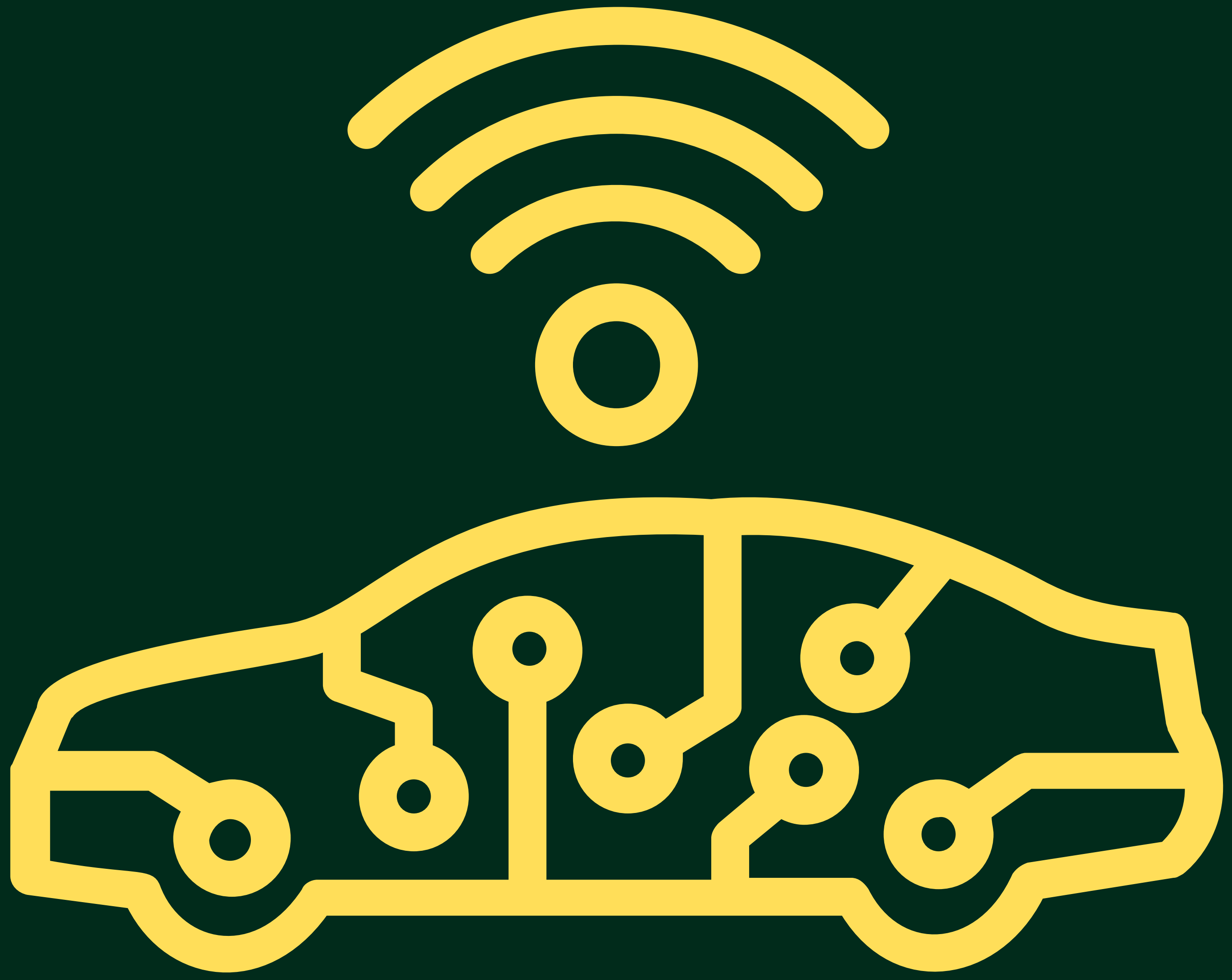
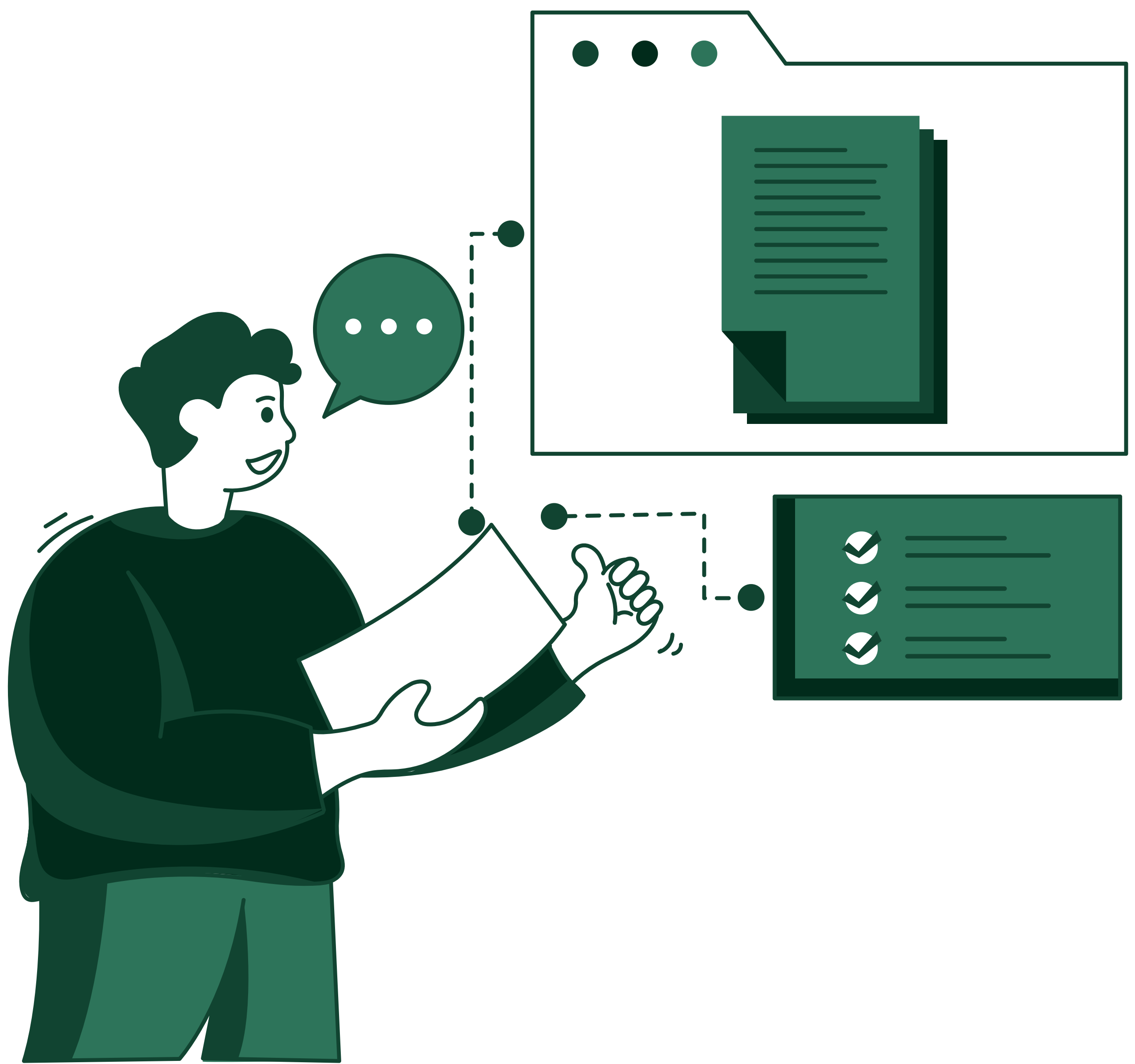
# TEAM 20: EDGING

Design and Implementation of a Plug-and-Play Edge-Cloud Architecture for Retrofitting EVs with Autonomous Safety Features



Presented by:
Tanushri Ravish
ChandraMouli K
Gopika Gokul
Bhuvanesh

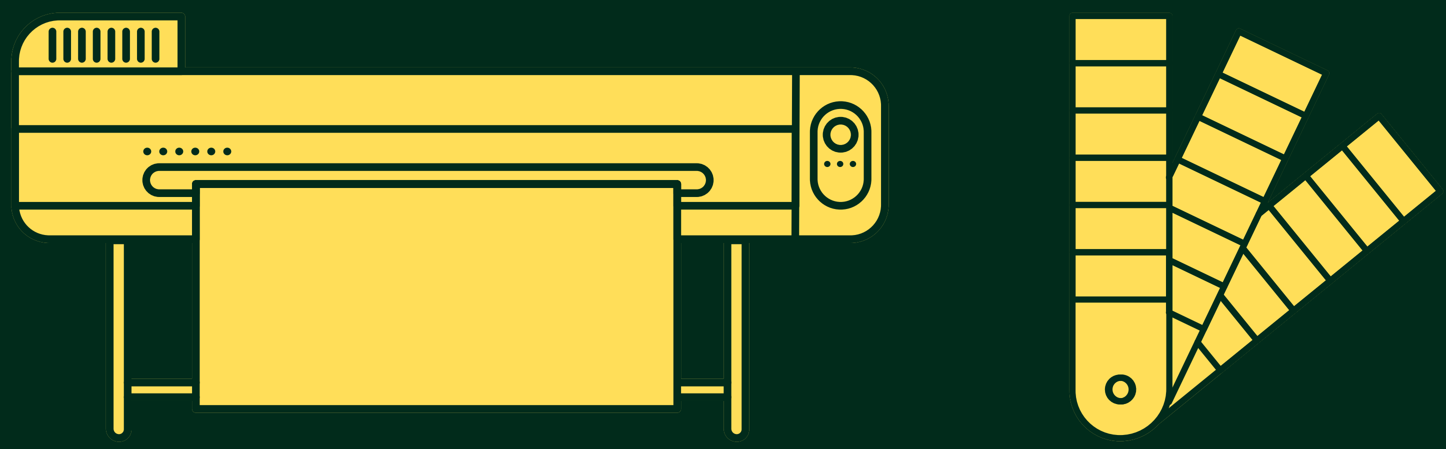
# What exactly we are doing:



1. We are designing a plug-and-play system to retrofit EVs with autonomous safety features using edge-cloud architecture.

2. Tasks are intelligently balanced between the edge (real-time safety) and cloud (analytics, updates).

# What is the problem:



Most existing EVs lack smart safety features , and current retrofitting methods are complex and expensive

They often fail to meet real-time requirements or rely heavily on connectivity and manual integration.

Our project is going to focus on automatic braking in emergency situations.

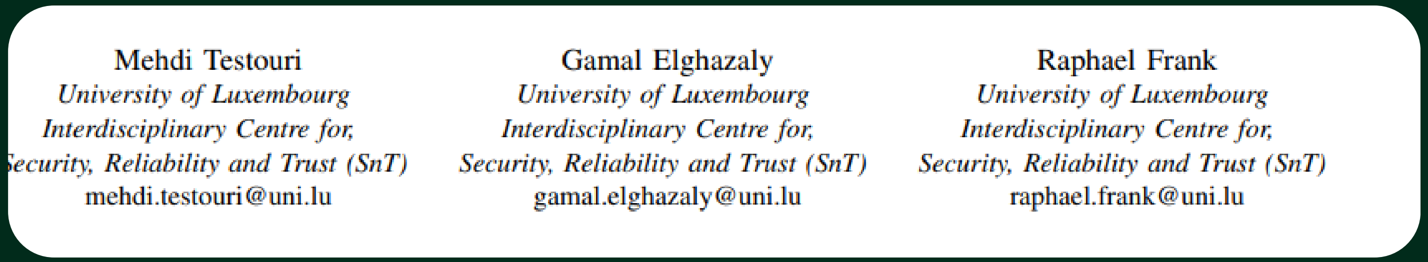
# Why is it an edge project:

1. Edge computing handles latency-critical tasks locally, ensuring immediate response for safety functions.

2. Dynamic task balancing allows the system to shift non-urgent processing to the cloud, optimizing performance and resource use.

# STATE OF THE ART. LITERATURE REVEIW:

## 1. ROBOCAR



Mehdi Testouri
University of Luxembourg
Interdisciplinary Centre for,
ecurity, Reliability and Trust (SnT)
mehdi.testouri@uni.lu

Gamal Elghazaly
University of Luxembourg
Interdisciplinary Centre for,
Security, Reliability and Trust (SnT)
gamal.elghazaly@uni.lu

Raphael Frank
University of Luxembourg
Interdisciplinary Centre for,
Security, Reliability and Trust (SnT)
raphael.frank@uni.lu


**Summary:**

**Purpose**: Offers an open-source, low-cost, modular research platform for autonomous driving, based on the 2018 KIA Soul EV.

**Architecture:** Modular software in ROS2, with a real-time control loop integrating GNSS-INS (Trimble BX992), LiDAR (Ouster OS1), and cameras (Sekonix, RealSense).

**Components:**

Hardware: Drive-by-wire electric vehicle, sensors, edge computing unit (Intel i7, GTX 1060).

**Software:** ROS2 middleware, with core modules for localization, mapping, perception, planning, control, and HMI.

Planning: Uses a Model Predictive Path Integral (MPPI) algorithm for real-time trajectory planning.

**Edge Relevance:** The system shows strong modularity and real-time perception-planning-control loop suitable for edge deployment.

# STATE OF THE ART. LITERATURE REVEIW:

## 2. Vehicular Edge Computing and Networking: A Survey

Lei Liu, *Student Member, IEEE,* Chen Chen, *Senior Member, IEEE,* Qingqi Pei, *Senior Member, IEEE,* Sabita Maharjan, *Member, IEEE,* and Yan Zhang, *Senior Member, IEEE*

Summary:

- **Purpose:** Provides a detailed survey of research on Vehicular Edge Computing (VEC), covering architectures, enabling technologies, and application domains.
- **Architecture:** Three-layer system — vehicles (user layer), RSUs (edge layer), cloud (remote layer).
- **Key Features:**
  - Computation Offloading: Edge nodes (e.g., RSUs) reduce latency by handling delay-sensitive tasks.
  - Caching: Popular content (like maps or instructions) is stored at RSUs/vehicles to reduce bandwidth use.
  - AI-Enabled Decisions: Deep learning and reinforcement learning used for dynamic offloading and task scheduling.
- **Challenges Identified:** High mobility, dynamic topology, task migration, resource allocation, and security.

# HOW WE ARE INFORMED:

| RoboCar Paper | VEC Foundation Paper |
| --- | --- |
| Offers a concrete platform with software-hardware stack for retrofitting | Defines theoretical backbone for integrating edge-cloud architectures |
| Sensor fusion and real-time planning can be migrated to edge nodes | Edge nodes can process latency-critical tasks like emergency braking |
| Demonstrates simulation + real-world deployment | Gives design principles for computation offloading, caching, and networking |
| Uses ROS2 for modularity, aiding plug-and-play design | Mentions mobility-aware offloading and caching to reduce bandwidth |

# INTEGRATION STRATEGY

**Edge (ESP32):**
Reads distance from HC-SR04 Ultrasonic Sensor
Controls DC motors via L298N Motor Driver
Makes real-time braking decisions locally
Sends data (distance, speed, status) to cloud via Wi-Fi
**Hardware Setup:**
RC Chassis: Holds all components
ESP32: Central controller
Motors + Driver: Enable movement
Sensor: Detects obstacles
Battery: Powers all modules

# INTEGRATION STRATEGY

**Cloud (ThingSpeak)**

- Receives data from ESP32 over Wi-Fi
- Logs & visualizes obstacle distance, system state
- Enables remote monitoring

**Data Flow**

1. Sensor → ESP32
2. ESP32 → Motor Control (locally)
3. ESP32 → ThingSpeak (via Wi-Fi)

# T A S K S P L I T

TANUSHRI: SENSOR CALIBRATION AND
RC TEAR APART.

CHANDRA MOULI: ESP32 AND BRAKING LOGIC FOR ACCURATE
SAFETY PROTOCOL

BHUVANESH: EDGE- CLOUD NETWORK AND CLOUD
INTEGRATION.

GOPIKA: ESP32  INTERFACE WITH EXPRESSIF IDE OR SIMILAR
TO EFFICIENTLY PUSH CODE.