and Time Intensity:

- Real-life testing is costly and time-consuming, involving vehicle deployment, staffing, and regulatory requirements.

- To overcome these challenges, simulations and data generation offer a safer, more efficient alternative:

### Risk Mitigation:

- Simulations eliminate safety risks, allowing rigorous testing in controlled environments.

### Scenario Replication:

- Simulated scenarios cover diverse and rare situations, ensuring thorough testing.

### Cost Efficiency:

- Simulations reduce expenses associated with real-world tests.

### Data Precision:

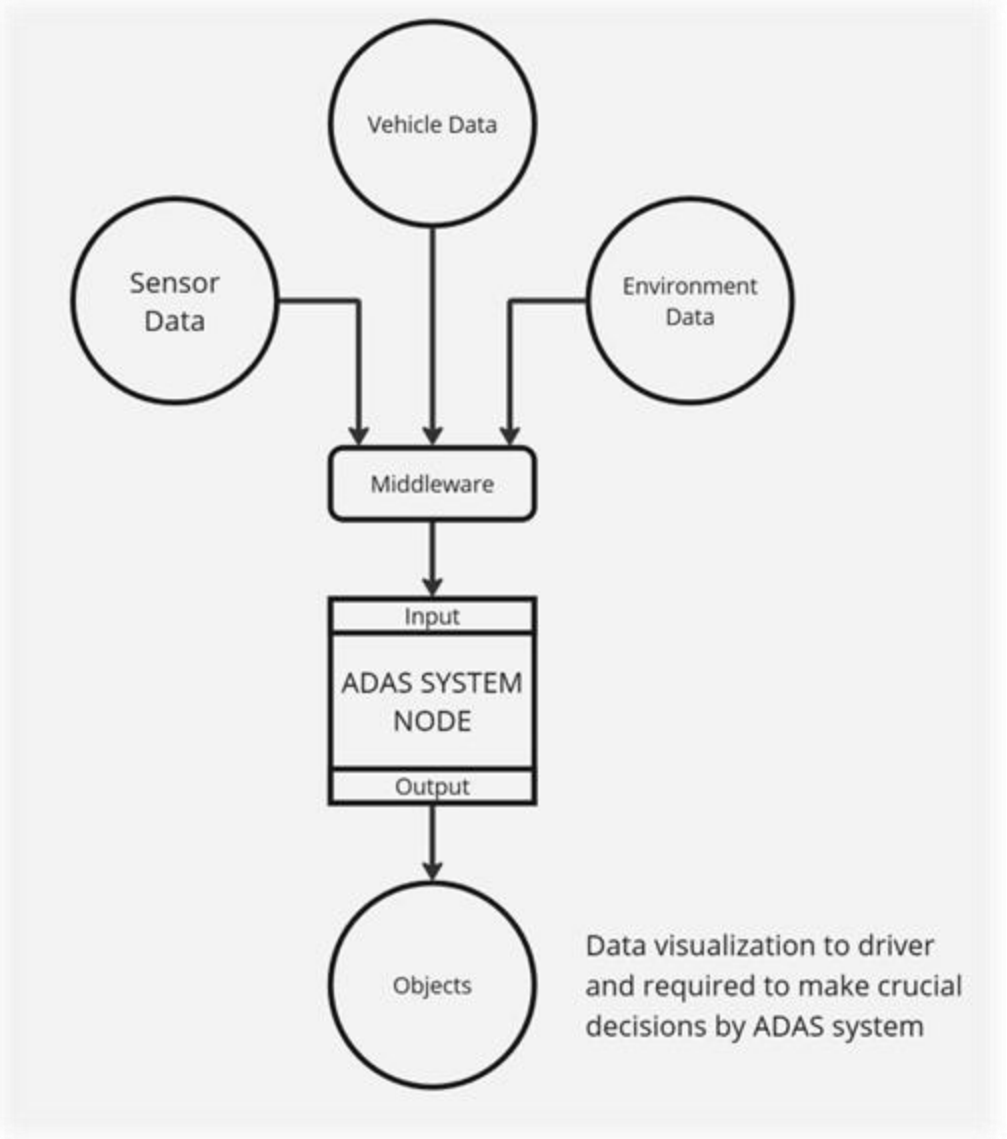
- Controlled environments ensure accurate evaluation of ADAS performance and algorithms.

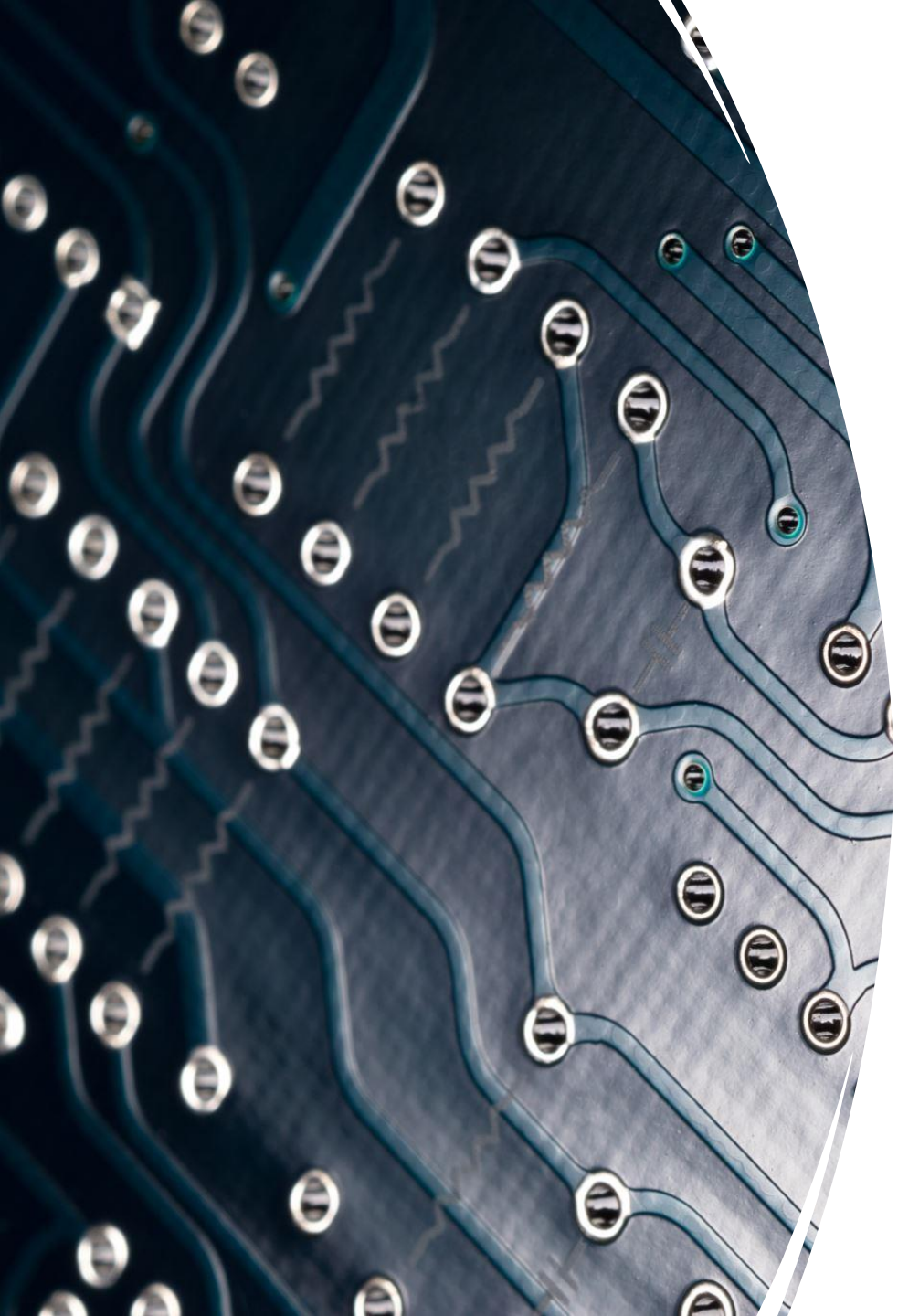
### Flexibility and Scalability:

- Adjusting variables like weather and traffic enables targeted testing and scalability.

# ADAS Stack

- Data - Other that the various sensor data (radar, camera, lidar), it also includes vehicle data (vehicle width, height, sensor position), and environment information (weather details, temperature, date and time).
- The data is sent to the ADAS system through a middleware
- Based on the input data, the system generates objects
- These objects are shown to the driver and used to make crucial decisions





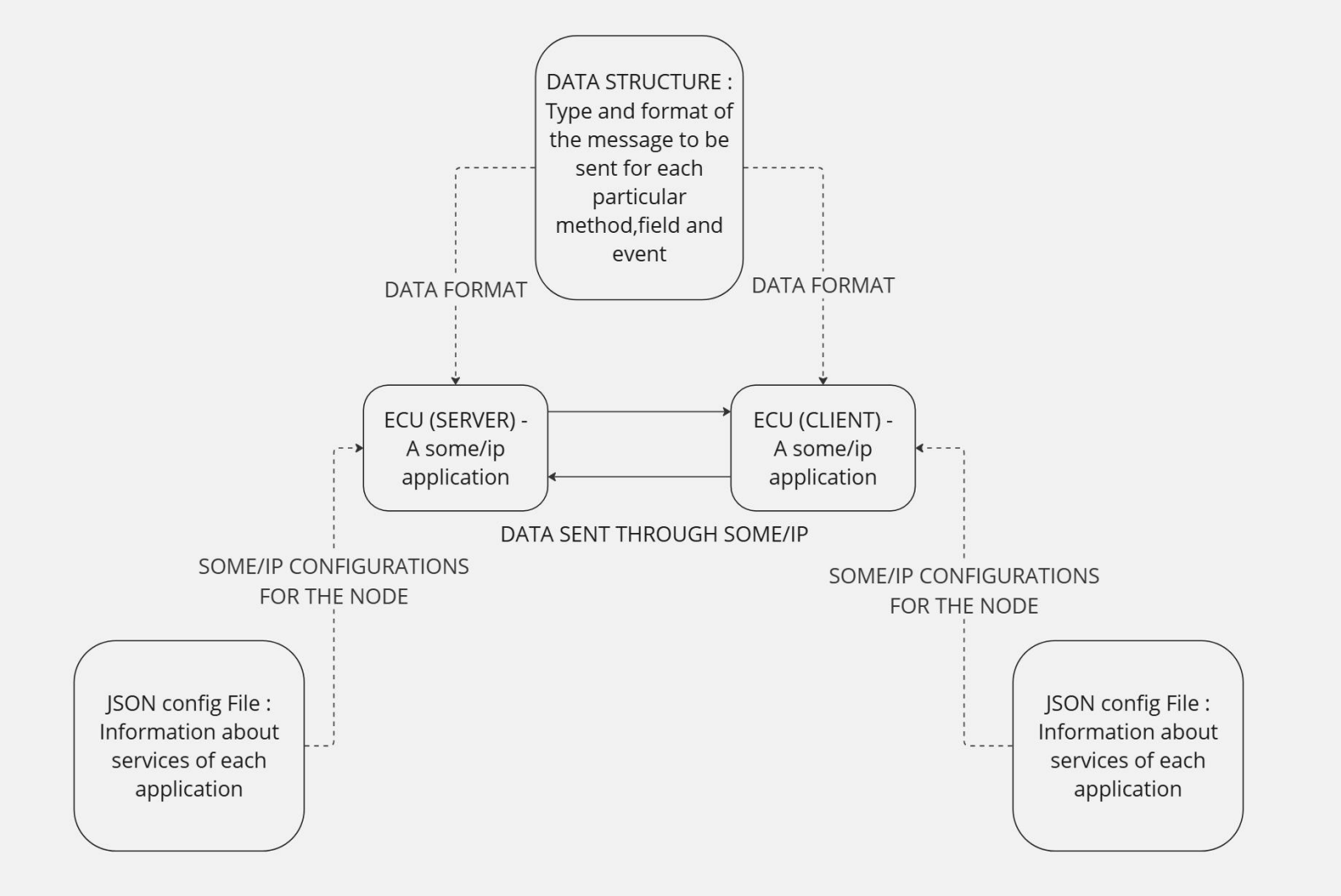
# ADAS Test Scenario Generator

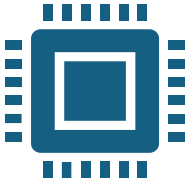
ADAS Test Scenario Generator is a collection of python scripts designed to simulate Mathematical model-based data for an ADAS (Advanced Driver Assistance System) node, mimicking the output typically generated by sensors in both format and value. This simulated data includes:

- **Sensor Data Simulation**: The script generates sensor data such as radar, lidar, and camera readings, ensuring the format and values resemble real-world sensor outputs. This includes parameters like distance, velocity, and object recognition.
- **Vehicle-Specific Data**: It incorporates vehicle details crucial for ADAS analysis, including vehicle dimensions (height, width), sensor positions relative to the vehicle, and dynamic attributes like speed and acceleration.
- **Environmental Conditions**: The scripts simulate environmental data impacting ADAS performance, such as weather conditions (rain, snow, fog), temperature variations, and climate conditions (e.g., clear skies, overcast).

The generated data is transmitted using SOME/IP (Service-Oriented Middleware for IP) protocol, ensuring compatibility with modern automotive communication standards.

# Architecture





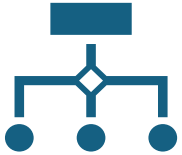
## ECU Class

ECU classes are specific for each sensor and environment

Stores SOME/IP application variable.

It contains details of all provided and required services of the ECU

The config file path is found here



## Config Files

JSON files for each ECU for configuring SOME/IP paraments.

Some parameters include

- Unicast address
- Multicast address
- Logging
- Application
- Service Discovery
- Services

## Data Files

Contains details of all services are their corresponding methods, events and fields.

It contains the data format and value types of all valid messages and notifications.

# SOME/IP

SOME/IP (Scalable Service-Oriented Middleware over IP) is a communication protocol used in automotive and other industries.

It operates at the application layer over IP, typically UDP or TCP, for efficient data exchange between electronic control units (ECUs).

It supports service-oriented architecture, efficient data serialization and service discovery

ADAPTIVE AUTOSAR uses SOME/IP as one of its communication protocol.

We have used the vsomeip libraray which is an implementation of SOME/IP protocol by COVESA .

## Method

- Client requests the server and Server gives response

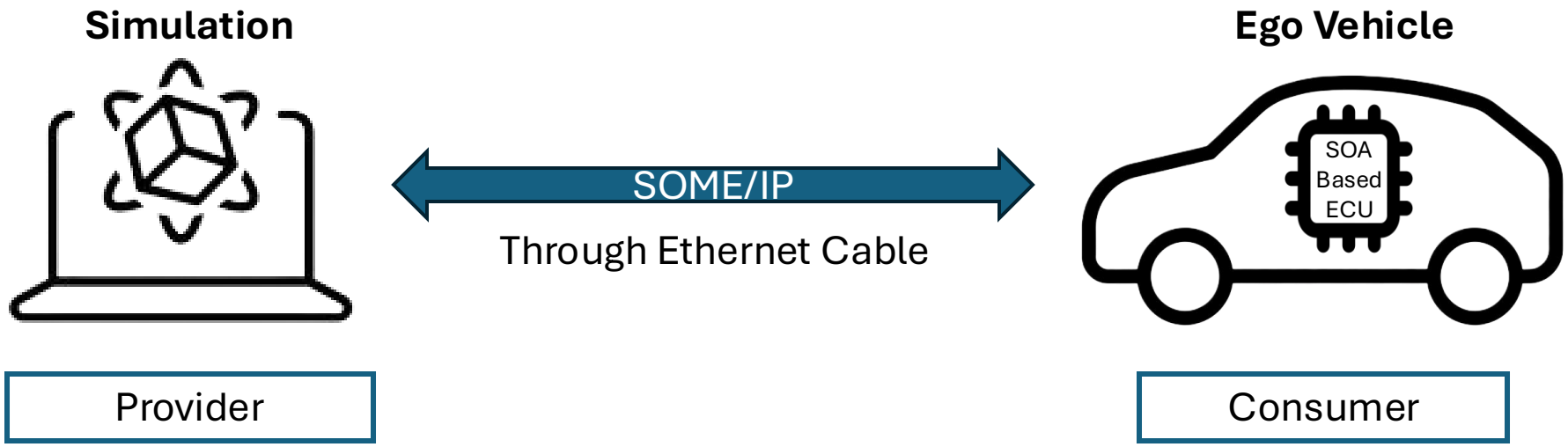
## Events

- Client subscribes to an event and gets notified whenever server publishes. Either update on change or periodic

## Fields

- A combination of method and events. A client can set and get data in the server and gets notified whenever the server publishes.

# Intended Test Bench Setup




Data simulated by the simulator include -:

- Sensor data: Short-range and mid-range radar pre-detection lists.
- Vehicle data: Dimensions, chassis details, and vehicle identification number.
- Environment data: Weather information and temperature.
- Other information: Ego vehicle state, health status of various equipment, and communication details.

Data not being simulated -:

- Camera output: Images.
- Environment data: Terrain information.

The simulator currently accommodates 60% of the required data for the Vehicle's software framework.

# Next Steps

- Expand simulation output data to include
  - Camera data (Images)
  - Environmental data such as terrain information
- Develop a user-friendly GUI interface for operating the simulator
- Implement real-time data visualization capabilities.