

The background of the entire page is a dark blue gradient. Overlaid on this are several white geometric shapes: a large triangle pointing downwards in the center, and a horizontal line passing through its base. Concentric white circles are centered within the triangle, creating a target-like effect. The text is positioned in the upper and lower portions of the page, avoiding the central geometric design.

# **AUTONOMOUS DRIVING ROBOT**

## **TECHNICAL PROPOSAL**

**JUNE 8, 2020**

**RAGHUNATH (RANA)**  
RA@IEEE.ORG (OR) RAGHUNATH@TVSMOTOR.COM  
+91 74110 72448

# A Technical Proposal to TVSM for Autonomous Driving Robot

# CONTENTS

---

The Author.....	5
1 Background .....	6
2 Generation-1 Implementation .....	7
2.1 Technical Summary & Vision .....	7
2.2 Commercial Summary.....	9
2.3 Architecture .....	11
2.3.1 Safety.....	11
2.3.2 Physical Entities.....	15
2.3.3 Embedded Software .....	16
2.3.4 Electronics (Hardware) .....	17
2.3.5 Mechanical Actuators.....	19
2.3.6 Communications.....	20
2.3.7 Deep Learning Models.....	23
2.3.8 Actuator Control .....	24
2.3.9 Sensor Network.....	26
3 Gen-1 Tentative Specifications .....	28
3.1 Autonomous Driving Robot .....	28
3.2 Cameras .....	29
3.3 RADARs .....	30
3.4 Other Sensors.....	30
3.5 Electronic Control Unit.....	31
3.6 Actuators.....	32
4 Failure Tree.....	33
5 Challenges in Execution.....	33
5.1 Communication Network Bottlenecks .....	34
5.1.1 Network Coverage and Speed .....	34
5.1.2 Network Delay.....	34
5.1.3 Missing Network Channel.....	35
5.1.4 Lost Communication .....	36
5.2 Context Identification for Road Symbols.....	36
5.3 Vibration and Tilt .....	37

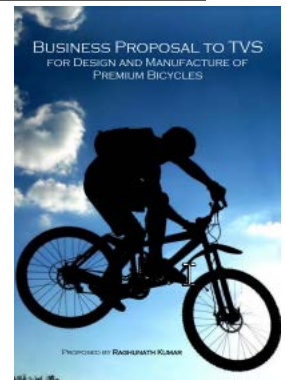
5.4	Concealed Objects.....	37
5.5	Missing Road Markings and Borders .....	38
5.6	Unstructured Traffic and Hand Gestures .....	39
6	Scope of Work .....	40
	References .....	43
	Annexure - 1.....	46
1	Choices and Preferences .....	46
1.1	Vehicle.....	46
1.2	Hardware.....	46
1.2.1	Power & Signal Distribution.....	46
1.2.2	CPU vs. GPU vs. TPU vs. NPU vs. FPGA .....	47
1.2.3	FPGA vs. DSP .....	48
1.3	Sensors.....	48
1.3.1	Mounting .....	48
1.3.2	RADAR vs. LiDAR.....	48
1.3.3	RADAR.....	49
1.3.4	Cameras .....	51
1.4	Embedded Software.....	53
1.4.1	Bare-metal OS vs. RTOS.....	54
1.4.2	RTOS.....	55
1.5	Deep Learning Models.....	55
1.6	Actuators.....	58
1.6.1	Steering Wheel.....	58
1.6.2	Foot Pedals .....	58
1.6.3	Gearstick .....	58
1.6.4	Steering Mounted Controls.....	59
1.6.5	Robotic Limbs.....	59
1.7	Limp-home Mode.....	60
1.8	Memory Management .....	60
1.8.1	Storage.....	60
1.8.2	Data Transmission .....	61
1.9	Automatic Doors.....	61
1.10	Smart Features .....	61

2	Testing .....	61
3	Scope for Innovation.....	62

## THE AUTHOR

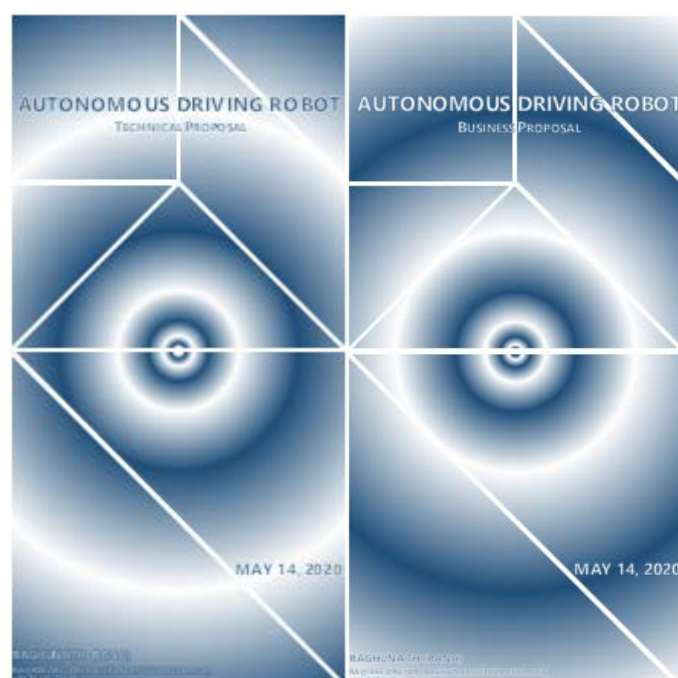
**RaghuNath** (RaNa) is an IIT-Madras graduate with a Dual Degree (B.Tech+M.Tech) in Electrical Engineering.

- About 14 years of industry experience
- Received the service award (10-year) from TVSM
- Worked on PoC for Anti-lock Braking Systems (ABS), Battery Management System (BMS), and Electric Motors
- Holds the *highest* number of *granted patents* in AEG/TVSM as the first inventor, also has 15+ patent applications pending decision
- Proposed **Premium Bicycles Business** to TVSM in 2011, and the market growth predictions are close to reality
- Instrumental in the planning & support for the proposal of **LED Lighting Business** to TVSM in 2012
- Research paper in **Computational Economics** got accepted for a presentation at WEAI-2013 (Western Economic Association International) in Seattle, USA
- Delivered technical presentations in several international conferences in the fields such as ABS, BMS, Embedded Systems, Electric Motors, and Optimization
- Worked on **machine learning and deep learning projects** aimed at autonomous driving
- Has hands-on experience of full product design cycle including system modeling, hardware design, software programming, testing, and project planning



He is technically strong with a superior research appetite and a practical business aptitude - an exceptional combination.

These are the set of documents prepared for the proposal of *Autonomous Driving Robot*.



# 1 BACKGROUND

---

Self-driving vehicles have an extensive history, dating back to the automobiles' presence on the roads. Leonardo da Vinci's self-propelled cart [2] is probably the first robot, rather than an automobile. Francis Houdina [3] demonstrated a remote driving setup in the New York City roads in 1925. In 1939 New York World's Fair, Futurama [1] designed by Norman Geddes and sponsored by GM had reference to autonomous vehicles and automated highways.

The Stanford cart [4] project, active during 1960-80, is one of the first few robots. It used a camera to detect and follow a white line. In 1969, John McCarthy referred to an *Automatic Chauffeur* in his essay *Computer-Controlled cars*, which operates based on visual information available to the human driver.

In 1987, Ernst installed many cameras and processors on a sedan (project VaMoR) to detect objects on the road using what he called *dynamic vision*, and it navigated the Autobahn at around 100kmph.

Dean Pomerleau described how neural networks could use cameras' raw images to generate steering control angle in real-time in the 1990s. In 1995, his minivan traveled 4500km.

In 2002, DRAPA announced a grand challenge to navigate 228km through the Mojave desert.

Self-parking gained popularity in the 2000s. Google started its Waymo project in 2009, and in a few years, its cars finished 4.8 lakh km without a single incident. Then all other automotive OEMs stepped-in with their own parallel autonomous vehicle programs. These programs gave rise to safer ADAS features in the present generation of vehicles.

Tesla released its Autopilot in 2014, and it became popular, although not error-free. Audi A8 is most likely the first Autonomous Level 3 production car.

Several large multinational organizations are working either independently or collaboratively to work on autonomous vehicle technology. Few startups are working products that solve a very focused problem in the application of autonomous vehicle technology.

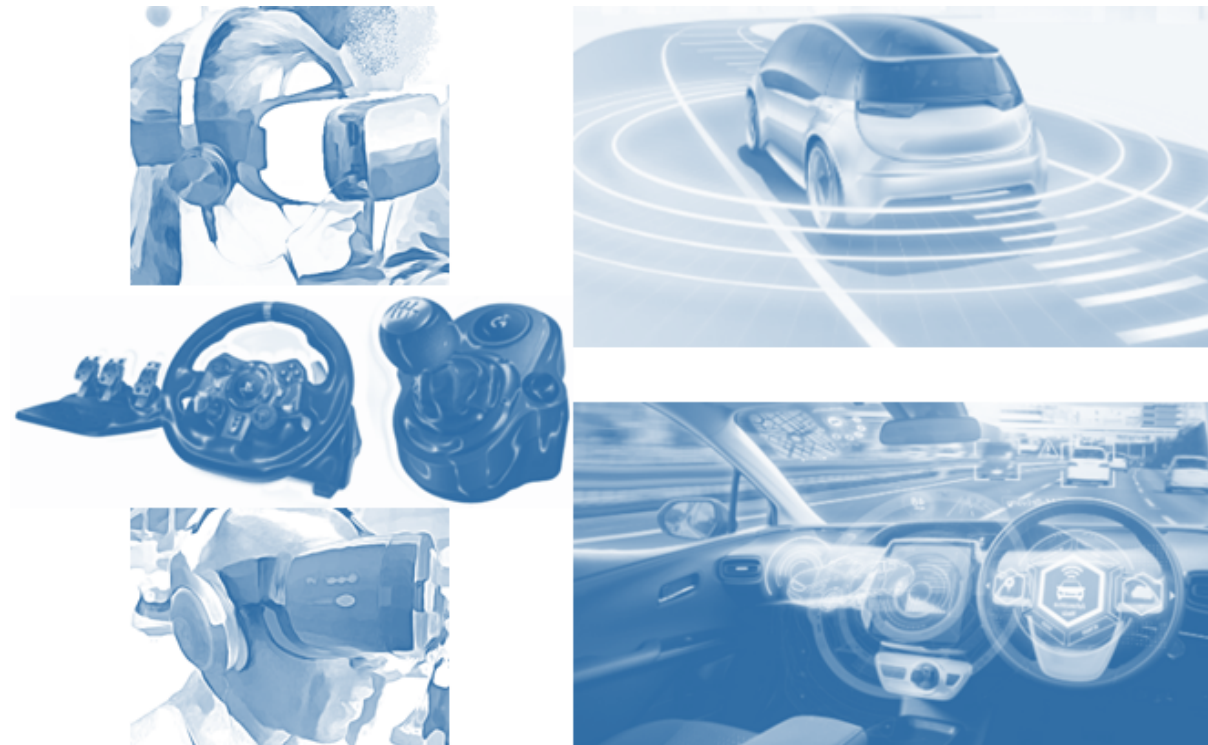
India has seen fewer demonstrations or activities in this direction. There have been many projects in the academia for the proof-of-concept or for validating one specific algorithm. But the research for product development has not been done.

An individual enthusiast demonstrated his autonomous vehicle (Tata Nano) in February 2016. Swaayatt Robots demonstrated their first functional prototype on a retrofitted Mahindra Bolero in August 2016. Interestingly, Mahindra demonstrated their prototype tractor working in the fields in September 2017. Fisheyebox demonstrated its prototype on Maruti Suzuki Celerio (AMT) in October 2017. An individual farmer from Rajasthan demonstrated a home-brew autonomous tractor with limited functionality in May 2018. Swaayatt Robots showed good progress over the years and implemented some of the smartest and fastest algorithms using only camera data for Indian traffic navigation.

This document tries to identify the scope of this project to create an impactful solution.

## 2 GENERATION-1 IMPLEMENTATION

### 2.1 TECHNICAL SUMMARY & VISION



The generation-1 (Gen-1) driving robot fitted vehicles will have at least L3 autonomy and surrogate driver support (to operate the vehicle controls) from the command center.

The remote operation of the vehicle will be enabled by the telepresence system accessible through the virtual reality (VR) headset at the command center. Faster data transfer via 4G/5G networks combined with data optimization algorithms will enable almost delay-free vehicle control by the surrogate drivers. We will equip the robot controller with multiple eSIM modules to ensure faster data transfer and network coverage at all times.

Investments in 5G infrastructure from technology giants such as Reliance (Jio network) and satellite-based internet connectivity by SpaceX's Starlink project give hope to higher-speed connectivity to services such as the surrogate driver of our project. Encryption at multiple levels ensures data security (network communication, data storage in-vehicle/cloud, and algorithms & software storage onboard processor memory).

When the vehicle connectivity with the command center is reduced, we switch to low-bandwidth mode using our special algorithms. When the connectivity is unavailable in all the parallel channels, the vehicle identifies a low-risk spot nearby. It safely navigates and parks itself till the connection is established, or some authorized user from within the vehicle overrides the vehicle control manually to move the vehicle.

The vehicle (with diesel IC engine, automatic transmission for Gen-1) will have RADAR (LRR & SRR), (Stereo) Cameras, and other sensors. We will avoid using commercial LiDARs. The lack



of LiDAR will be compensated by robust image processing algorithms that build a 3D model of the environment from the camera & RADAR data.

We will have multiple deep learning models running on GPU cores for all critical activities (pedestrian detection, road detection, context detection). We will capture essential vehicle data along with the user inputs to train our deep learning models for improvements.

The models will be tuned to operate in the unique traffic conditions and roads of India. The driving robot will have computationally less-intensive algorithms that detect and classify objects. The driving robot builds a 3D model of the surroundings with an abstract representation of all the important objects, simplifying later modules in the pipeline. We will implement the modular development of system components that help reduce the complexity of the design and enable assigning work packages to independent groups.

The vehicle will most likely operate in a known route and will be requested for service through a mobile app by the users from their locations. The vehicle will pick the users from the nearest earmarked locations and will drop them at their destination. The driving robot honors multiple requests and recalculates the route.

We replace the factory-fitted steering wheel with a custom one, including a rotary actuator, and attach the wiring harness connectors. We replace the driver's seat with the custom seat in which we secure the driving robot with bolts and nuts to fix the position.

We latch the driving robot's pedal actuators to the corresponding pedals on the vehicle with automatic transmission. Once the actuators are in place, we calibrate them to match the positions of the vehicle controls or load with the pre-calibrated data from the server. Further developments will focus on robotic limb development that can adapt to any vehicle.

We will implement ARC (Adaptive Robust Control) for the actuators' control. It handles any uncertain situations and variations in the vehicle parameters over time

- Vehicle load alters the throttle response
- Brake pad wear changes brake feel and response

The vehicle will react to all the obstacles in its route, including slowing down and overtaking slow-moving vehicles. The vehicle will understand the indicators and other signals from the rest of the vehicles around it and move accordingly.

When the vehicle has a problem moving around or observes a failure, it will halt safely and send messages to the command center seeking help. Designated passengers in the vehicle can override the vehicle controls under exceptional situations either through the mobile app/joystick interface or through the physical control interface directly.

The vehicle will have at least one of the five vehicle control mechanisms active (driving robot, surrogate driver, joystick/mobile app, mechanical override, and limp-home mode).

The vehicle fitted with the driving robot will be fully functional. It will have the required safety certifications from concerned authorities. This implementation paves the way for further developments in the next generations of the project.

## 2.2 COMMERCIAL SUMMARY

The document titled "Autonomous Driving Robot Business Proposal" for discussing the business viability of this project is prepared and included together with this. The executive summary for the business proposal is here.

We address the transition between people-driven (standard) vehicles and autonomous vehicles. We mount a driving robot in the driver seat that adapts to any vehicle and provides the same driving experience as any other autonomous vehicle. The driving robot (our USP) is contrary to the traditional approach of building autonomous vehicles from the ground up.

Indian passenger vehicle market grew at a CAGR of 6.2% during 2013-19, with an approximate addition of 3.3 million vehicles a year [1], while the world passenger vehicle market adds 77 million vehicles a year [2] to the roads.

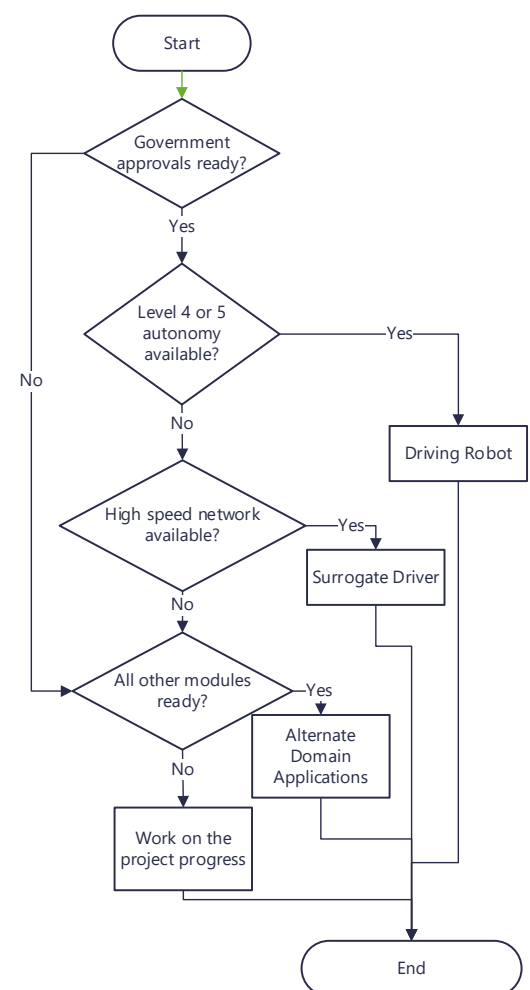
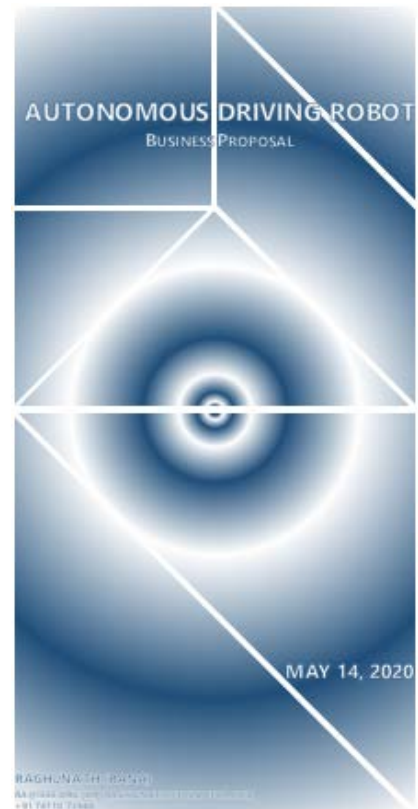
With an average vehicle usage of 11 years [3], about 850 million standard vehicles (36 million from India) will coexist with the autonomous vehicles when they become a reality.

A 2% market capture, with US\$ 11,000 per kit that adds autonomy to standard vehicles, results in a worldwide market of US\$ 185 billion (US\$ 8 billion in India). The subscription model gives scope for a higher realistic price and ensures profitability for all.

The relevance of this project comes from the ongoing social distancing practices getting interwoven in everyday activities. Commercial fleet operators face increasing indirect costs with human drivers. Driving robot is a useful product to address these challenges, in addition to improving vehicle utilization and reducing the accident rate.

Level 4 & 5 autonomy is in the research stage, and most companies are trying to build autonomous vehicles on a new platform from scratch by integrating the sensors & controls. This approach involves high development costs and results in expensive end-product.

Our product fits on standard vehicles, even without electric transmission. It can be shifted to other vehicles when the first vehicle encounters a problem, ensuring that the investment is available for utilization at all times.



Realistically, we can expect to deliver the 1<sup>st</sup> generation product (Gen-1) in 3 years with an estimated 10-year running life. Next-generation product focuses on improving the performance & features while reducing cost, size, and weight.

When the opportunity exists, we can configure our products to work for industries with hazardous environments or those looking for reduced human involvement.

We plan to work on this project with the least possible investment requirements

- Sharing the resources with the existing R & D setup
- Using the open-source tools/datasets
- Outsourcing modularized work
- Lower permanent employee count
- Validating extensively through simulations
- Eliminating LiDARs

A series of risk mitigation methods are discussed, including the most important one – derivative business units. The project execution results in several independent derivative business units that help with the cash flow, even before the full product is ready.

The project applies to the following areas (only indicative, not limiting)

- Optimistic technology-scenario (If the industry develops L4 or L5 autonomy solutions)
  - *Autonomous Driving Robot*
- Realistic technology-scenario (If 4G network delay is not limiting, or 5G is available)
  - *Surrogate Driver*
- Pessimistic technology-scenario (Possible, no definite limitations exist)
  - *Vehicle Testing Robot*
  - *Construction Equipment Operator*
  - *Farm Equipment Operator*
  - *Last-mile Delivery Vehicle Operator*
  - *Hospitality Industry*
  - *Medical Assistance for Non-Critical Services*
  - *Assembly Plant Operator*
  - *Hazardous Zone Vehicle Operator*
    - *Radio-Active, Mine-Field, Under-Water, Extreme Temperatures, Poisonous Gasses, Deep Excavation, Excessive Load Handling*

Given the market potential, the scope for profitability, individual's social distancing needs, increased availability of skilled resources in the pandemic time, reduced risk through derivative businesses, availability of computing power, and faster interface with the servers, now is most likely the right time to start this project to get ahead of the competition.

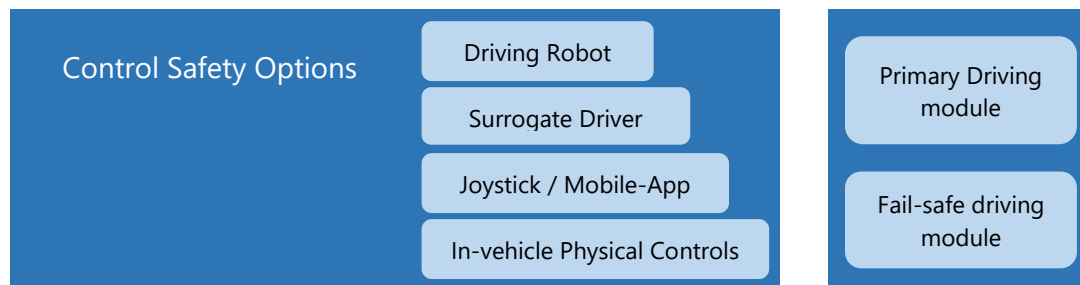
## 2.3 ARCHITECTURE

### 2.3.1 Safety

Safety will have the highest priority in the design. The following modules highlight the alternatives in each of them for ensuring user safety.

#### 2.3.1.1 Vehicle Control

Safe vehicle control is essential. There are four levels of controls for the vehicle, and at least one of them will be operational at any time. Driving robot and surrogate driver modes are the default for all the vehicles as a part of the product offering. The drive-by-wire (joystick or the mobile app) interface will be a part of the debugging system during the development. We will give the joystick access to the customers only when required.



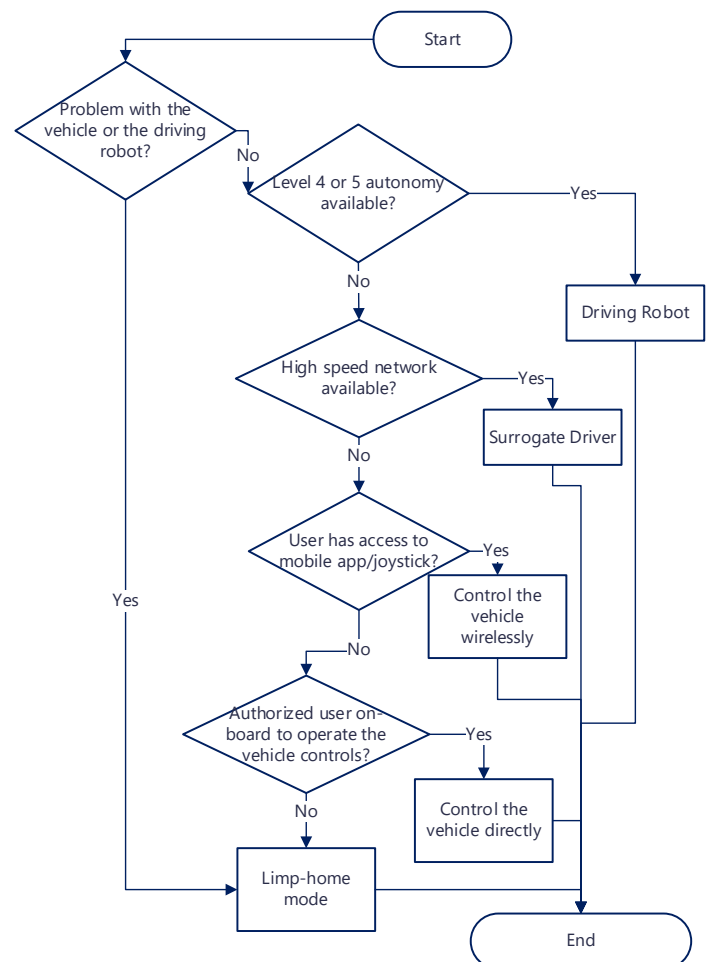
We ensure that every control operation that the driving robot arrangement does come with a mechanical override, in the worst case of vehicle power supply failure.

The fail-safe driving module gets activated

- When the driving robot detects a problem with the vehicle (or sensors or the actuators)
- However, it cannot establish contact with the command center and when the users cannot override the controls

The purpose of this is to identify the nearest safe parking zone and guide the vehicle at very low speeds.

The driving robot comes equipped with a compact Li-ion battery pack, in addition to its connection with the power supply in the vehicle. The battery pack helps the robot maintain communication with the command center and transfer data when the vehicle power system fails. The regular operation of the driving robot does not need this power supply.

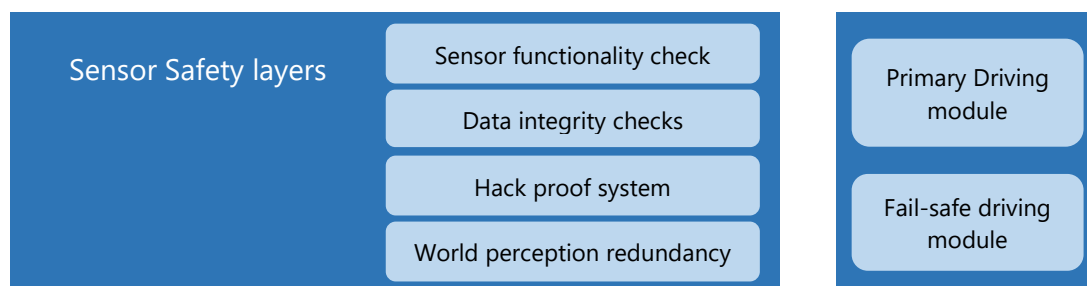


Each actuator comes with a standalone battery pack of adequate size allowing the users to operate the vehicle using the joystick/mobile-app method. This arrangement lets the user move the vehicle to safety. This battery pack will be used only during worst-case and emergencies.

The driving robot will not occupy the full seat space. It leaves sufficient space for the users to be seated, allowing the users to operate the vehicle under mechanical override mode.

### 2.3.1.2 Sensors

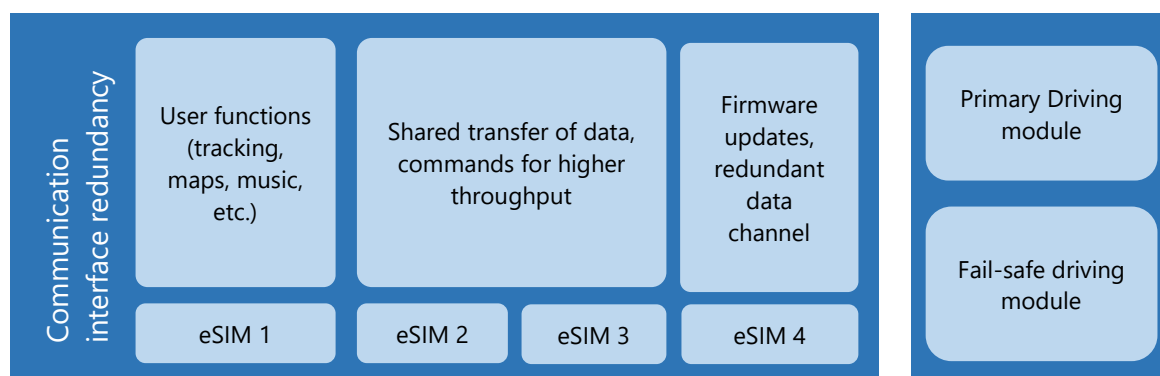
Sensors are the key to understanding the surrounding. Our implementation of the driving robot depends primarily on camera-based perception (appended by the radar information) of the world. Failure of any of the sensors results in wrong control of the vehicle; in some cases, an accident.



The processor checks sensors' health at the start of operation and during the operation of the vehicle. The data is monitored continuously for any misses during communication (EMI or other noise source effects). The driving robot controller runs sanity checks on the data from the sensors to ensure that the data received is within the expected band, and not fabricated. We implement encryption systems (PKI based) to avoid any Cybersecurity related issues. Finally, the surroundings will be covered by more than one sensor type at each place so that the redundancy will be maintained. The controller checks sensors data to see if there are any discrepancies in what different sensors observe at any location. If there is any continuous deviation at a location, the driving robot enters a fail-safe mode of control. It runs the self-calibration on all (or the corresponding) sensors.

### 2.3.1.3 Communications

Connectivity with the command center, infrastructure, or other vehicles is crucial in maintaining the safe driving behavior for the driving robot. The connectivity primarily depends on the mobile network. We use multiple eSIM interfaces for redundancy.



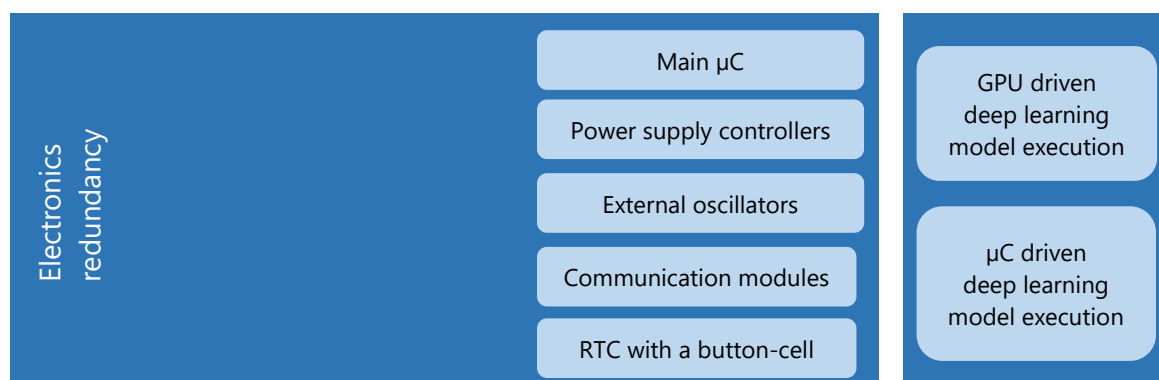
We establish redundant communication interfaces that help us achieve higher data transfer. We will probably not choose to transfer all the raw data from the sensors to the command center. The data after sanity checks and redundancy removal will go to the command center. The expected requirement of the upload rate is about 50MBps. This data gets shared among the available network service providers to go to the command center.

We will employ multiple network connections, to reduce the risk of one network operator not providing sufficient signal coverage in a given location. In the surrogate driving mode, the operator has access to the standard resolution of the images. For higher resolution or additional data, the operator sends a request to the vehicle.

The communication modules are provided with independent Li-ion battery packs for uninterrupted connectivity as long as the vehicle is active. If for some reason, all the communication modules stop functioning, the control shifts to fail-safe mode.

#### 2.3.1.4 Electronics, ECU, Power Supplies

Electronic circuits are physical entities that operate the driving robot. These circuits also include GPUs that run many computations in parallel (and independently) to run a few deep learning models. ECU hosts the embedded software that runs all the control algorithms. The correct functioning of all of these is essential for the safety of the vehicle.



We will select all the ICs, passive components, and connectors of the ECU only from the automotive-grade list. The controller includes redundant power regulators and voltage converters for the critical modules & components. This redundancy ensures that the surrogate driving or the joystick driving is made possible during vehicle power system failure.

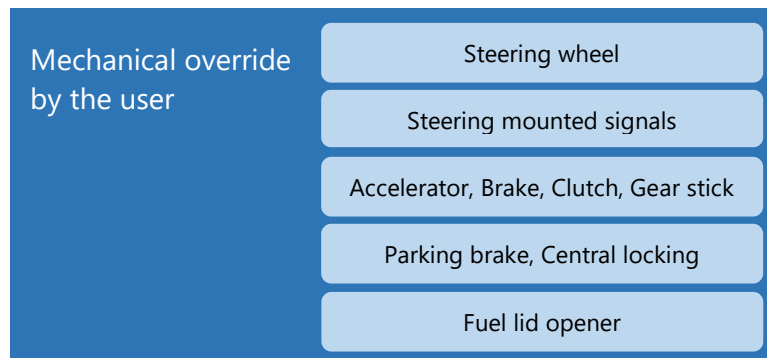
When there is an issue with the GPU system to run the deep learning models, or if the GPU system fails for some reason, we execute the algorithms and the model in the microcontroller. We will prepare the compatible deep learning model to suit both GPU runs and the microcontroller runs. Usually, the microcontroller run will take longer than the GPU run. In such a case, the robotic controller goes into limp-home mode and limits its top speed, or goes into fail-safe control mode.

The ECU will have a duplicate of the main microcontroller. Both the ICs will have access to all the inputs. Both calculate the outputs, but only the one assigned as the master sends the commands to the required modules. If there is a failure of the primary IC, the second microcontroller assigned as a slave will be auto-assigned as the master.

More than one oscillator and RTC modules will be on the controller for higher reliability.

### 2.3.1.5 Mechanical Controls

We provide a mechanical override for each of the controls operated by the driving robot. The override will be useful in an extreme situation where the driving robot fails to operate, the communication network fails, the joystick does not operate, and the power fails. The user has an option to physically use these controls by sliding the driver-seat rearward to make space for him and accessing the steering, and other levers.

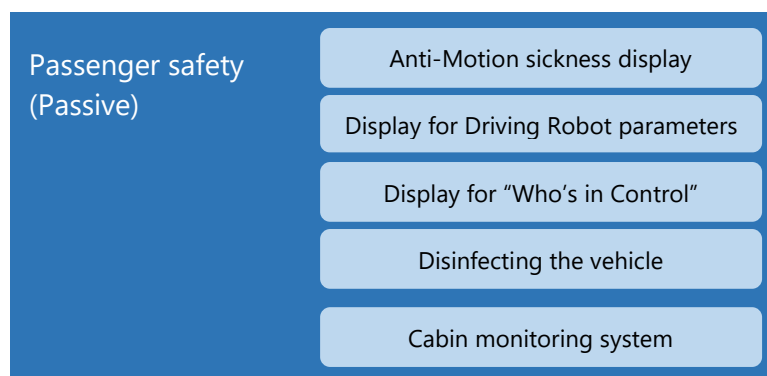


Each of the items mentioned in the above list is essential to ride the vehicle by the designated or authorized passenger.

Fuel lid openers will be optionally modified to have no mechanical linkage from the driver's seat. Ford's newer variants of cars come with an Easy-Fuel system where the fuel lid is electronically operable by sensing the fuel type, and the shape of the object entering the fuel tank. The top cover is a push-to-open type. This system is theft free, and leak-free too.

### 2.3.1.6 Health

Passive health systems monitor and assist passenger well-being while they are in the vehicle. The active safety system is the primary controller that works to ensure vehicle safety (and hence passenger safety).



We provide an optional display (accessory) for the passengers that shows the horizon at all times. The display takes feed from the forward (or a side) facing camera. It removes the vibrations and tilt-effects to maintain the display axis parallel to the ground level. The display helps with the eye-ear coordination to reduce nausea (due to visual vertigo).

We provide a dedicated display unit for indicating all the vehicle parameters and the other internal parameters of the driving robot. This display re-assures the passengers that the driving robot is handling all the road tasks.

We provide securely authenticated data to the passengers in the case of the surrogate driver mode. This data includes the operator (from the command center) driving license details, experience, ratings, and the visuals available to the command center.

We disinfect the vehicle as required at the end of the ride at our partner locations.

For the personal safety of the other passengers, we monitor the cabin space of the vehicle to ensure we keep track of passenger activity. This data will be made available to the law enforcement agencies in case of any mishap (verbal or physical abuse, aggressive co-passenger, property damages, items left-behind) in the vehicle.

### 2.3.2 Physical Entities

The following represents the architecture plan for the hardware, sensor network, and electronics.

Figure 1 gives a clear picture of the description in section 3(1), including the safety layer implementation from section 3(3) (1). We can notice that the joystick and manual override controls are part of the mechanical system. These two interaction mechanisms provide a local (non-remote) control for the vehicle.



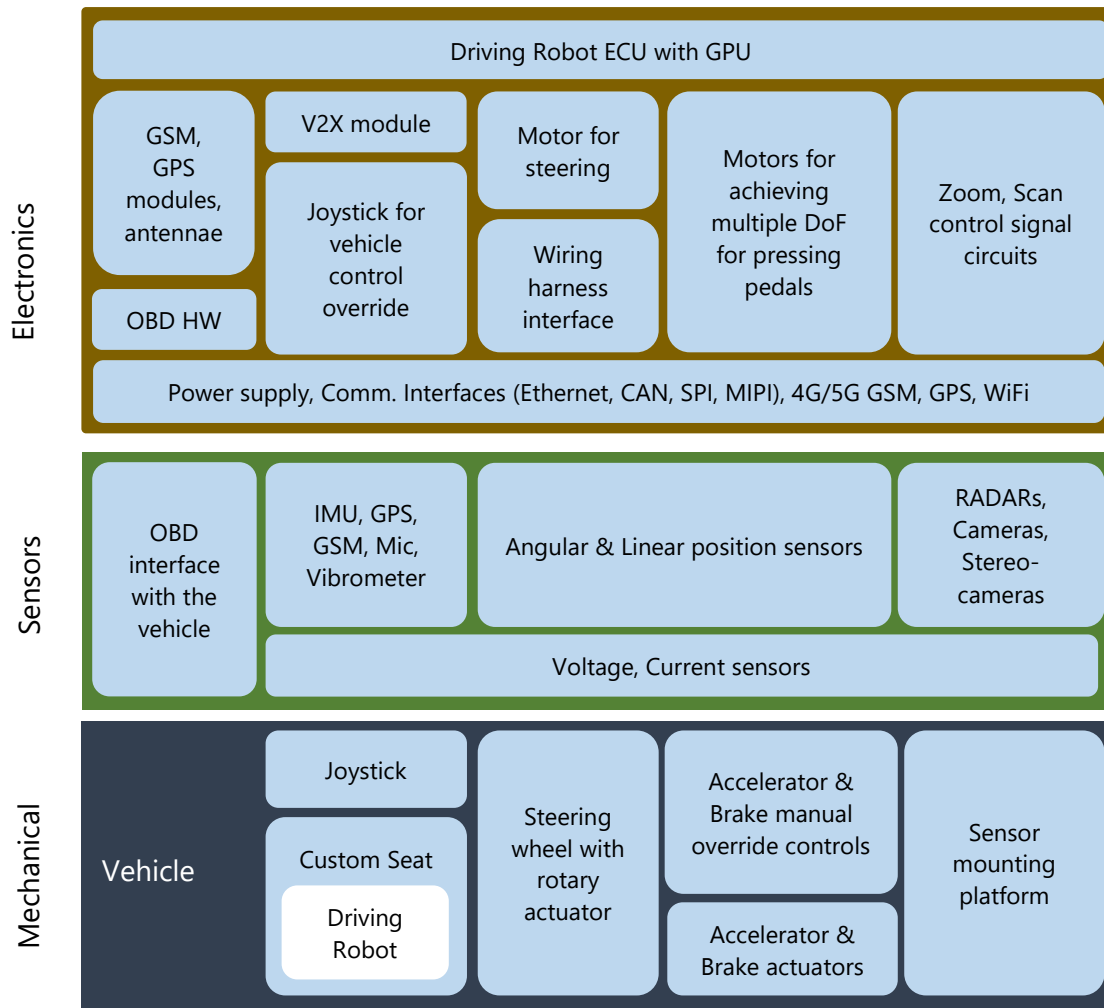


Figure 1: Architecture of physical units of the driving robot

### 2.3.3 Embedded Software

The software written on the microcontrollers gives expression to the control algorithms in addition to creating a primary execution environment for the user functions. The microcontroller will have a secure boot loader. We will also implement a two-stage boot-loader to reduce the risk of losing contact with the ECU if an error occurs during the program update. We will also store the earlier program core on the microcontroller flash memory in addition to the current program to enable rollback to the previous stage.

Figure 2 gives a clear picture of the description in section 3(1), including the safety layer implementation from section 3(3) (1). We can notice that the high-speed 4G/5G interface helps with the high-speed data transfer when the surrogate driver mode is on. The joystick interface is in the figure.

We implement an RTOS-based embedded platform to create AUTOSAR compliant programs that satisfy the functional safety requirements (ISO 26262) also.

The size of the code required for the implementation of the driving robot will be huge, and it becomes incredibly complex to make changes or debug. We will follow model-based design

principles for the control code generation. We will also follow the standard industry practices of using the hardware abstraction layer to increase modularity and portability.

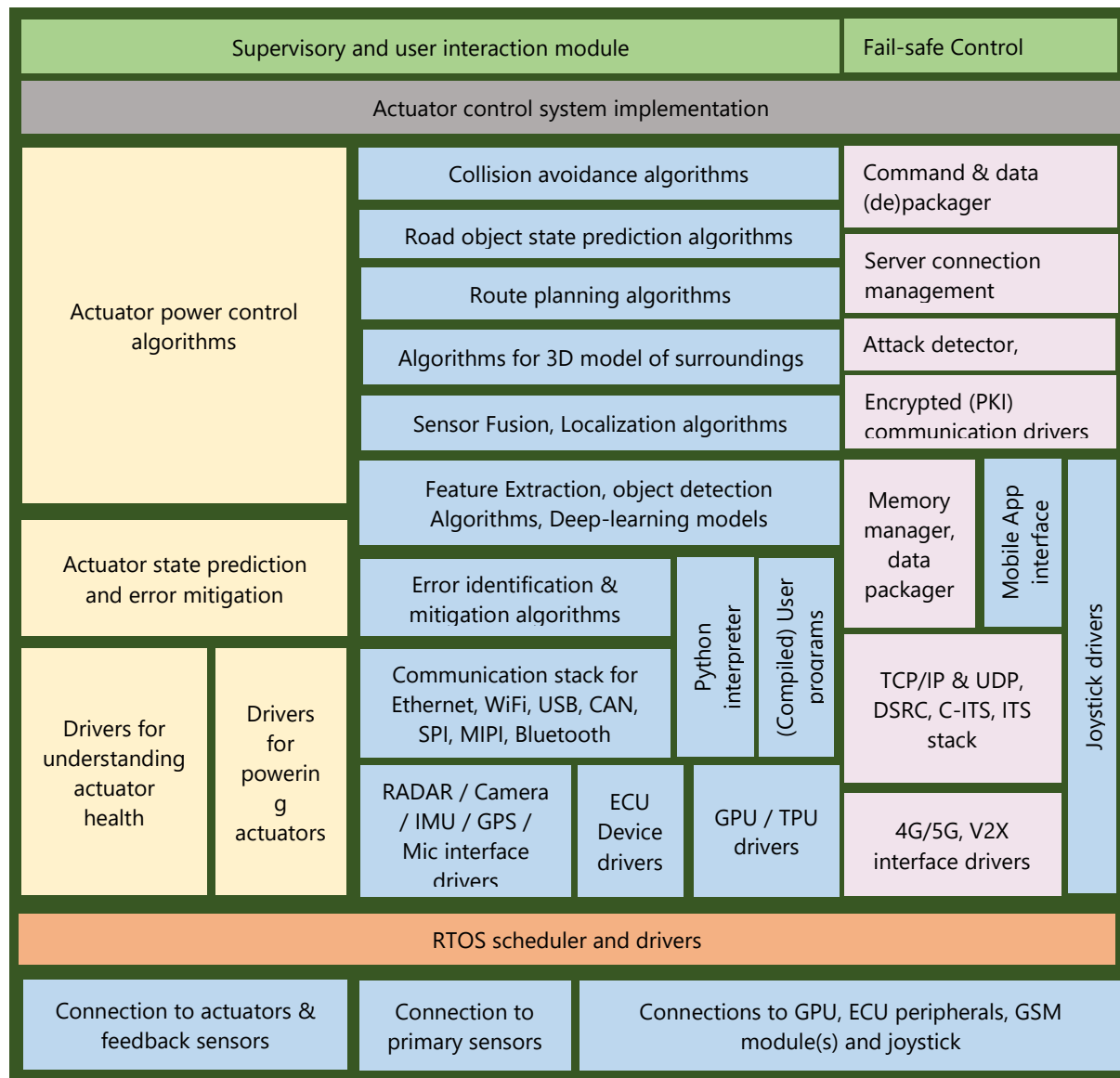


Figure 2: Software architecture of the driving robot

The choice of compiler depends on the hardware platform. The compiler decides the size of flash memory used and has a significant impact on latency.

The embedded platform will have unit tests, functional tests, and code-coverage tests created as a part of the standard development. We will also have diagnosis functions that can periodically check for any memory leaks or any cyber-attacks.

### 2.3.4 Electronics (Hardware)

The hardware will have all the necessary protections in place. We isolate the HF signal area and the signal conditioning area of the PCB from the high current handling area.

The choice of the microcontroller is essential for implementing the correct functionality of the driving robot. The microcontroller supports secure boot loading, a large flash memory, a wide variety of communication protocols, support for high clock frequencies, secure flash memory, including other essential functionalities.

We try to create modular hardware, which helps us upgrade individual modules faster. The modularity is applicable to display unit drivers, 4G/5G chipsets, and sensor interfaces.

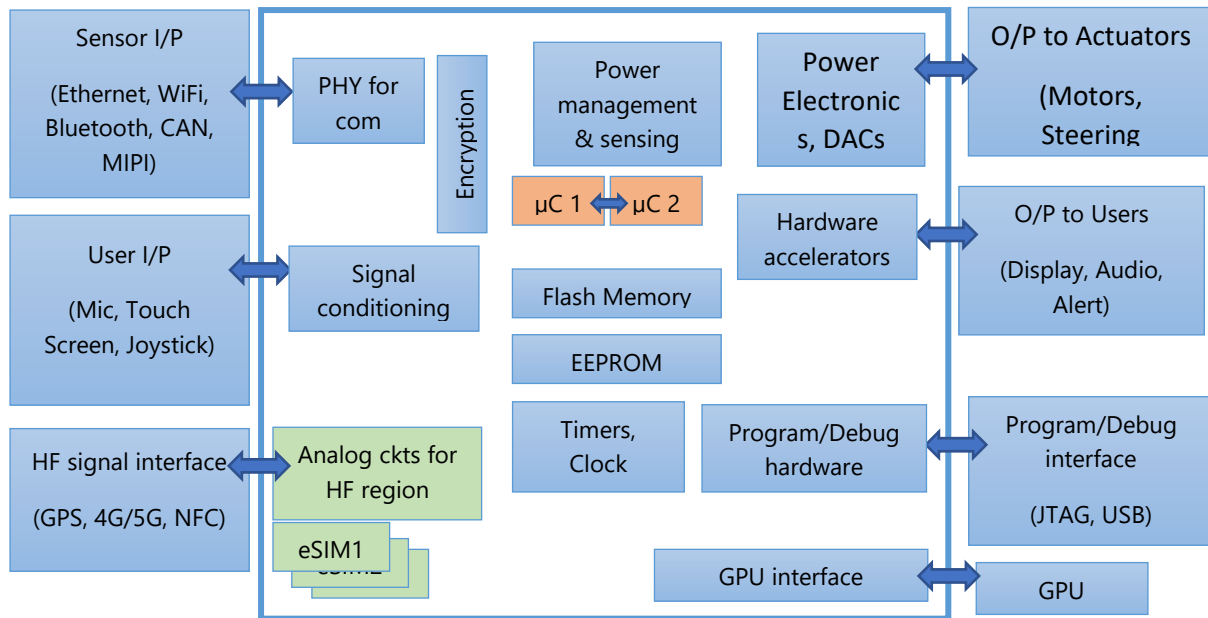


Figure 3: Block diagram of the ECU hardware

All the PHY layer elements of the communication systems will be in the ECU. Hardware encryption module unpacks the data bundles and assigns meaning to them through decryption. All the non-digital inputs will be processed through signal conditioning circuits to ensure the removal of noise elements.

GPS and GSM network PHY handles all the high frequency (HF) signals. As seen in the block diagram, the ECU will have multiple eSIM modules for redundancy of connectivity.

The power management module generates voltages as needed by different modules in the ECU, with appropriate current limits. The power module will be a switching regulator type efficient operation and better control.

External flash memory and EEPROM are on-board for storing the configuration parameters, microcontroller program image backup, and run-time data. Some data related to the driving conditions will be in the flash memory. The driving robot will send the reusable vital data to the command center or the cloud storage for tuning the algorithms and deep learning models for better performance.

The ECU comes with an identical pair of microcontrollers. At each startup, one of them will be auto-assigned as master and the other as a slave. Slave also receives the same data as the master microcontroller but does not give outputs. It only monitors the master microcontroller's outputs. If the master microcontroller fails, the slave assumes the duties of

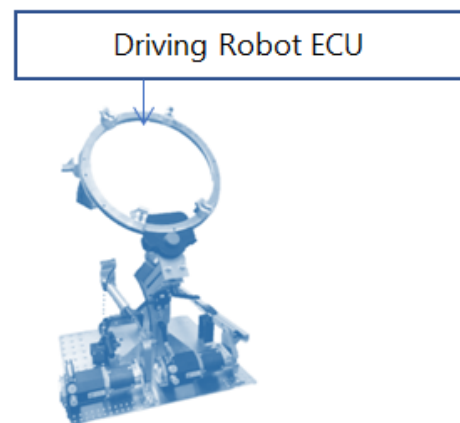
the master. When the GPU is not responsive, or the communication fails, the slave microcontroller will operate for computations. Since it takes longer to calculate on microcontroller compared to the GPU, the driving robot enters limp-home mode. This mode reduces the speed limit, providing sufficient time for the microcontroller to finish the computations.

The controller will have more than one 4G module to address regions with a single network provider and to support redundant data transfer through multiple channels avoiding loss of data packets. This multiple network channel setup will also help send more data through parallel channels faster.

### 2.3.5 Mechanical Actuators

Figure 4 gives a simple implementation of the driving robot's mechanical system. Figure 5 indicates the mechanical actuators that the ECU controls, and it shows multiple options for the steering control and foot pedal control, depending on the possibility of implementation.

The primary objective of any of the mechanical implementation is to make sure that the manual override is possible, and sufficient space for a human operator exists.



*Figure 4: An implementation of mechanical actuation*

Each actuator will have an independent Li-ion battery pack that is useful when the vehicle power supply fails. By design, the actuators' force for actuation is sufficient even when the power assist operation fails.

The steering control design is straight forward for the simple approach (a motor guided steering either mounted on the steering wheel either axially or circumferentially). For further implementation, we will look for robotic arm design.

The implementation of foot pedal operation will be done either through hydraulic actuators or linear motors or robotic leg equivalents.

The mechanical implementation of the robotic arms or legs is simple. The size of the motors used in these applications is shrinking, and the torque density is increasing. It becomes a complex control problem, and it depends on the choice of components for better controllability and faster response times.

For better and cleaner implementation, the motors in the robotic limbs will have integrated control electronics, including a secured wireless data and command transfer system. A set of wires run through the length of the limbs to reach all the motor modules for power connections.

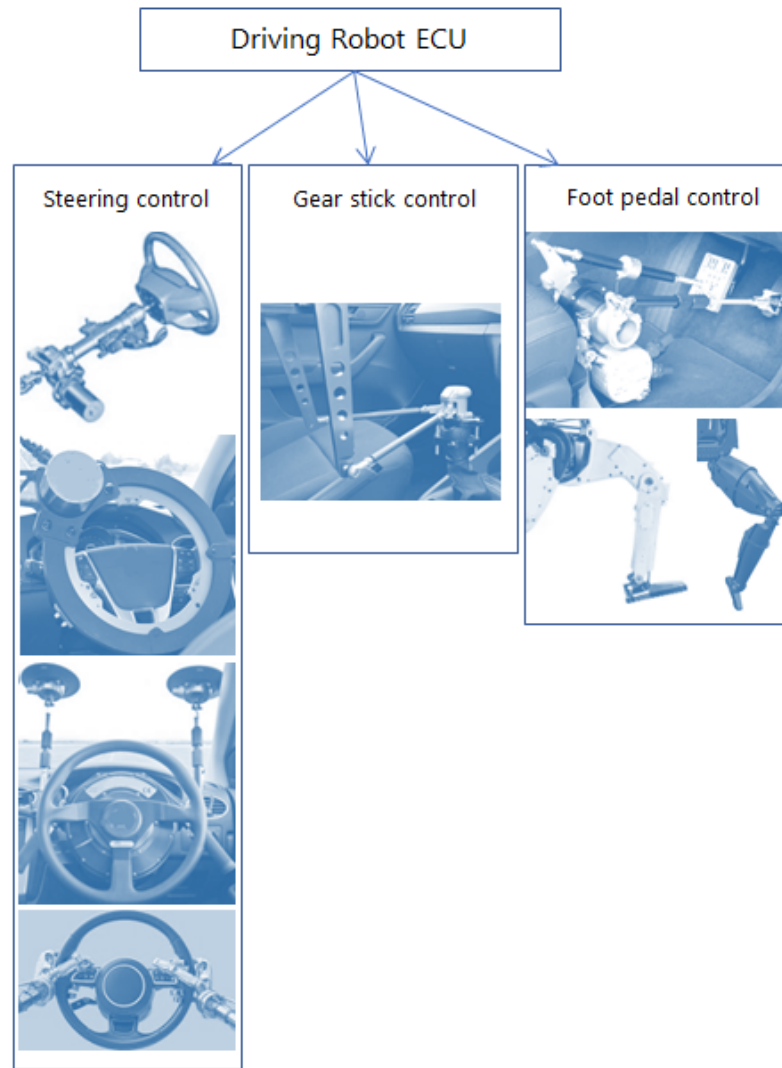


Figure 5: Mechanical actuator connection schema of the driving robot

### 2.3.6 Communications

Data and command transfer channels are critical life-line for this project. The communication interface enables us to have uninterrupted access to the vehicle to control real-time data.

The vehicle in focus has multiple channels of connection with the external world. The interfaces include the following:

- Multiple (at least 3) network connections
  - GSM for 4G/5G network coverage for drive-time data transfer
  - Different network service providers
  - The data transfer happens through all of the channels in parallel
  - Two channels transfer shared data, and one channel does duplicate transfer for critical data (up to 50MBps to be achieved)
  - Sensor raw data will be processed and compressed to reduce the size
  - The controller sends command data will with highest priority and redundancy
- NFC interface to process payment information
  - Interfaces to Bank server for processing payments

- Vehicle-to-vehicle connection interface
  - Direct and mobile network-based interface with the target vehicle
  - Collision avoidance cooperation with adaptive frequency hopping
  - Lane change assistance
  - Railway cross assistance
  - Priority assignments for emergency vehicles
  - For sharing local map data and traffic updates
  - IEEE 802.11p compatible 5.9GHz VANETs (Vehicular Ad-hoc Networks) known as WAVE (Wireless Access for Vehicular Environments, IEEE 1609) standards, defined by DSRC (Dedicated Short Range Communication consortium, SAE J2735, J2945)
- Vehicle-to-Infrastructure (RSU) connection interface
  - Direct and mobile network-based interface with target infrastructure
  - Tollgate communication, and payment through interaction with bank servers
  - Signal post communications to know the intended information – stop/go or speed limit, or left only, lane information
  - Bluetooth, WiFi, DSRC, short-range radio interfaces
- PC interface
  - Programming and Debugging interface - JTAG, USB, CAN, WiFi
  - User interaction (Display and Audio only)
  - Joystick interface
  - OBD interface
- Data storage interface
  - Event Data Recorder (aka black box)
  - Error history recorder
  - Backup images for the ECU and critical controllers
  - For collecting and analyzing the road data and building a central global map
- GPS receiver interface for location identification
- Law enforcement interface
  - Through mobile networks, in-case of any emergency or complaints
- Bank server interface
  - Through mobile networks, for any payment related transactions
  - Parking charges
  - Highway toll charges
  - Congestion charges, if any
- Cloud storage and cloud computing interface
  - To store the data related to the vehicle and running conditions
  - To process non-time-critical information to reduce time overheads on-board
- Command center interface
  - Through mobile networks
  - Firmware updates
  - Surrogate driving
  - Driving Robot parameter/health monitoring
  - Interact with the passengers

The contact between the driving robot and the command center is always maintained.

The following image shows the vehicle network with the external world.

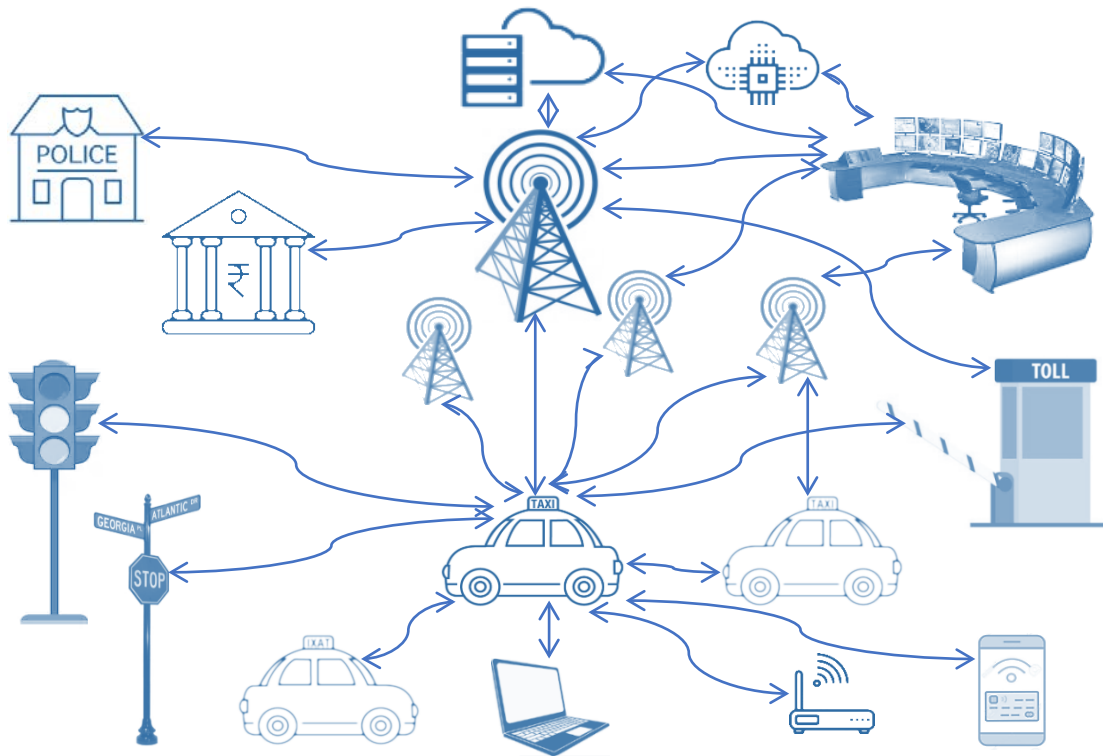
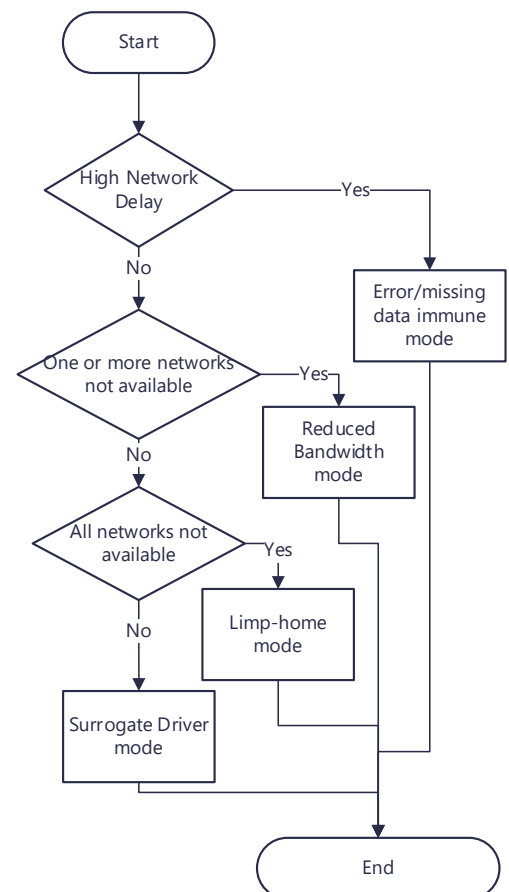


Figure 6: Vehicle connection network with the external world

Multiple eSIMs connect to different network operators for redundancy and reducing the risk of lower signal availability from one of the operators. The controller sends the data over all the possible channels for increased throughput.

Cameras send a large amount of data, and the controller has dedicated specialized high-speed interfaces connected to these sensors. MIPI CSI (or USB 3.0 Vision) is the standard interface used for smartphones and automotive applications for data rates up to 5Gbps (360Mbps). CoaXpress is a protocol (with speeds up to 12 Gbps) for bulk data transfer that is useful after extracting image data from multiple cameras. RADARs and LiDARs use UDP over Ethernet to avoid missing timelines. GPS and joystick work with the serial interface. IMU uses Ethernet or CAN or LIN. Vibration sensors, Voltage, and Current sensors interface to ADC ports. The mic connects to high-resolution ADC. UDS or OBD get implemented over CAN channels.

The output from the controllers is most likely digital signals or analog voltages or data sent over communication channels. These communication channels connect to the



camera modules, actuator modules (including control hardware), and few other active modules. The in-vehicle display gets driven through the HDMI interface.

The following image shows a proposed [43] functional architecture for the V2I interface. A similar interface on different vehicles enables them to communicate with each other.

The following image shows the network interface possibilities within the vehicle.

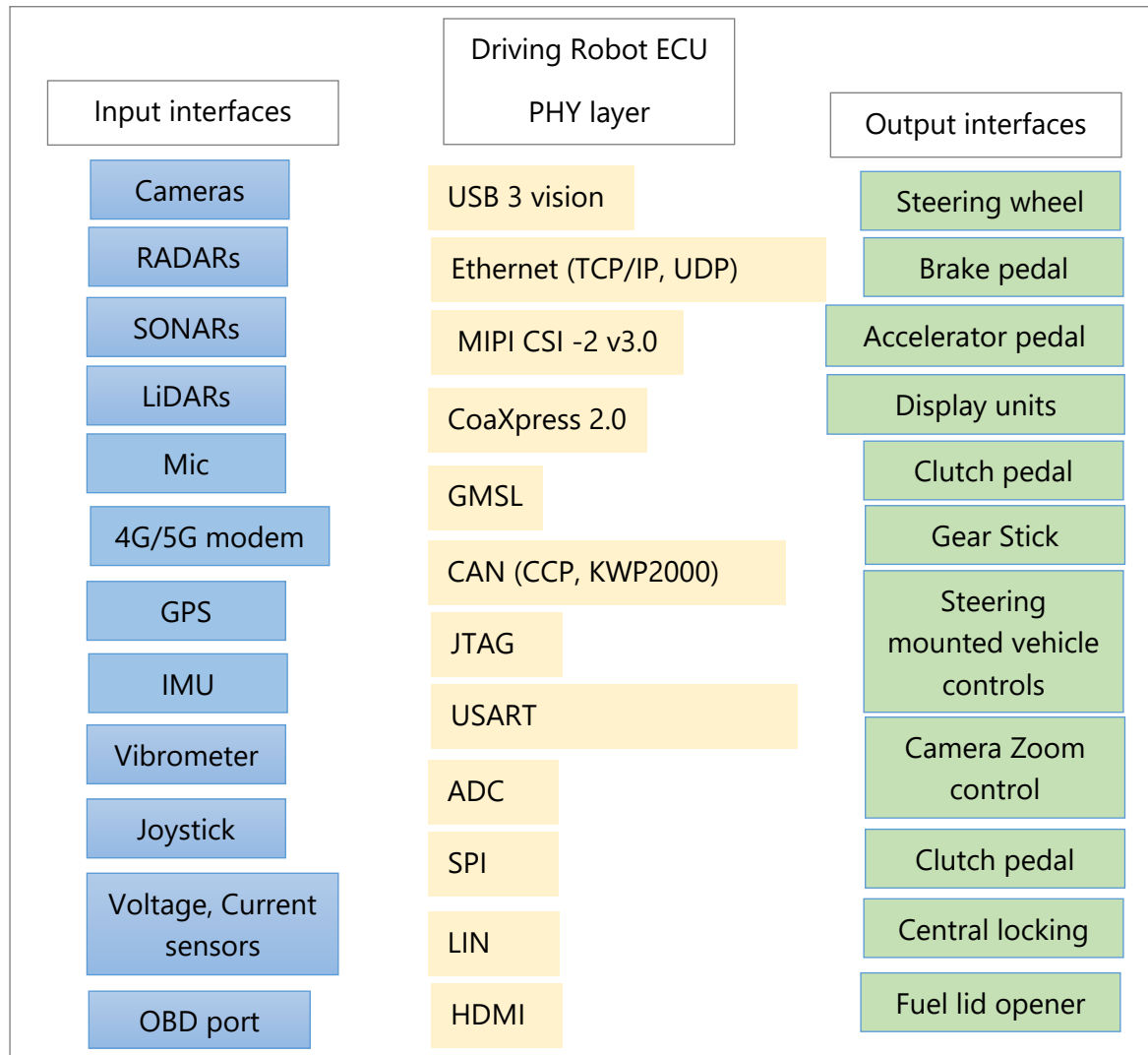


Figure 7: Connection network interfaces within the vehicle

### 2.3.7 Deep Learning Models

The sensors provide the controller with multiple data points to create a perception of the world around it. Each of the sensor data goes through a set of pre-processing operations for feature extraction to make the data ready for executing the deep learning models.

The sensor data with the extracted features will be passed to the deep learning model to get output. In figure 8, we see the architecture of a deep learning model to run a virtual car in a simulator only based on one camera input.



Cameras are usually placed all around the vehicle at convenient and stable locations. Sometimes, the visibility of the road and the objects can get skewed. Cameras' and RADARs' the vertical FoV is limited, objects close to the vehicle will have proper representation. In contrast, the objects far from the vehicle will have much error in their size, speed, position, and category estimation.

Layer (type)	Output Shape	Param #	Connected to
lambda_1 (Lambda)	(None, 160, 320, 3)	0	lambda_input_1[0][0]
cropping2d_1 (Cropping2D)	(None, 60, 320, 3)	0	lambda_1[0][0]
convolution2d_1 (Convolution2D)	(None, 30, 160, 24)	1824	cropping2d_1[0][0]
convolution2d_2 (Convolution2D)	(None, 15, 80, 24)	14424	convolution2d_1[0][0]
convolution2d_3 (Convolution2D)	(None, 8, 40, 24)	14424	convolution2d_2[0][0]
convolution2d_4 (Convolution2D)	(None, 8, 40, 24)	5208	convolution2d_3[0][0]
dropout_1 (Dropout)	(None, 8, 40, 24)	0	convolution2d_4[0][0]
convolution2d_5 (Convolution2D)	(None, 8, 40, 24)	5208	dropout_1[0][0]
dropout_2 (Dropout)	(None, 8, 40, 24)	0	convolution2d_5[0][0]
convolution2d_6 (Convolution2D)	(None, 8, 40, 24)	5208	dropout_2[0][0]
dropout_3 (Dropout)	(None, 8, 40, 24)	0	convolution2d_6[0][0]
flatten_1 (Flatten)	(None, 7680)	0	dropout_3[0][0]
dense_1 (Dense)	(None, 80)	614480	flatten_1[0][0]
dense_2 (Dense)	(None, 50)	4050	dense_1[0][0]
dense_3 (Dense)	(None, 30)	1530	dense_2[0][0]
dense_4 (Dense)	(None, 10)	310	dense_3[0][0]
dense_5 (Dense)	(None, 1)	11	dense_4[0][0]
Total params: 666,677			
Trainable params: 666,677			
Non-trainable params: 0			

Figure 8: A model architecture for a car running in a simulator using behavioral cloning

### 2.3.8 Actuator Control

We develop appropriate control systems for controlling the steering wheel, gear stick, and foot pedals. An essential aspect of this design is that the control loop has to be adaptive to different vehicle environments – actuator travel, force requirements, and movement restrictions. For example, the steering effort for few vehicles from OEMs like Ford may be lower than those from OEMs like TATA. The clutch force will be different in each vehicle. Clutch travel will be different for vehicles depending on the design of the clutch system.

Deterministic robust control system (DRC) implementations help achieve the control system stability when we deal with various (changing) parameters in due course of the usage in a vehicle. Some of these parameters are

- Acceleration changes with different occupancy in the vehicle or different road slopes
- Brake force variations in the pre-ABS mode of a brake to suit the road conditions
- Clutch force variations with clutch wear
- Steering & accelerator control when the wheel lifts off from the road because of a bump or pothole

Adaptive robust control system (ARC) implementations help in transitioning between vehicles with different (time-varying) drive controls – manual transmission to automatic transmission, or gear stick's position changes

The general schema for the ARC looks like below.

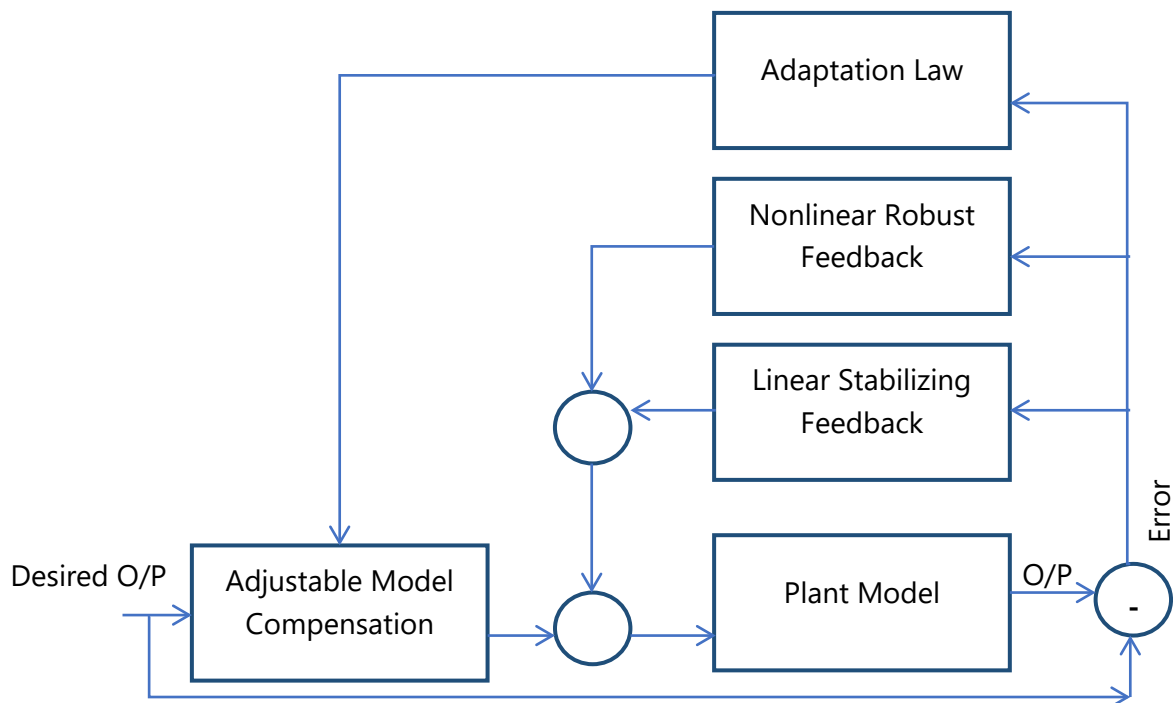
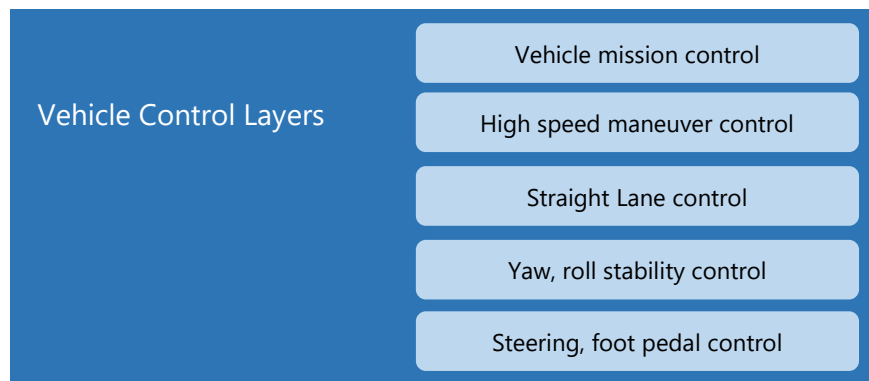


Figure 9: Block diagram of an Adaptive Robust Control System (ARC)

However, there are many layers of control architecture, starting from actuator control to the complete vehicle stability control.



### 2.3.9 Sensor Network

We will primarily depend on the Camera and RADAR systems and depend less (only during development) on the LiDAR systems. All the sensors get mounted on the roof of the vehicle as a module. The mounting bar will be 1 meter long and sufficiently wide platform rigidly fixed to the vehicle.

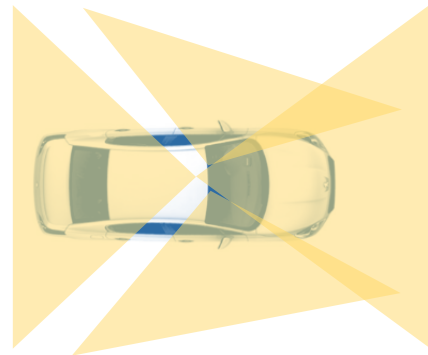


Figure 10: SONAR mounting schema

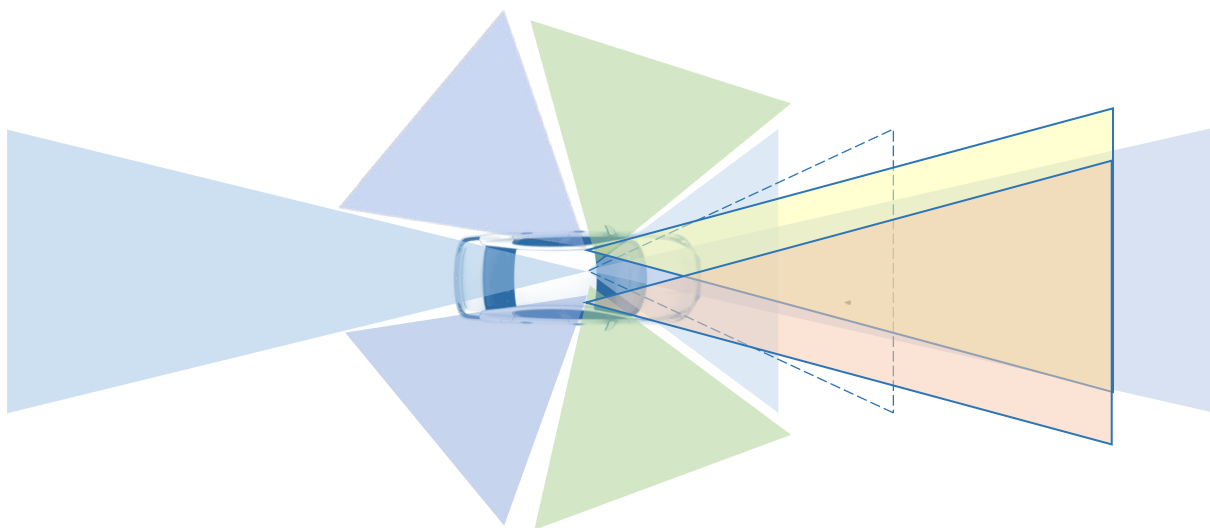


Figure 11: Camera mounting schema

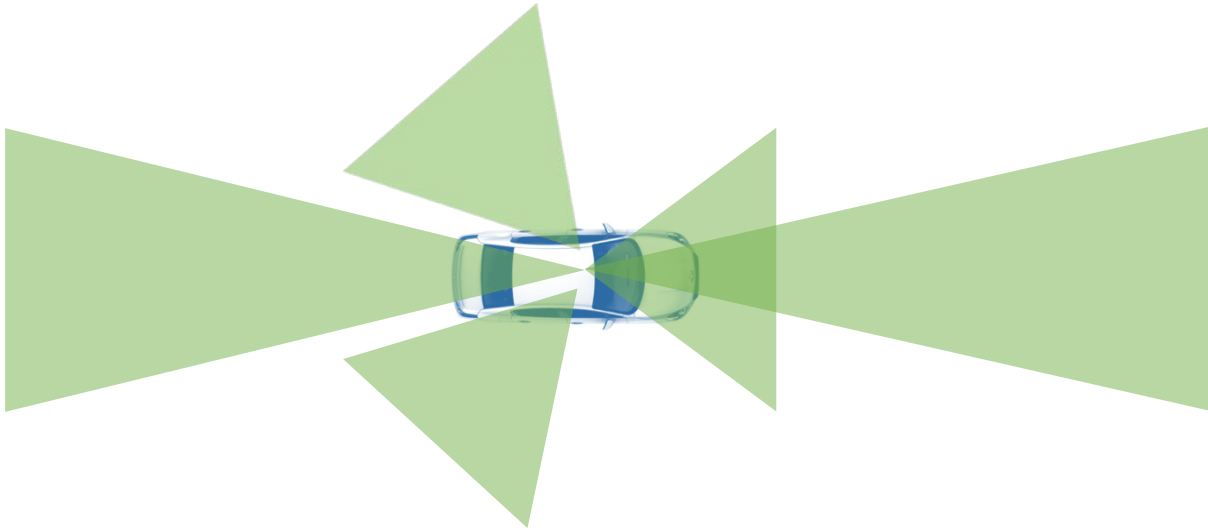


Figure 12: RADAR mounting schema

Cameras will be the primary input for creating the perception of the world. The vehicle has three wide-FoV cameras (one in the front, 2 in the sides facing rear), and an optional two wide-FoV cameras (on the sides facing sideways in the green shade), two narrow-FoV long-range cameras (one in the front, one in the rear), one electronic zoom lens (facing front, in dotted line), a pair of cameras mounted at the ends of the roof-top platform forming stereo camera system( both facing forward, in yellow and red shades). Two cameras are optional because the event occurrence from this angle is less than that in other directions.

The vehicle has 3 SRR and 2 LRR type RADARs. Additionally, the vehicle also comes with 4 SONARs for additional safety and full surrounding coverage. IMU, Mic, Vibrometer, and other sensors will be in the driver seat along with the driving robot.

The following figure gives a comparison of LiDAR, RADAR, and Camera systems

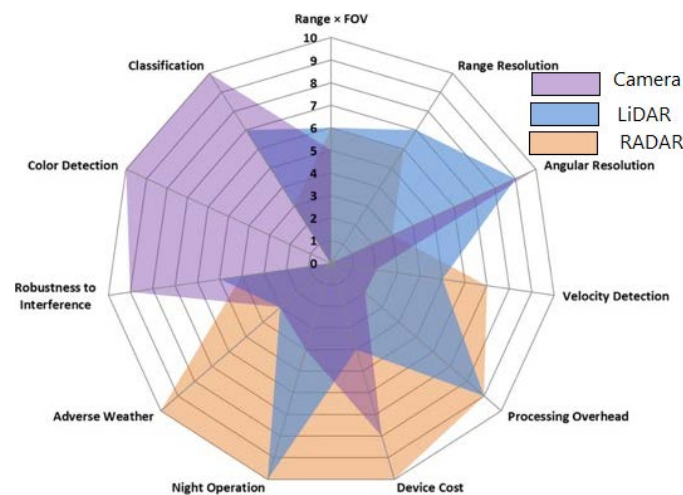


Figure 13: Comparison of the critical sensor options [55]

We can infer from this that a combination of RADAR and Camera can result in compensating the lack of LiDARs.

## 3 GEN-1 TENTATIVE SPECIFICATIONS

---

The following is a possible list of specifications for the parts and the product. This list is hypothetical and will improve in due course of development.

### 3.1 AUTONOMOUS DRIVING ROBOT

The generation-1 deliverable will have the following configuration

- Suitable for four-wheeled vehicles of any make
  - L5, L6, L7, M1, and N1 categories of UNECE R.E.3 rev.6 under section 2
  - Only automatic transmission supported
    - Includes electric vehicles with no clutch configuration
  - Independent of fuel type
    - Diesel or Petrol
  - Brake assist only supported
    - Hydraulic or electric-assist recommended
  - Power steering only supported
  - Active safety features are not necessary
  - Independent of vehicle dimensions
    - Within the recommended lengths of corresponding categories
  - Independent of vehicle interior layout for controls
  - Independent of seating capacity
    - Within the recommended load range
    - Without any structural modifications
- Independent network connections with 4G operators
  - At least three embedded SIMs to be activated
- Always-on connection with the command center
  - Possibility to request surrogate driver feature
- Possible to override the mechanical controls manually
- Driving Robot is safe to operate in the following conditions
  - Maximum speed limited to 35kmph
  - Under reasonable lighting conditions
  - With proper road signage (no wrong markings) - markings for lanes, zebra crossings, signal crossing
  - No unexpected cross-traffic or high-speed traffic
  - Road condition is not a significant concern including potholes and speed breakers
  - Within the vehicle's recommended loading capacity
  - Road turning radius is not a problem, but no tight spaces
  - Non-slippery road without steep slopes
- The vehicle can do the following on its own, under observation by the experts
  - Start from rest and accelerate safely
  - Overtake slow-moving vehicles on the way
  - Change lanes including indicator signals

- Blow the horn to alert other road users
- Brake safely and smoothly under typical situations
- Brake swiftly to avoid bumping into obstacles and road objects
- Read road signals and act accordingly
  - One way only, stop at the red signal, go at the green signal, and yield
- Follow verbal instructions about the destination from the users
- Arrive at the location upon receipt of user request
- Arrive at the fuel station when the fuel levels are below a threshold
- Access the map servers to identify the route

## 3.2 CAMERAS

Long-range cameras (Front-facing and rear-facing) configuration

- Higher than 15MP still resolution
- Higher than 10MP video resolution @ 60fps
- A small aperture (f/11) for long-range visibility
- CIS (CMOS Image Sensor) element with BSI (Back Side Illumination)
- MIPI CIS-2 v3.0 data transfer support
- Fixed focus optics ( 1m to infinity)
- GS (Global Shutter) support in the firmware
- Horizontal & vertical FoV up to +/- 4 degrees
- Mounted on the center of rooftop mounting platform
- Power consumption < 5W
- Operating temperature range -15 to +80 degrees C

Short-range wide-view cameras (front-facing, two {and optional two} side-rear facing)

- Similar to Long-range cameras, except
  - A large aperture (f/1.5) for long-range visibility
  - Fixed focus optics ( 0.1m to infinity)
  - Horizontal & vertical FoV up to +/- 60 degrees

Mid-range zoom lens cameras (front-facing)

- Similar to long-range cameras, except
  - Higher than 10MP still resolution
  - Higher than 6MP video resolution @ 60fps
  - Up to 8X optical zoom with electrowetting lens
  - Power consumption < 8W
  - Variable FoV due to the zoom

Stereo cameras (front-facing)

- Two cameras, 1meter apart, mounted on the rooftop platform with their axes parallel
- Similar to long-range cameras, except
  - Medium aperture (f/2.5) for medium-range visibility
  - Horizontal & vertical FoV up to +/- 30 degrees

Each camera module has an independent processing unit (Raspberry Pi unit with Tensor Flow Lite) for essential object detection and tracking. This information flows to the driving robot ECU along with the camera data.

Cameras' data will be converted to JPEG (or MPEG) format and sent to the driving robot for saving data bandwidth and reduce data transmission time.

### 3.3 RADARs

LRR (Long Range RADAR) (front-facing and rear-facing) configuration:

- 76GHz - 77GHz band
- 1m - 250m range
- 55 dBm power transmit limit
- 0.1m distance accuracy
- 0.1m/s velocity accuracy
- Up to 20 object detection at a time
- Ghost image elimination - in the firmware
- Elimination of interference from surrounding RADARs – in the firmware

SRR (Short Range RADAR) (front-facing and two rear/side-facing) configuration:

- Similar to LRR, except
  - 77GHz - 81GHz band
  - 0.1m - 50m range
  - 0.02m distance accuracy

### 3.4 OTHER SENSORS

IMU configuration:

- Accelerometer range is +/- 18g
  - 3-axes configuration
- Gyro range is +/- 500 degree/second
  - x, y, z angular rate sensors
- The sampling frequency is up to 200Hz
- The temperature sensing range is -30 to 100 degrees C
- 16-bit data output from the sensors on the I2C bus at 400kHz

Microphone configuration

- 200Hz to 14000Hz
- Minimum SNR 64dB
- Sensitivity is -26 dBFS
- Temperature range is -30 to +85 degrees C
- The output format is an analog signal

GPS sensor configuration

- Anti-jamming technology included
- 12 channel receiver
- Receiver sensitivity is at least -165 dBW
- Operating temperature is -30 to +85 degrees C
- Up to 5Hz sampling rate

### 3.5 ELECTRONIC CONTROL UNIT

ECU is a central monitoring unit that does only the most important computations. The driving robot has a distributed computing architecture to reduce the cost. Otherwise, we would have to go with tailored SoC such as Nvidia Drive AGX Xavier. This approach increases cost dramatically.

All the sensors have an interface with local processors (cameras interface with a Raspberry Pi for object detection, independent of main ECU) to have faster and parallel detection. The results of these detections reach the main ECU as well.

When the internal computations are slower, this data will be a backup for making decisions.

The following is only indicative and, by any means, not exhaustive. ECU comes with the following configuration

- Microcontroller (Primary and backup)
  - 32-bit processor with ARM Cortex-R architecture or equivalent
  - Multicore processor (at least 8) with 1GHz or higher
  - 2TB Flash with ECC, external memory support included
  - 32GB RAM with ECC
  - DMA, secure flash, a secure boot loader
  - ADCs with 16-bit or higher resolution
  - Support for Ethernet, USB 3.0, CAN, DACs, SPI, I2C, 4G/5G modems
  - Up to 10 camera interfaces, four radar interfaces
- GPU
  - Multiple GPU units each with 500+ cores that run deep learning models
  - 20 TOPS and more processing power
  - Some are interfaced with the sensors directly, while some with microcontroller
- Wiring harness interface
  - For controlling the signaling functionality to indicate to the other road users

Embedded Software has the following configuration

- RTOS based on 32-bit Linux
- Preemption enabled
- Supports the data flow architecture among the sensors, processors, and GPU.
- OTA support enabled
- Includes encryption algorithms
- HAL heavily utilized to reduce the changes required with hardware alterations
- Compiled with optimization for best code run times



## Communication network

- Low latency channels
- 4G LTE connectivity through eSIM interface
- 5G interface
- In-vehicle Ethernet for bulk data transfer
  - from all the processors near the sensors to the primary processor
- MIPI channels for camera data transfer quickly
- CAN 2.0b network for sending quick messages and commands
- V2X interface
  - IEEE 802.11p compatible 5.9GHz VANETs known as WAVE (IEEE 1609) standards, defined by DSRC (SAE J2735, J2945)
- NFC, USB, WiFi interfaces

## 3.6 ACTUATORS

The robotic limbs have the following specifications. All the following are written only qualitatively since the exact requirements will be known when the design starts.

### Robotic arm

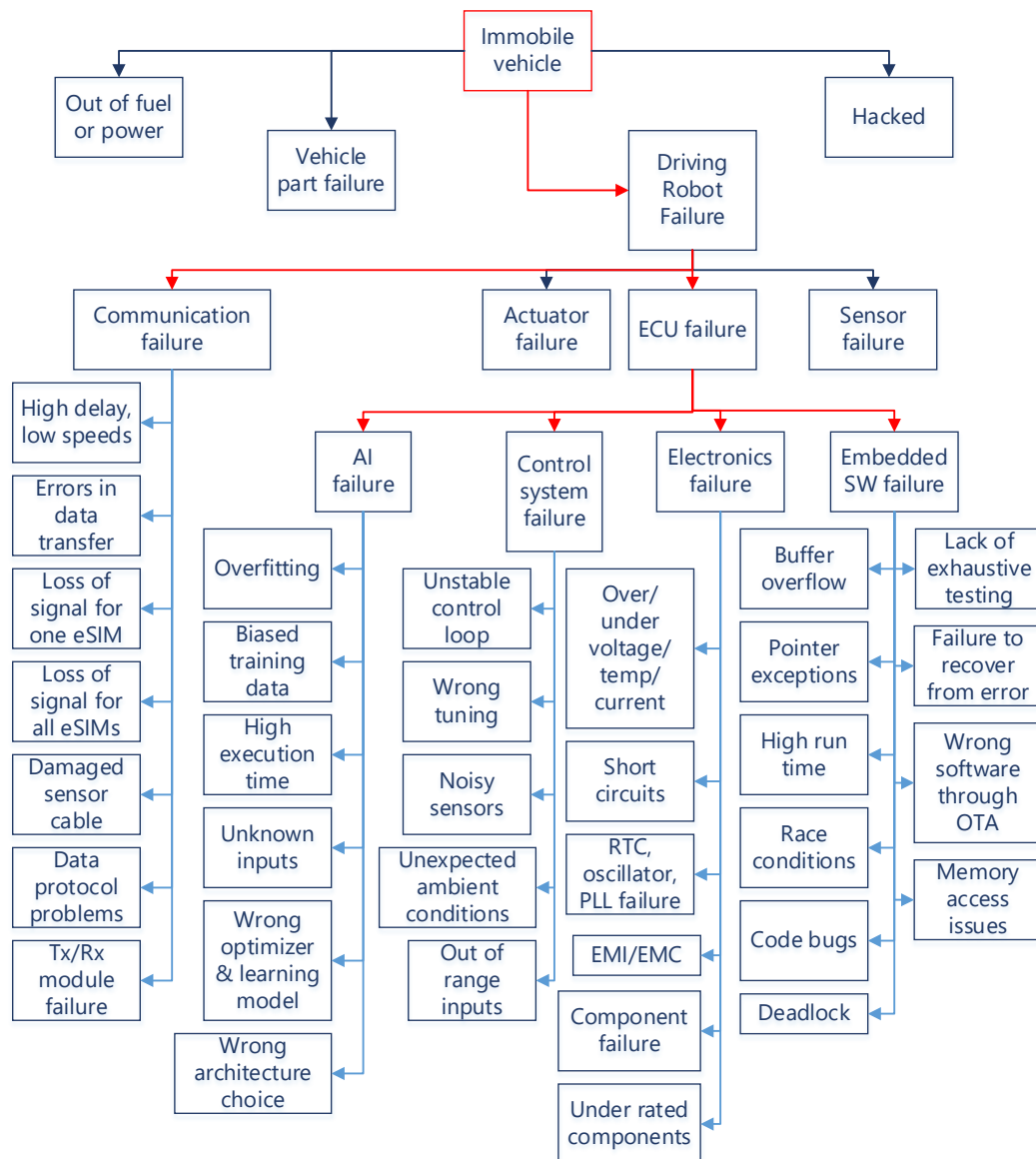
- Sufficient payload to operate the steering wheel even during the assist failure
- Enough DoFs to
  - Turn the steering wheel till extremes
  - Operate the controls on the steering wheel and dashboard
  - Operate the gearstick to change the gears, synchronizing with the clutch
- Quick response and execution time within the human operable levels
- Endurance for long hours of operation
- Compact, and folding into a smaller unit when turned off
- Internal sensors to monitor the position and component health
- An internal battery pack that functions when the vehicle power supply fails

### Robotic leg

- Sufficient payload to operate the foot pedals
- Enough DoFs to change between clutch and brake, similar to the human drivers
- Quick response and execution time within the human operable levels
- Endurance for long hours of operation
- Internal sensors to monitor the position and component health
- An internal battery pack that functions when the vehicle power supply fails

## 4 FAILURE TREE

The following failure tree represents some of the failures that the driving robot could face. Only a few sections marked in red arrows are elaborated. All these failures will be covered under DFMEA and are handled in the product without the loss of functionality.



## 5 CHALLENGES IN EXECUTION

The project implementation is not trivial and requires very different approaches from standard practices. The implementation also requires many choices and compromises to be done. A detailed list of choices and preferences are in annexure-1.

## 5.1 COMMUNICATION NETWORK BOTTLENECKS

For implementing the surrogate driver mode, we primarily depend on the mobile network. There are some challenges with this implementation.

### 5.1.1 Network Coverage and Speed

The driving robot should operate well in all the locations, even when the network coverage from the service provider is not reliable (reduced data speeds).

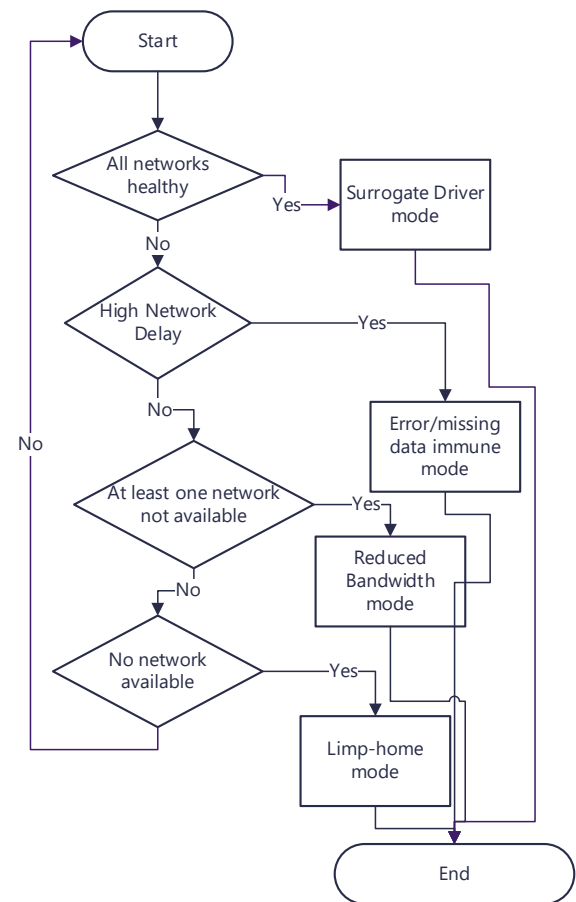
We addressed this issue by proposing multiple parallel (all active) interfaces that connect to (preferably) different network service providers. Redundancy is to make sure that at least one service provider will be available for supporting the data and command transfer. Our data-intensive transmission happens for the camera data. All other sensors' data size is small compared to the camera data.

When all the networks are available, we split the data and send them through the parallel channels for faster transfer.

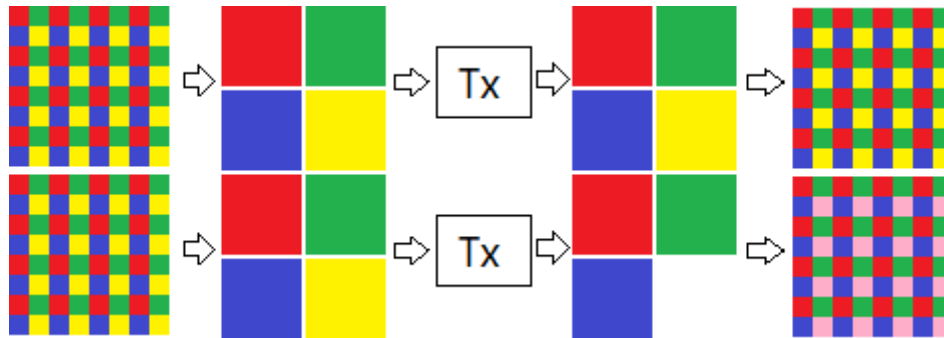
### 5.1.2 Network Delay

When the data goes through the network in parallel paths, we cannot be sure of the order of arrival of the samples. If one network channel has more delay than the other, we will probably be looking at a full image constituted of different time samples.

- We include a timestamp for all the data packets sent across the networks. The receiver side makes sure to keep all the timestamps in order and only patch up those data packets corresponding to the same time. It usually considers the most recent timestamp for decisions, but also considers earlier timestamp samples to identify events from the changed pixels and when the current pixels are missing.
  - If the data for the same time is missing, other redundant sensor's information is given temporary priority to bypass this data.
- The command center maintains a delay threshold. If a packet corresponding to a given time arrives later than the threshold, it will get ignored, and the decision will be taken based on the known (already arrived) packets only. Repeated instances of missed packets will mark the channel as unavailable for a small predefined time duration, and it will be back in the pool of available channels. During the time it is marked unavailable, the bandwidth conservation steps mentioned in the following subsection will be helpful.
- The image data from a given time-sample will be chunked into smaller packets for parallel distribution by sampling the pixels at regular intervals and not by dividing the full image into large segments of packets. With this approach, the loss of one or more of the data packets (from one channel) will only result in a pixelated image.



- We will also work on creating a deep learning model that is tolerant of information loss at smaller parts of the image.
- The following image shows the color-coded allocation of the pixels to different network channels. If all the channels have the same negligible delay, we reconstruct the full image. If one of the channels have more delay and missed during the time threshold, we will fill the missing pixel value (pink color) with the average pixel values from the neighboring pixels.

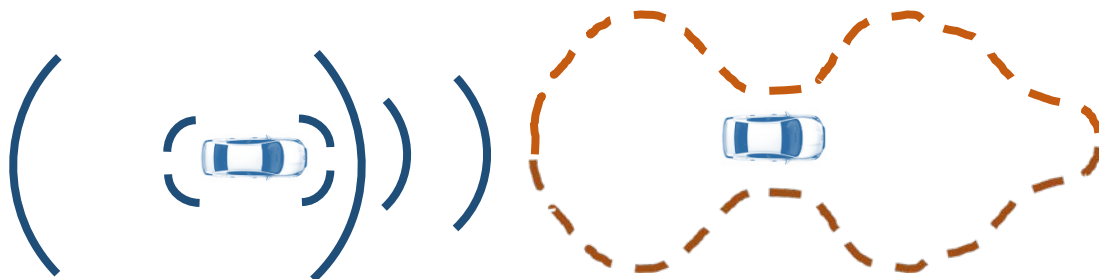


- Reduce the data density (lower quality image data with down-sampling)

### 5.1.3 Missing Network Channel

When at least one network is not available, or the delays are beyond the threshold, we do at least one of the following.

- Eliminate redundant and overlapping data. The following images show the surrounding views available to the driving robot from different cameras. To reduce the bandwidth, we send the stitched images for the driving robot to make decisions
- Compress the compressed data, if the (de)compression delay is better than the uncompressed data transfer



- Send the sparse matrices with only the changed pixel data from the previous image
- Use the principles of steganography to overlap images and send the information without missing too much
- Send only the abstract information about the surrounding objects with distance, size, and velocity information
  - The command center operator only sees cuboids (stationary or moving) based on the 3D model information shared
- Send information about the surroundings as configuration data for the object models known to the graphics processor at the command center

- The command center's 3D surrounding model has in-built pre-programmed objects that the vehicle is likely to encounter in regular rides
- Unknown objects will get approximated to known objects of similar size

#### 5.1.4 Lost Communication

When the vehicle connectivity with the command is unavailable in all the parallel network channels, the vehicle identifies a low-risk spot nearby. It safely parks itself till the connection is established, or some authorized user from within the vehicle overrides the vehicle control manually to move the vehicle. If such a safe spot is not available, the vehicle continues to move using its basic autonomy features at reduced speeds. Also, it indicates to the surrounding vehicles by flashing hazard lamps.

This situation is very similar to the loss of communication with space vehicles such as Mars Rover or Lunar Rover or artificial satellites, or airborne vehicles such as drones.

Space vehicles are equipped with essential autonomy to handle obstacles and falls. Likewise, modern drones have RTH (Return-To-Home) feature where it tracks its path and traces it back if it lost communication with the control unit. Additionally, these drones sometimes come with essential autonomy to handle avoid crashing into obstacles.

## 5.2 CONTEXT IDENTIFICATION FOR ROAD SYMBOLS



Figure 14: Stop sign in different styles

Understanding and recognizing the traffic-signs is not a tough task in terms of building the model. However, understanding the context of the traffic sign is very important to make a decision. For example, the stop traffic sign in different contexts as below needs different actions for the vehicle. The following shows the stop sign in different ways.

The context of the symbol makes a huge difference. A stop sign on a toll gate barrier means "stop" when the signboard is vertical, and it means "go" when the signboard is tilted (no

barrier). The stop sign on school bus doors means – stop when the door is open, ignore when the door is not open. The stop sign can sometimes be made invisible by tree branches, or birds, or even other vehicles.

The deep learning model has to adapt to these contexts and variations of the road sign representations also. To do that, we caption with each image with more than one keyword to give a full description of all the other objects, background, and orientation. A sufficiently trained deep learning model with more image tags will be able to identify the context.

### 5.3 VIBRATION AND TILT

The sensors are mounted at a fixed location on the vehicle. The real road conditions may lead to several unexpected failure modes.

- The cameras experience vibration from the engine or suspension effects. This vibration results in noisy images leading to the non-detection of objects.
  - We will make sure that the vibrations are filtered out by mounting the sensors rigidly on the platform while the high-frequency vibrations are eliminated by using appropriate dampers.
  - We will also make sure that the cameras' image sampling rate is high, and they use GS (Global Shutter) technology for blur-free images.
- When the vehicle is on an inclined road, or over a bump, or a pothole, the road view may not be available fully. We will make sure to train our deep learning models to identify the tilted objects and partially visible objects to handle such situations.

### 5.4 CONCEALED OBJECTS

We will make sure that the missing objects (in the present view of the sensors) are updated in the 3D model from the probabilistic models.



- When the vehicle is behind another vehicle, the driving robot will not be able to identify the road information and other obstacle presence. The same situation arises when an object is in the blind spot. We will make sure that our road model includes past information and the predicted location of all the objects at present. By doing so, all the visible objects' probabilities are set to 1, and concealed objects' probabilities are set to the calculated values.
  - With the concealed objects' probabilistic locations identified, the driving robot will make sure to avoid these regions during the path planning.



- When the vehicle is behind another vehicle, the driving robot also cannot see the road condition (potholes, objects blocking the road, situations requiring emergency braking or steering away) ahead. When the driving robot identifies the abnormally steering vehicle at the front, it applies the vehicle brakes and also cautiously follows the vehicle, similar to how a person driver would react.

## 5.5 MISSING ROAD MARKINGS AND BORDERS



Figure 15: Range of path markings and the types in India

The road conditions in India range from no tarmac to 12-lane roads. It is necessary to identify the road boundaries in all the conditions. We can use the mathematical models and few lane-marking algorithms to identify the road boundaries when the roads are good. When they are not available or when the roads are blocked by accidents or by stray animals, or when the road is waterlogged, we will do one of the following.

- Follow the traffic. The driving robot identifies the drivable portion of the road by observing the paths of the vehicles ahead of it. The driving robot learns the safe path by observing several vehicles. The internal local map is updated with this information and issues navigation decisions correspondingly.
  - Under some circumstances, vehicles go through a smaller pothole to avoid a larger obstacle. The driving robot identifies such behavior and does the same to steer out of the obstacle, which it would not have done otherwise.
- The driving robot also identifies the road by color. In some cases, the roads are laid but left without any markings. We can easily differentiate the road from the surroundings by the long stretch of similar color.
- The driving robot also looks for relatively flat regions to mark them as drivable area ahead. Such roads are common in non-urban regions where roads are sometimes made with cement, granites, and stones. The stereo camera provides the necessary depth information. Additionally, the driving robot looks for shadow information based on the position of the Sun to identify the unpaved roads.

## 5.6 UNSTRUCTURED TRAFFIC AND HAND GESTURES



*Figure 16: Variety of traffic conditions in India*

Indian traffic conditions are highly unpredictable and unstructured in most cases. The variation in the traffic conditions is huge.

- We will handle vehicle dynamics with extreme care ensuring that the vehicle behaves well on the road under any circumstances and does not bump into other vehicles
- When we fix the driving robot in the vehicle, we program the driving robot with the vehicle parameters, such as geometry and turning radius. This data helps the vehicle find its position and assess if the space available is sufficient to go through or not.
- We need to prioritize the sensors differently for low-speed operation. The camera feed from the near-front and side cameras is given more priority during transfer.
- When our vehicle bumps into another vehicle or an obstacle, the IMU data lets the driving robot know. The driving robot then makes the necessary corrections to the path to come out of the situation.
- In addition to the traffic conditions, few vehicle drivers give hand signals instead of the vehicle-mounted light signals for turning or braking. In some circumstances, people from cross-traffic will request to pass through gestures and expressions. These situations are tricky to handle for the driving robot for identifying the intent of the vehicle/driver. However, the blocked path will bring the vehicle to a halt.
- Cross-traffic negotiation is very subjective and person dependent. The driving robot cannot be aggressive and assume that the cross traffic would stop. In a selfish environment, the cross-traffic purposefully drives fast so that the autonomous vehicle skips negotiation and waits. This wait could be longer than that in structured traffic.
- Pedestrian negotiation is a tricky subject. Road users sometimes cross the road on foot at places without zebra crossing, most likely underestimating the oncoming traffic. The driving robot does not try to negotiate with the pedestrians but waits.



## 6 SCOPE OF WORK

---

As seen in the background, there has been a lot of active research going on in the field of autonomous vehicle development. The independent full autonomous (ego-only) vehicle with self-awareness is still in development.

End-to-end driving (direct perception) generates motion directly from the sensor inputs. Popular approaches to this method are direct supervised deep learning, neuroevolution, and deep reinforcement learning. This method is a single deep learning model with several layers acting directly upon the actuators through control loops (perhaps).

The modular system is where the actuators are driven by a pipeline of smaller components, each solving smaller problems using known principles from robotics, computer vision, vehicle dynamics. It is possible to add rule-based safety on top of a deep-learning module in this approach, increasing the system reliability. This approach also creates an easier way to define tasks for individual (remote) teams.

Some of the modules of the driving robot and their primary deliverable are given below.

- Vehicle preparation
  - Integrate the actuators, sensors, custom chair in the vehicle
  - Easy removal, and poka-yoke
  - Vibration removal at sensor mounting locations
  - Manual override for mechanical system
  - Display unit inside the vehicle
- Actuator development
  - Start with linear and rotary actuators for quick initial implementation
  - Create robotic limbs with the required DoFs in 2 years
  - Joystick development for wireless vehicle control
- Perception (sensor) systems and networking
  - Develop a mounting platform
  - Mount sensors with required protections
  - Establish reliable communication interface with the sensors
- Sensor Fusion
  - Stitch image data for surround view
  - Validate data
  - Eliminate data duplicates
- Deep learning models
  - Architect and create the models
  - Prepare a training set and test set
  - Train and validate on real data
- Object detection
  - Identify various types of objects in view
  - Semantic segmentation
  - Context extraction
  - Road lanes identification

- Road object model
  - Prediction of future positions
  - A probabilistic model for all required objects nearby
- Motion Planning
  - Route planning with map integration
  - Identify free space for motion
  - Path planning
  - Obstacle avoiding maneuver planning
- Network communication module
  - Create a 4G/5G network access infrastructure
  - Establish a failsafe mechanism to compensate for the loss of network
  - Pack & unpack data protocols
  - Plan transferring data through multiple network channels
  - OTA infrastructure
- Actuator & Vehicle control system
  - Create a high fidelity actuator & vehicle models and validate
  - Simulate the models along with controller models and tune optimally
  - Implement ARC (Adaptive Robust Controller) to suit changing vehicles, parameters, and drive conditions
- Localization & Mapping
  - SLAM (Simultaneous Localization And Mapping) implementation
  - Map server integration
  - Local 2D map generation from cameras
- ECU
  - Process all the signals from vehicle sensors
  - HF region for 4G/5G modules
  - Communication modules PHY implementation
  - GPU integration
  - All necessary protections
- Embedded Software
  - RTOS with OTA support, the full driver stack
  - Integrated security with the application
  - A secure 2-stage boot loader
  - DL model integration to GPU
  - MBD with auto-code generation support
- Emergency response module
  - Software part checking for fault with sensors and actuators
  - Algorithms to check for probability of a crash
  - Algorithms to react and control actuators
  - Algorithms to handle communication module to send messages (V2X or over the network) for alerting necessary units

Some of the modules of the command center and their primary deliverable are given below.

- Communication Infrastructure
  - Required hardware, workstations, and software to support the data exchange
  - Receive multiple channels of 4G/5G data
  - Sort based on the timestamp and create the full picture
- 3D visualization setup
  - VR goggles and corresponding setup
  - Model rendering based on the sensor feedback in real-time
- Drive emulation setup
  - The steering wheel and foot pedals with force feedback
  - Networked system with the support for required protocols
- AI Backend
  - A scalable AI for all the support operations
  - Augment the commands sent by the surrogate drivers to ensure safety
- Data Analytics
  - Storage and retrieval system for the vehicle data including the user inputs
  - Training modules that continue to learn based on new inputs
  - Deployment center to send updates

## REFERENCES

---

- [1] "Automated Driving in its Social, Historical and Cultural Contexts" by *Markus Maurer et al (Eds.) (2016)*. Autonomous Driving: Technical, Legal, and Social Aspects. *Springer*. pp. 41–68 (p. 48). ISBN 978-3-662-48847-8
- [2] <https://web.archive.org/web/20120110042753/http://www.da-vinci-inventions.com/self-propelled-cart.aspx>
- [3] [https://en.wikipedia.org/wiki/Houdina\\_Radio\\_Control](https://en.wikipedia.org/wiki/Houdina_Radio_Control)
- [4] <https://web.stanford.edu/~learnest/sail/oldcart.html>
- [5] "Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles", SAE standard J3016
- [6] <https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety>
- [7] <https://www.outsight.tech/>
- [8] <https://www.stereolabs.com/>
- [9] <https://www.flir.com/iis/machine-vision/stereo-vision>
- [10] <https://nerian.com/products/scenescan-stereo-vision/>
- [11] <https://www.framos.com/us/framos-depth-camera-d435e-camera-only-22806>
- [12] <https://www.optotune.com/products/focus-tunable-lenses/electrical-lens-el-16-40-tc>
- [13] <https://webviz.io/worldview/#/>
- [14] <http://carla.org>
- [15] <https://www.intelrealsense.com/>
- [16] [https://www.photonics.com/Articles/Focus-Tunable\\_Lenses\\_Enable\\_3-D\\_Microscopy/a57323](https://www.photonics.com/Articles/Focus-Tunable_Lenses_Enable_3-D_Microscopy/a57323)
- [17] <https://diffratec.at/>
- [18] <https://www.edmundoptics.com/knowledge-center/application-notes/imaging/liquid-lenses-in-imaging/>
- [19] <https://www.corning.com/in/en/innovation/corning-emerging-innovations/corning-varioptic-lenses.html>
- [20] <http://rll.berkeley.edu/blue/>
- [21] <https://arxiv.org/pdf/1904.03815.pdf>
- [22] <https://www.igus.in/roboLink/robot>
- [23] <https://www.starsky.io/>

- [24] <https://www.nrec.ri.cmu.edu/solutions/defense/other-projects/enhanced-teleoperation.html>
- [25] <https://www.forbes.com/sites/richardbishop1/2020/03/24/starsky-robotics-failed-does-that-mean-automated-trucking-is-dead/amp/>
- [26] <https://medium.com/starsky-robotics-blog/the-end-of-starsky-robotics-acb8a6a8a5f5>
- [27] <https://medium.com/aurora-blog/teleassist-how-humans-collaborate-with-the-aurora-driver-a8b3529fb937>
- [28] <https://aurora.tech/technology>
- [29] <https://www.mouser.in/applications/autonomous-car-sensors-drive-performance/>
- [30] <http://www.ti.com/lit/wp/slyy150/slyy150.pdf>
- [31] <https://www.microwavejournal.com/articles/33705-comprehensive-survey-of-77-79-ghz-automotive-radar-companies-sensors-and-ics>
- [32] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6832580/>
- [33] <https://www.lapinamk.fi/loader.aspx?id=983f39aa-c97f-4f2d-bb39-abe006c6bad3>
- [34] <http://www.cimms.ou.edu/~lakshman/Papers/radarcompression.pdf>
- [35] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4732078/>
- [36] <http://liu.diva-portal.org/smash/get/diva2:328101/FULLTEXT02.pdf>
- [37] <https://www.cs.ru.nl/E.Poll/papers/2018wasa.pdf>
- [38] <https://arxiv.org/pdf/2003.07191.pdf>
- [39] <https://vtechworks.lib.vt.edu/bitstream/handle/10919/36580/etd.pdf>
- [40] <https://www.latimes.com/business/story/2020-02-24/autopilot-data-secrecy>
- [41] <https://arxiv.org/pdf/1906.09918.pdf>
- [42] [http://www.fengzhao.com/pubs/yang\\_x\\_v2v.pdf](http://www.fengzhao.com/pubs/yang_x_v2v.pdf)
- [43] <https://rosap.ntl.bts.gov/view/dot/3592>
- [44] [http://transops.s3.amazonaws.com/uploaded\\_files/V2I%20DC%20Tech%20Memo%204-FINAL%20V1.2%20011017.pdf](http://transops.s3.amazonaws.com/uploaded_files/V2I%20DC%20Tech%20Memo%204-FINAL%20V1.2%20011017.pdf)
- [45] <http://local.iteris.com/cvria/html/standards/cvstandards.html>
- [46] <http://local.iteris.com/cvria/html/applications/applications.html>
- [47] [http://www.mogi.bme.hu/TAMOP/jarmurendszerek\\_iranyitasa\\_angol/math-ch09.html](http://www.mogi.bme.hu/TAMOP/jarmurendszerek_iranyitasa_angol/math-ch09.html)
- [48] <https://www.stemmer-imaging.com/en/knowledge-base/camera-interfaces/>
- [49] <http://ww1.microchip.com/downloads/en/DeviceDoc/00002800A.pdf>

- [50] <https://mipi.org/specifications/csi-2>
- [51] <http://www.coaxpress.com/>
- [52] <https://repository.tudelft.nl/islandora/object/uuid:e1badd8d-a384-49a1-b958-a0c1e499c539/datastream/OBJ/download>
- [53] "Adaptive Control Systems" by Gang Feng and Rogelio Lozano
- [54] [https://engineering.purdue.edu/~byao/Research/ARC/ARC\\_introduction.pdf](https://engineering.purdue.edu/~byao/Research/ARC/ARC_introduction.pdf)
- [55] [https://community.cadence.com/cadence\\_blogs\\_8/b/breakfast-bytes/posts/autos182](https://community.cadence.com/cadence_blogs_8/b/breakfast-bytes/posts/autos182)
- [56] <https://saras152.github.io/>
- [57] <http://flyingv.ucsd.edu/krstic/teaching/282/ioannousun.pdf>
- [58] <https://www.greyb.com/autonomous-vehicle-companies/>
- [59] [http://makita.in/index.php?route=product/product&product\\_id=560](http://makita.in/index.php?route=product/product&product_id=560)
- [60] <https://www.roboticstomorrow.com/article/2018/01/rtos-for-safety-critical-systems/11269>
- [61] <https://www.unece.org/fileadmin/DAM/trans/main/wp29/wp29resolutions/ECE-TRANS-WP.29-78r6e.pdf>
- [62] <https://graphics.stanford.edu/projects/sizes/sizes.pdf>
- [63] <https://www.sciencedirect.com/science/article/pii/S0968090X15003447>
- [64] <https://ieeexplore.ieee.org/document/6957660>
- [65] [https://www.nhtsa.gov/DOT/NHTSA/NRD/Multimedia/PDFs/VRTC/ca/capubs/NHTSA\\_for\\_kenbrock\\_driversteeringcapabilityrpt.pdf](https://www.nhtsa.gov/DOT/NHTSA/NRD/Multimedia/PDFs/VRTC/ca/capubs/NHTSA_for_kenbrock_driversteeringcapabilityrpt.pdf)

## ANNEXURE - 1

### 1 CHOICES AND PREFERENCES

---

#### 1.1 VEHICLE

The vehicle of choice plays a critical role in the demonstration of the project. The vehicle should have the following characteristics

- price should be within an affordable band for most customers
- sale volume should be sufficiently high
- most used as a commercial vehicle
- automatic transmission for the sake of simplifying the controls in the first adaption
- steering effort should be less (power steering with a hydraulic assist or electric steering)
- optional cruise control

The driving robot will be able to fit any car. However, the first attempt should be on such a vehicle that reduces the initial efforts of implementation.

One such car that satisfies the above requirements is Suzuki Dzire VXi AT. This vehicle is the most sold five-seater, used as a commercial vehicle, has power steering, has AMT, and costs about INR 8.7Lakhs (petrol, on-road price, Bangalore). The diesel variant got discontinued. Toyota Etios would have been the next best option if it had continued in production. Toyota Innova exceeds the average budget of INR 10lakhs for most people. Mahindra Verito has a manual variant. Hyundai Xcent's AMT is available only in the petrol variant. Tata Tigor comes with only petrol AMT. Honda Amaze diesel CVT can be a better fit. In contrast, Hyundai Aura, with a diesel engine and AMT transmission, can be a better fit. Hyundai Verna and Skoda Rapid are also possible candidates with their diesel engines and automatic transmissions, but cost more than the INR 10 lakh mark.

The cars I considered in the above list are all sedan types with a diesel engine to suit the taxi segment requirements of more luggage space.

The first vehicle for demonstrations should be avoiding the perception of bias towards the manufacturer.

#### 1.2 HARDWARE

##### 1.2.1 Power & Signal Distribution

The driving robot system comes with a variety of sensor and actuator arrangements. The operating voltage ranges for each component is different. For example, LiDARs from a few manufacturers operate exactly 12 V. In comparison, some operate between 9 and 18 V, and others operate between 12 and 32 V. The power consumption ranges between 8 and 60 Watts. The microcontroller and the GPU operate at less than 5 V. The power increases with the increasing computational load.

We need to design independent power modules for each of these components to avoid the risk of all components with a single failure. The power converter has to be switching regulators as opposed to a linear regulator.

We make sure to have star grounded connection for all the components in the network to avoid common-mode voltages. We must isolate the high frequency, and signal grounds from the power sector ground to avoid any voltage biases.

All the high-frequency signals circuits get protection from the external EMI/EMC by a faraday shield. Extra care is given in the PCB design to make sure that the circuit impedances are within the expected range of values. We also make sure that the PCB tracks are the shortest for this zone.

If the sensor is far from the controller, the corresponding signal cable gets shielding to protect it from the external EMI/EMC. Additionally, they will be a software check for data integrity.

### 1.2.2 CPU vs. GPU vs. TPU vs. NPU vs. FPGA

Each of the named processing units has a specific use in their application domain.

Artificial Intelligence (AI) models are computationally intensive and require many calculations to happen in parallel for time-efficient computation. Most critical of these computations is matrix multiplications.

- CPU is a widespread unit found on most microcontrollers and microprocessors. CPU has all-purpose architecture. We need the CPU on the controller to execute all the general scheduled tasks such as getting the data from the sensors, interacting with the customers, conducting actuator calibration, and testing the sensors.
- GPU is originally for graphics rendering. GPU has many smaller processing units (each with individual ALU for executing independent instructions, unlike CPU with only one ALU) that can do parallel computations. In recent times, the user applications are run on the GPU cores using particular compilers that can parallelize the program. GPU is beneficial for running machine learning or deep learning models quickly to get the results. We need GPUs for running our machine learning (ML) models.
- TPU by Google is a custom ASIC-chip designed from ground-up for machine learning workloads. Although GPU has multiple ALUs, each ALU has limitations by the Von Neumann Bottle-neck (has to access the L1 cache memory). The systolic array configuration removes the bottleneck. All the ALUs in a TPU get connected in a matrix. TPUs run the models created with TensorFlow only. TPUs are optional for our application. We should try to optimize the model to run on GPUs instead of investing in expensive TPUs.
- NPU is mostly used in mobile devices for its capability to execute image, sound processing pipelines, including speech recognition. Microsoft designed NPUs based on FPGA. This arrangement helps the programmer write the program directly on the hardware, avoiding the middle layer of conversions. The re-write operation is rapid.



We will have a CPU in the controller as a default configuration. A GPU and another CPU will be in the controller. When the GPU is non-operational or inaccessible due to pending loads, the CPU needs to execute the task at hand. Else, the second CPU should be working as a back-up CPU for the first CPU, taking over when the first CPU fails.

### 1.2.3 FPGA vs. DSP

DSP is a specialized signal conditioning processor that is optimized to run DSP algorithms. These are processor-based and are sequential in execution. Although multiple DSP cores exist that increase the computations per unit time, this does not help drastically.

FPGA is not processor-based; they are hardware implementation of logic units, memory, DSP elements. FPGA eliminates the sequential bottleneck that DSPs have. FPGAs are harder to program than the DSPs due to their intricate architecture and compiler configurations.

We will use DSPs for specialized tasks since the range of tasks we require are fixed and predefined. We can take advantage of the lower cost of DSPs and ease of maintenance.

## 1.3 SENSORS

### 1.3.1 Mounting

For the regular vehicle ride, the human driver comes in the loop providing the sensors, actuators, and intelligence. All the vehicle controls and the feedback systems are at ergonomic locations for the convenience of the person drivers. For example, the wipers for the windshield are required for human drivers to drive the vehicle in rainy weather. When the sensor is in the driver seat, the same control is required. However, when the sensor is on top of the vehicle, the wiper control is not required. However, the sensor needs to have a self-cleaning method.

If we replace the steering system with a custom system with an interface to the vehicle wiring harness, we can still access the wiper switches. However, this makes the driving robot less portable, and shifting between the vehicles becomes time-consuming.

This document proposes that all the sensors (Cameras, RADARs, SONARs, Mic) get mounted on a platform fixed on the roof of the vehicle. This arrangement gives better visibility of vehicle surroundings, in addition to reducing the need to adjust the rearview mirrors and wipers.

### 1.3.2 RADAR vs. LiDAR

LiDAR is expensive at this moment, and it takes about ten more years for it to become affordable for all autonomous vehicles. Google, Toyota, and Uber depend heavily on LiDAR, but Tesla depends on RADAR and Camera systems. Tesla uses RADAR for cost reasons, and its vehicles have the highest level of autonomy in commercially available products.

A small discussion about RADAR vs. LiDAR follows.

- **Range:** LiDAR and RADAR systems can detect objects at distances ranging from few meters to more than 200m. LiDAR has a problem with short ranges. RADAR can detect at less than 1 meter, but it has to be short-range radar.
- **Spatial resolution:** Due to the possibility of 905nm to 1550nm, IR light spatial resolution can be as low as 0.1deg. Radar's 77GHz (or 4mm) struggles to resolve small features as distance increases.
- **FoV:** LiDAR has better vertical FoV (than radar) angular resolution for horizontal and vertical - necessary for object classification
- **Weather conditions:** Radar is robust in rain, fog, snow. Using 1550nm (IR wavelengths) help LiDAR achieve better performance under unfavorable weather situations.
- **Cost & Size:** Radar has become compact and affordable. Right now, LiDAR costs below 10kUSD. Some predict that the LiDAR costs fall below 200USD in few years.

To build a commercially viable product, RADAR is the best bet. The controller will do a lot of image processing to compensate for the lack of LiDAR.

Some of the LiDAR specifications available in the market are in the table below.

Type	name	weight	range	power	point density
Mechanical Type	Velodyne puck lite	590gm	100m	8w	300k pts/sec
Mechanical Type	Velodyne alpha puck	3500gm	300m	30w	9.6mil pts/sec
Mechanical Type	Velodyne hdl-32e	1300gm	70m	12w	700k pts/sec
Mechanical Type	Velodyne hdl-64e	13000gm	120m	60w	2.2mil pts/sec
Mechanical Type	Velodyne puck 32mr	925gm	120m	10w	1.2mil pts/sec
Mechanical Type	Velodyne ultra puck	925gm	200m	10w	1.2mil pts/sec
Solid-state Type	Robosense rs-lidar-m1	800gm	180m	25w	1,125mil pts/sec
Solid-state Type	Robosense RS ruby	3750gm	250m	45w	4.6mil pts/sec
Solid-state Type	Leddar Vu8	128gm	185m	2w	-NA-
Solid-state Type	InnovizOne	515gm	250m	23w	22.5mil pts/sec

### 1.3.3 RADAR

We need both SRR (Short Range RADAR) and LRR (Long Range RADAR) for our vehicle. Commercial implementations of 77GHz (higher allowance for equivalent isotropic radiated power - EIRP) radar solutions are available from many manufacturers such as NXP, ST, Infineon, TI, Bosch, Aptiv (Delphi), Continental, Denso, TRW (ZF now) and Hella. Few startups in this area are Einstein, Metawave, Uhnder, and Vayyar. The 24GHz band will be phased out by early 2022 in the USA and Europe.

A RADAR (**R**adio **D**etection and **R**anging) system includes an antenna that alternates its connection with a transmitter and a receiver using a duplexer. The receiver includes ADC,

Pulse compression module, Doppler processing module, and detection & tracking module (a DSP). Phased Array Radar includes multiple antennae & T/R modules. It directs resulting beams in the required direction by assigning different phases to different modules. The front-end of the RADAR is still analog because of the high frequencies involved. Some of the products in the market include only front-end sensors and mixed-signal ICs (CMOS single-chip format), including front-end and on-chip processing modules. The single-chip solution reduces their size, power consumption, and cost.

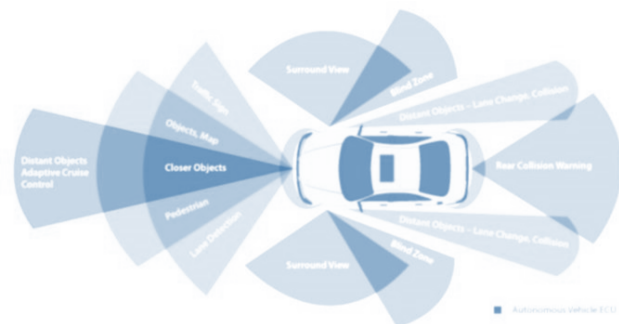
There are some technical problems with the RADAR. Multiple reflections from the targets can cause ghost targets; the advanced tracking algorithms eliminate these. There is an ongoing concern about multiple radar systems from different vehicles interfering and creating a faulty output. Active research is going on to mitigate these effects. FMCW (Frequency Modulated Continuous Wave) RADAR implementation detects a ghost target (due to interference) if there is an identical radar with an identical frequency nearby.

The output of RADAR can be a few 10's of MBs to 100's of MBs. UDP protocol is preferred when the time performance of the data is essential. TCP is required when the data loss is unacceptable, but the real-time transfer need not happen. Data from RADAR to the decision-maker (driving robot controller or the surrogate driver) goes via UDP protocol.

The raw RADAR data is usually used for the signal processing at the site, and not usually distributed outside the RADAR site. Processed data from this raw data gets usually distributed as "moment" data containing reflectivity velocity and location information.

Data compression is one of the ways to reduce the bandwidth requirement for transferring the data from the sensor to the processing unit. The RADAR output will go through some AI modules for object detection in the modern processors.

The next image shows an implementation of the surround perception around the vehicle by a research group. This arrangement includes Cameras and RADARs.



We need to have LRRs for front and rear placement, SRRs for lane change view, side view, and other purposes.

The vehicle speed for the gen-1 is limited to 35kmph (about 10mps). An LRR with a scan range of about 200 meters gives a buffer time of 20 seconds for reacting to an obstacle. For gen-1, we can work with MRR (Medium Range RADAR) with a range of about 100 meters.

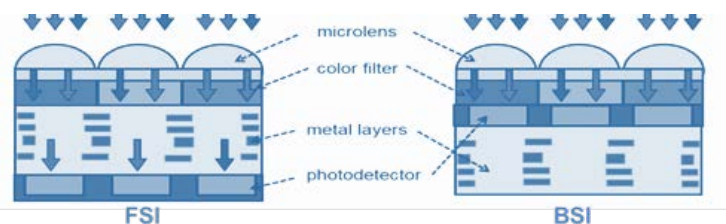
RADAR, including the DSP unit, comes in a single physical package and consumes very little power. We will need 2 MRRs (front, rear), 6 SRRs (left & right side facing, all four corners) to cover the vehicle all around. At a later point in time, we can revisit this number of sensors based on the horizontal FoV coverage for each of the mounted sensors. We prefer to use FMCW based RADARs to reduce the probability of ghost targets due to the interference from neighboring vehicles.

### 1.3.4 Cameras

This section includes many types of cameras and lenses. Our system of vehicle control depends on many vision algorithms that consider images from both visible spectrum and thermal sensors. RADAR data will be appended to this information when the weather situations are not favorable.

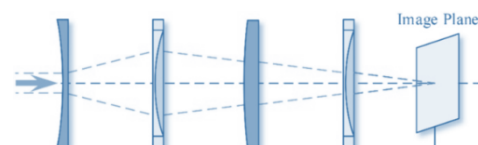
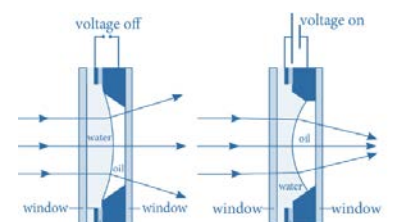
CCD (Charged Coupled Device) sensors for cameras are analog type linear sensors with an output directly related to the photons received. CCDs move charges from pixels to pixels until the amplifier stage in the signal capture zone. CIS (CMOS Image Sensors) for cameras are the latest development and give out parallel read-out. CIS results in digital output at each pixel, which can be read-out independently. CIS is available in most modern cameras and costs less.

Recent developments in the CMOS sensors result in smaller pixel sizes with higher resolution, smaller sensor size, and lower power consumption. Traditionally,



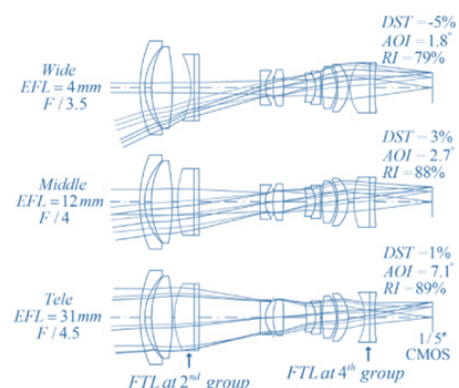
CMOS sensors come with FSI (Front Side Illumination). The recent trends are towards BSI (Back Side Illumination) for better image quality. The BSI has color filters and micro-lenses on the back of pixels resulting in the short optical path and deep photo-diode. This arrangement causes higher quantum efficiency and lower crosstalk.

Rolling Shutter (RS) is a method in which the image is scanned sequentially from one side of the sensor. Global Shutter (GS) method results in an entire image scanned together by all the pixels. This construction of GS type sensors requires a supporting memory structure and additional transistors to provide this functionality. GS type sensors avoid distortion in the image and artifacts due to the parasitic light sensitivity and found its use in various applications in the automotive field.



A more recent development is optical phased array sensors for the lens-less cameras. This optical phase array potentially can have slim cameras that can get pasted on surfaces.

For focusing the image, a fixed type lens such as those working on the principles of electrowetting will be helpful. A combination of electrowetting lenses with a fixed lens helps us get some image zoom. This approach saves space and improves system reliability by avoiding moving groups.



A possibility is to get an 8x optical zoom with four groups of fixed optical groups, with two of them being tunable focus lens (based on electrowetting principle) is represented here.

Alternatives to electrowetting are

- The acoustic-optical technique uses a lens with GRIN (Gradient Index of Refraction) that uses standing sound waves to produce continually changing GRIN within a liquid contained in the lens. The sound waves send vibrations, causing the molecules to move together, altering the refractive index. By changing the shape and location of the sound waves, it is possible to obtain the required equivalent lens shape.
- Electro-mechanical methods have a lens with a container filled with an optical fluid sealed off with a thin elastic membrane. The membrane's deflection is proportional to the cell pressure. The pressure changes through electromagnetic or manual means.

Most of the camera systems on the vehicle will have a fixed focus. We will have different lenses fixed to the front of the vehicle that helps achieve the best possible use of the camera. The front-facing system will have

- Two fixed focus cameras
  - One with a wide-angle lens,
  - Other with narrow FoV to look ahead
  - Both with color image output)
- Zoom tunable lens
  - To selectively focus to either narrow FoV or wide FoV
  - Acting as a backup camera for each of these

We need depth sensing for object distance estimation, which can be achieved by a dedicated set of cameras. A stereo camera system (or a depth camera system with ToF sensing) helps with this. Such a system can be created by arranging a pair of independent cameras and running all the post-processing algorithms on both the resulting images. Commercially available solutions can probably run the most used algorithms very fast due to the optimizations and unique hardware configurations.

Depth sensing is possible in multiple ways:

- The passive stereo system depends on the available ambient light but performs poorly in low light conditions
- The active stereo system employs a light source (LASER or structured light, Infrared). Its performance in low light conditions is better than the passive system, but the same as the passive system in well-lit conditions or over a long-range.

The baseline is the distance between the cameras. More baseline distance ensures better depth perception. Better resolution of the cameras results in a higher disparity between images. A higher resolution enables calculation of better depth information, but at the cost of more computational load. With lower focal length, we can see further but at a reduced FoV (Field of View).

The long-range depth-sensing comes with much error. The human eye is equivalent to a 576 Megapixel camera. So with a smaller (50 to 75mm baseline distance), we can perceive better depth even at more considerable distances.

We need to establish the broadest possible baseline suitable for all the cars when we design the stereo system. Consider this to be mounted on the vehicle top. A structure with 1-meter width with cameras mounted at both the ends will give us a lot of baseline width. With moderate camera pixel resolution (of about 20MP), we can calculate the distances of objects of interest in our FoV.

The choice of all color images, as opposed to monochromatic images, is critical. Tesla uses monochromatic images. This usage once caused their vehicle to crash into a white truck in bright sunlight. Color gives high information for segmentation and object detection. However, this comes at a more computational cost and processing delays.

We will have the stereo camera system mounted on the top, 1 meter as base distance, 20MP resolution for each camera. The stereo camera is in addition to the number of cameras for object detection (1 narrow FoV camera for looking ahead, one wide-FoV camera for peripheral vision, two side cameras facing rearward for lane change assist, one rear-facing camera for behind view, one zoom camera facing forward as a backup for front vision). High-resolution cameras have data transfer requirements from 50 MBytes/s to 1.2 GBytes/s, depending on the number of cameras on board. Most (<78%) OEMs planned that 50 -70% of this data needs to be analyzed locally than to be sent off the vehicle.

The algorithms required to process images from the cameras depend on the objective. The range of these algorithms is extensive, starting from a simple line detection algorithm and going all the way to the object size estimation and the position estimation.

One of the primary activities done on an image is the extraction of features in that image. During the author's earlier work on image processing algorithms, he implemented many algorithms that have much mathematical rigor. One such algorithm is the Hough transform. This method is a feature extraction technique to identify geometrical features in an image. Powerful tools like OpenCV enable us to implement many algorithms without having to get into the details of the working principles.

Some of the algorithms used during the post-processing of the images are:

- Road lane detection
- road sign detection
- image stitching to create a 360° view around the vehicle
- pedestrian detection
- the nearby vehicle detection
- lane change obstacle detection
- weather condition bias removal
- sensor fusion

## 1.4 EMBEDDED SOFTWARE

Embedded software gives expression to the control algorithms for this application. The OS implementation can be a bare-metal OS implementation or an RTOS implementation.



Depending on the hardware platform, appropriate HAL (hardware abstraction layer) will be used to create modular software for ease of use irrespective of hardware changes.

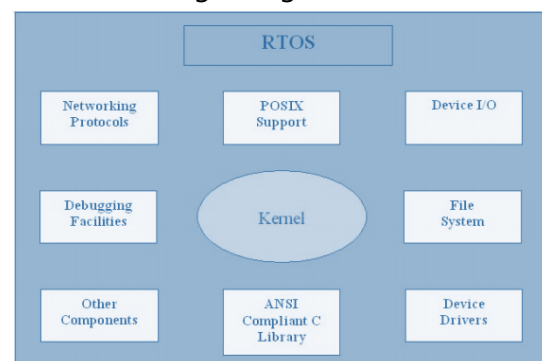
#### 1.4.1 Bare-metal OS vs. RTOS

Bare-metal implementation is the simplest model of the execution of a program on a microcontroller. Bare-metal OS is a highly customizable mode and is a preferred choice for a small program. There are no schedulers or semaphores in this. Multitasking is only possible as long as the hardware modules allow. Execution of the program on the microcontroller is controlled by one infinite loop, which calls a sequence of functions that run depending on the input and output conditions. An implementation of this bare-metal OS includes

- continuous polling of the flags for the corresponding hardware modules
  - to check if an execution of a task or a function is complete
  - so that the next task or a function can execute

The other implementation includes enabling the interrupts of the hardware modules and servicing the interrupt service routines (ISR) when an interrupt gets generated. Implementation of this bare-metal OS is user-specific and can get very messy when the code is very complicated.

An RTOS is a multitasking operating system intended for real-time systems as opposed to a generic OS without real-time preference. RTOS uses pre-emptive task scheduling pretty low priority task pauses to execute the high priority task. Every task in the RTOS as a bonded latency that is it gets executed within a known time limit. A high priority task can pre-empt the kernel of the RTOS. In a standard OS, the kernel has the highest priority. The task execution in an RTOS need not be the fastest. The time of execution should be deterministic. There are three types of RTOS systems-hard, firm and soft. In the hard-RTOS system, the task deadlines are very strictly imposed - the task must start executing at a given time and finish within an assigned duration. An example of a hard-RTOS system is running in aircraft systems, automobiles, and medical critical care systems. The firm-RTOS systems have slight flexibility on the task execution deadlines at the cost of the system performance-a missed deadline may not have a significant impact. However, it may reduce the quality of the output. An example of a firm-RTOS system is running in most multimedia applications. The soft-RTOS accepts some delays. An example of the soft-RTOS is running in online transaction systems.



The RTOS has functions for memory management, device driver, scheduler, and middleware (TCP/IP stacks, USB, HD video, communication drivers). These features help in reducing the application code size.

The bare-metal OS all these functionalities and drivers have to be written manually, and this takes a very long time to create a bug-free implementation. An RTOS helps the user change the code architecture or include any other significant change with minimal code

modifications. The RTOS is more portable and is a good fit for reuse. The use of RTOS means the programmers focus their efforts on the application code development.

The RTOS generally consists of 2 parts-kernel space, user space. User-space is where all the user functions are declared and used. The kernel includes the core functionality of the operating system. There are three varieties of kernels - monolithic kernel, microkernel, exokernel.

The monolithic kernel has all essential system services and provides abstractions for the underlying hardware. The number of context switches is low, resulting in a faster kernel. Some of the examples of the monolithic kernel are Windows and Linux.

The microkernel runs only necessary process communication and IO control. The other system services (including file system management, networking, and memory management) stay in user space as daemons. The microkernel has a very simple hardware abstraction, but this is more stable than the monolithic kernel. Some of the examples of microkernel are QNX and AmigaOS.

The exokernel gives more control of the hardware to the application. Exokernel only runs services that protect resources such as tracking the ownership, guarding the usage.

#### 1.4.2 RTOS

There are a lot of RTOS options available for implementation. Our essential requirements include security and support for bulk data handling.

Linux based RTOSes (RTLinux or Ubuntu with preempt) are popular. However, Linux is not originally an RTOS platform. Linux based RTOSes have substantial boot-time overhead and take a long time compared to regular RTOSes (FreeRTOS or uC/OS-II or RTX kernel or embOS or eCos or QNX neutrino). However, Linux based RTOS comes with extra conveniences such as file system support and integrated peripheral drivers. Linux based RTOS has an advantage when the platform is in a distributed network with a lot of data transfer, such as ours. However, it is not a good match for deterministic RTOS systems.

This document prefers to have a Linux based OS for the convenience of the networking and other libraries. The second alternative is not a generic RTOS available for free. However, we should go for a deterministic RTOS that offers all the peripheral drivers, and security (encryption) integrated.

### 1.5 DEEP LEARNING MODELS

We have a plethora of choices for implementing deep learning models. We will make a final choice during development, depending on the accuracy and the computational load.

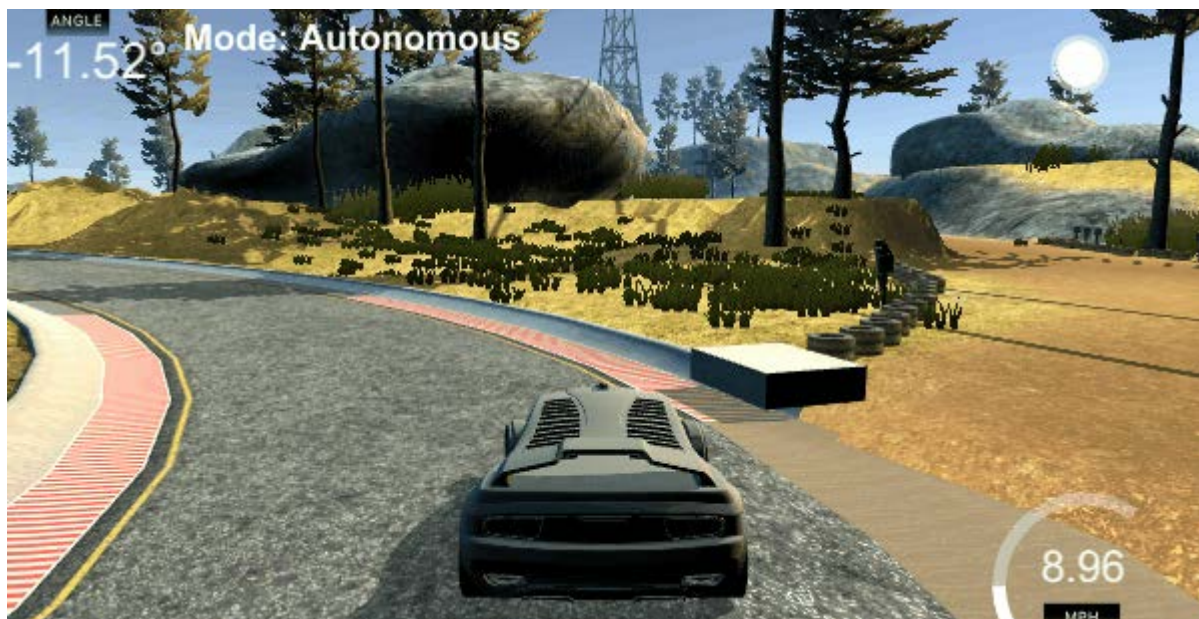
Image classification CNNs range from LeNet-5 to AlexNet-7CNNs to FixEfficientNet-L2 (480 million parameters) with the top-1 accuracy of ImageNet predictions going up to 88.5%.

Object detection, image segmentation, posture recognition, and several other tasks require that we train the deep learning models using an appropriate dataset.



The following example discusses driving a virtual car in a simulation environment using only the camera images. In this case, the training data for the deep learning model was from multiple manual runs on the same track. Throttle, brake, steering angle, and the camera images from the manual run are the inputs to train the deep learning model.

We make sure to remove the data bias so that our model is trained equally well for all the track positions and the camera views. In this particular case, the author created the model using TensorFlow running on Python. The training was done on GPU equipped computer to save time. The trained model is used in Python to take the camera images as input and give out the instantaneous steering angle, throttle, and brake values to the simulator.



*Figure: Simulator environment for behavioral cloning*

We create the local 2D route map of the easy navigation and faster model runs. This local 2D map is a bird's eye view map created from the camera images as well. This map also gets updated whenever we get an updated view of the surroundings.



*Figure: Standard camera image to the bird's eye view projection result*

This 2D route map is dynamic for a limited range to keep track of the road conditions and space to move around. An example of creating a 2D projection of the top-view from an isometric view is available in the following figure.

The projections all around the vehicles are stitched together in 2D space. This 2D map is continued with time, covering more area as the vehicle goes around. This local 2D map helps in navigating routes with multiple turns and road islands (for example, parking lots), and very sharp turns.

The following is the model architecture for the traffic sign classification from the images.

- Convolution Layer to accept 32X32X1 images to convert them to 28X28X10 through a 5X5 filter and a stride of 1
- ReLu Activation
- Maxpool with a stride of 2 and 2X2 as window size
- Convolution Layer to accept 14X14X10 and to output 10X10X20
- ReLu Activation
- Maxpool with a stride of 2 and 2X2 as window size
- Flatten the layer to result in 500 outputs
- Fully connected layer with 500 inputs, 700 outputs
- ReLu Activation
- Fully connected layer with 700 inputs and 200 outputs
- ReLu Activation
- Dropout with 0.85 keep fraction
- Fully connected layer with 200 inputs and 80 outputs
- ReLu activation
- Dropout with 0.85 keep fraction
- Fully connected layer with 80 inputs and 43 outputs

This model gives the following top 3 predictions for the provided input images.

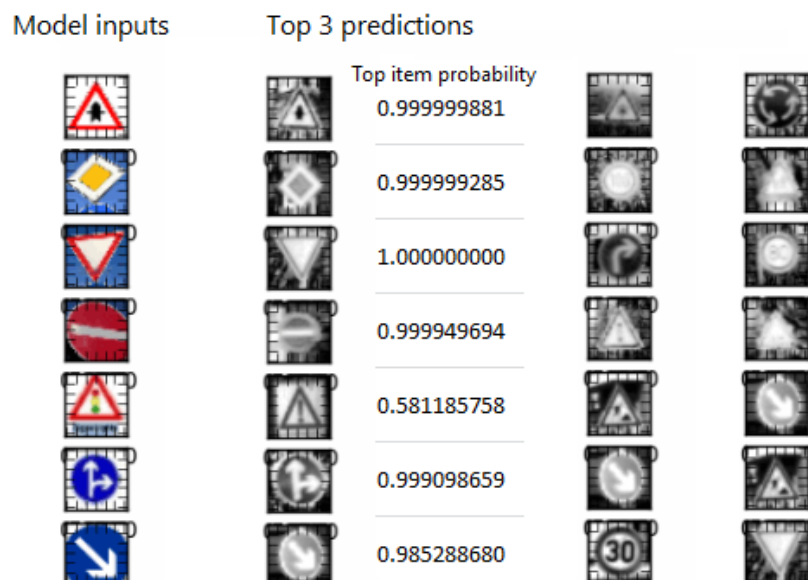


Figure: Model prediction results for the traffic signs

## 1.6 ACTUATORS

### 1.6.1 Steering Wheel

We have many options to execute this mechanism. The first and most straightforward solution is to implement the rotary actuator that most companies for the testing robots. It can have circumferentially mounted or axially mounted configuration. In both of these configurations, the steering wheel is as existing. This approach involves much money to purchase a steering actuator because of its specifications-torque range, angle range, and response time.

The next alternative is to have a custom steering wheel with a low-cost rotary actuator mounted on the steering shaft. This alternative requires some engineering time and prototyping efforts for each vehicle type.

A more sophisticated implementation is to have robotic limbs that hold the steering wheel. This implementation requires the design of the mechanical system, electronics, control systems, and implementation. This implementation will be a general-purpose solution but requires many engineering resources.

This document proposes that we start with the bought-out steering actuators, and then increase the complexity (and reduce costs) gradually.

### 1.6.2 Foot Pedals

We can implement this mechanism in multiple ways. The first option is to have hydraulic cylinders that press the pedals at standard angles to the pedals. This option is the easiest but challenging to have a linear output. We need to have more sophisticated hardware and the control system to make this work as per our requirements. The hydraulic system includes managing fluid and line pressures.

The second option is to have the linear motors that can be controlled by appropriate hardware. Cost is a concern here.

The third option is to have the robotic limb implemented that works like a human leg to press the required pedals. We only need to implement two limbs as opposed to three actuators from the above two options.

This document proposes that we start with the linear motor and then proceed with the robotic limb when the opportunity comes.

### 1.6.3 Gearstick

Gearstick control is not necessary for vehicles with automatic transmission. In the vehicles with a manual transmission, the driving robot moves the gear stick to the appropriate slot. This actuation has to happen synchronously with the clutch actuation.

One possible option is to have a system two DoFs for a planar movement for the tip (handgrip area) of the gear stick. In this case, the gear stick is fixed to the actuator set rigidly. An implementation with hydraulic pistons should be sufficient for this since there is no

requirement of force control. The gear stick stays in its position once it reaches the required gear position, and does not need to maintain position control algorithms rigorously.

Another option is to implement the robotic limbs (that hold the steering) to operate the gear stick when required. This operation is similar to how a human driver uses one of the hands to move the gear stick. An additional advantage with this method is that we can also use the same robotic limb to control the parking brake lever. We just need to define the robotic limb to have sufficient operating space.

This document proposes that we start with the set of actuators fixed to the tip of the gearstick.

#### 1.6.4 Steering Mounted Controls

There is no easy way to do this. We can implement a custom steering wheel which has a coupler setup that gets integrated with the wiring harness of the vehicle. Then the controller can pass the required switch commands by signaling required voltages at the required pins. This arrangement requires much time to fix the driving robot on the vehicle.

Another option is to implement the robotic arm that can reach all the controls on the steering wheel.

This document proposes that the sooner we start working on the robotic arm, the easier it will be. We can start with not having these controls in the internal trials. However, we can work on having the wiring harness integration with a custom coupler – without having to change the entire steering system.

#### 1.6.5 Robotic Limbs

Robotic arms and legs seem to be a solution that fills many functionality checkboxes. Mechanical implementation of the robotic limb is a very well established area, and we can take help from standard robotic limb suppliers. We can use some of the open-sourced designs. We need to make sure we choose the right control strategies for reaching out to the target location, satisfying the time requirements.

The significant change in our requirements is to have a reliable operation, even during a vehicle power failure. We cannot depend on the vehicle power system entirely. We need to have independent battery packs for each of the actuators (robotic limbs in this case). These battery packs are available only under emergency circumstances to maintain communication with the primary controller, which also comes with a back-up battery pack. The energy rating of the mounted battery packs should be sufficient for a 300-second operation. We can use the low-cost options for the battery cells (18620 type Li-ion cells) because they are mass-produced and are available for replacements when needed.

The robotic limbs will have the necessary intelligence of their own, in addition to the support provided by the driving robot controller for their movement. The internal intelligence can assess if the commands received are possible or if the limb is going out of its operational domain. This intelligence can store some of the past failures and the circumstances as well. This local storage helps us reduce the memory storage capacity of the primary controller.

The standard low-cost robotic limb uses DC motors. We should use the energy-efficient BLDC motors in the limb joints, with reduced maintenance, better control possibility, and smaller size. The workshop handheld power tools have moved towards the BLDC motors because of the lower space requirements, longer operation times, and higher torque densities. These tools operate with up to 18V (and 3Ah) batteries and generate very high torque densities. For example, the Makita power impact wrench can have a maximum tightening torque of 280 N.m. The cordless drill has a maximum torque of 54 N.m. Some of these motors are very compact, with their diameters ranging between 40 to 50 mm and lengths up to 60mm.

## 1.7 LIMP-HOME MODE

A 100% failure-proof system is never possible. The best system is the one that handles the failures gracefully.

The autonomous mode of the driving robot should have a failure protection mechanism. If there is any failure of critical systems disrupting the standard operations, the controller should activate the limp-home mode. This mode should activate the safe-parking mode, where the driving robot looks for the nearest safe parking space or the side of the road from the current lane and parks the vehicle. The driving robot also emits messages on all possible communication channels instructing neighboring vehicles about the slow-moving vehicle on their way. The driving robot also sends messages to the nearest service station and the command center about the error for possible resolutions.

For the initial duration of the project, it is ideal to have a standby driver in the vehicle (or a trained user) to operate the vehicle to handle such failure modes.

## 1.8 MEMORY MANAGEMENT

### 1.8.1 Storage

Human beings come face to face with a large amount of data every day, but we do not store all the data in our brains. Our memory system is very efficient in storing what is required, and it recreates the full memory from the fragments of memory. We should replicate a similar system for storing the event data on the driving robot or the cloud servers.

Storing the raw data from sensors is not completely useful unless we want to recreate the exact situations in which few actions have occurred. We have many sensors that capture redundant information about the surroundings. We can eliminate redundancies while storing the data. We can overlap sensor information and only store the fused information. We can use the data compression techniques for storing more data in limited memory. We can mask all the other information not relevant to the event and store only fragments of memory required for analysis. The controller should check for the sufficiency of the stored fragments to recreate that event only or debug in the situation before deleting all the information.

### 1.8.2 Data Transmission

We need to have local processing of data within the sensors. If we have eight cameras, all the cameras with at least 8MP resolution operating at 60Hz give out data at about 30Gbps rate. All these data need not go to the driving robot for processing. The camera modules should have local processing capability (for event detection or object recognition algorithms). Only those frames that have information will go to the driving robot for confirmation.

## 1.9 AUTOMATIC DOORS

We focus on passenger health in times of increased awareness about personal hygiene. The vehicle doors should have a mechanism to be opened from the driving seat, by the driving robot. The taxis in Japan have a mechanical system that can open the doors (rear passenger door only) with a pull of a lever. Such a system helps the passenger avoid touching the doors, which are one of the most customer-interacted parts. We can implement an electronic variant of this system to give better treatment to the customers.

### 1.10 SMART FEATURES

The vehicle should have smart features for sensing the passenger presence, their mood from the way they interact, the objects they carry with them (while entering and exiting). The vehicle can remind the customers if any items are left behind, detect if any items are in the cup/umbrella holders.

We should theoretically reduce the number of storage options for the passengers, providing them with one or two zones with much space for storage. All the smaller storages such as pockets at the back of the driver seat should be minimal, lest someone will forget an item.

## 2 TESTING

---

Testing is a crucial step in product development for safety-critical systems. We test the driving robot in multiple stages at multiple complexity levels. We test to make sure that the part/component/module matches the requirements stated in the product & project requirement documents.

Simulation for testing the algorithms at many stages is essential. We simulate and validate the control algorithms (vehicle dynamics control, actuator position closed-loop control) using tools with multi-physical system modeling capability. The model fidelity is crucial to get accurate tuning of the control algorithm that works in the actual vehicle.

We also simulate AI models in a virtual environment for autonomous vehicles such as CARLA, AirSim, OpenDS, Apollo, Deepdrive, LGSVL, and Nvidia Drive. This simulation includes different environmental conditions, road conditions, traffic scenarios, pedestrian behaviors, unique lighting conditions, and different tuning parameters for the models.

Training data decides the effectiveness of the deep learning models. We will access the public data released by the research institutes for basic training. We need to do many road tests and simulation runs to generate a wide variety of data to train our model. When we



choose the number of layers, the drop-outs, activation functions, and biases carefully, we avoid over-fitting the model based on the training data.

We test electronic hardware under standard ECU development environments, including EMI/EMC compatibility, besides the required functional tests and endurance tests. Hardware-in-loop setup is useful in evaluating the functional performance of the controller units under various input conditions.

We test the mechanical actuators in the part-level test rigs for the functionality in the initial stages. After that, we build a full test suite for checking the performance and endurance. Once all individual components are functional, we integrate and then test on the vehicle setup in a chassis dynamometer. After this stage, the vehicle goes into a controlled environment with known traffic and obstacles and tested under supervision. After this stage, we test the vehicle on the actual road conditions with a support driver.

The field testing of the driving robot or the parts of the project requires life-size testing ground with multiple scenarios. The best workable option is to construct a ghost city in the size of a small colony, with people dummies and vehicle dummies going around in a pre-programmed pattern or randomly. A server-level program operates the dummies.

### 3 SCOPE FOR INNOVATION

---

The field of autonomous vehicles has much scope for new researchers to innovate.

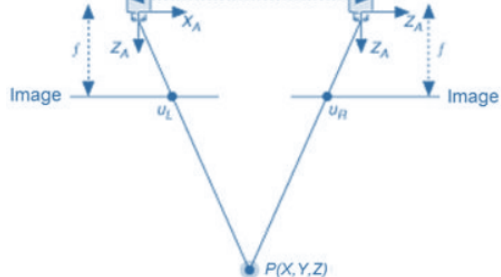
The following are some projects that can give enough challenge to the engineers:

- Construction of a 3D model of the environment around the sensor/vehicle. We can append this data with the inputs from the other sensors, such as LiDAR or RADAR. We need to depend less on the image-based deep learning models to find the steering angles or required speed (as in most of the traditional approaches). We should depend on deep learning models only to create 3d models of obstacles for the travel. This altered preference reduces the effects of training bias on the DL models. What it also means is that we need to work on compact models to represent the 3D environment besides being able to identify the signs and signals from road markings. Multiple models may have to work parallelly to provide inputs to the final controller so that the decision is quicker based on the computation time of series processing. A partial model of the entire world captured so far will be loaded in the working memory dynamically based on the estimated position, the access speeds, and loading speeds that need to be made fastest. Once this module is complete, it becomes useful for scenarios where connected mobility exists. The vehicles that have traveled in a path may upload the model data to the central server available to any other vehicle that



intends to go on that route. This sharing reduces the computational burden on the processors. We still have to work on the data bandwidth requirements.

- Stereo vision to identify the objects' relative distances (Stereopsis). This approach works similarly to the binocular vision in humans or other animals. This arrangement helps us identify the surroundings more accurately and focus on only that part of the model, which is of prime importance to the next action, such as movement or braking. With the scientific tools based on electrowetting or similar concepts to change the focal length on the fly, it is possible to compute the object distance accurately. This approach is better than the distance estimations got based on single camera-based methods. Once this module becomes functional, this gives inputs to the controller about the objects' positions around itself, and it can be a functional replacement for other sensors and would cost much less.


- Flexible vehicle control functionality. Every vehicle would have a unique position and the effort required for steering, braking, and accelerator lever. The dynamics of various cars are very different. A very well designed adaptive control system would reduce the need for custom configuration for each car. The driving robot, when fitted in the car, would start by calibration of the levers, steering, and foot pedals, much like a human operator. It could even run a test distance under supervision to select the control parameters. These parameters include steering effort, brake lever travel vs. braking force, accelerator pedal travel vs. power delivery. Some parameters, such as vehicle roll, suspension softness, are pre-configured. Once implemented, this reduces the efforts of using the same driving robot on an unfamiliar vehicle.
- Rare Earth Element free actuators. Considering the recent pandemic that started in China and disrupted the supply chain of almost all production facilities, this document proposes to design and build the actuators for the robotic arms and legs without magnets. The arms will have motors for motion control, and the motors have permanent magnets in them. Avoiding these magnets is possible when a custom motor design is taken-up with either switched reluctance motor or synchronous reluctance motor or custom stepper-motor. A custom design may be an expensive affair for low volumes but reduces the costs of the robotic arm drastically when we consider high production volumes. The custom design also helps us keep our solution unique.



Other possible research project titles:

- Imaging systems
  - Image stabilization with tilt and vibration compensation for vehicle mounting
  - Object distance estimation through variable focus lens / electrowetting lens
  - Object distance estimation through a binocular image system
  - Object size and speed estimation using LiDAR and RADAR combination
  - Object classification using data from an IR camera
- 3D models
  - Building a 3D model of the world using the sensor data, with the lowest storage requirement
  - Packaging the 3D model data into smaller data packets for storage on the server for easy retrieval
- Controller & AI
  - Custom processors including AI modules and signal conditioning hardware
  - Basic onboard AI for when the connection with the surrogate driver is lost
  - Control of movement of the robotic arm to the target position
    - With obstacle avoidance
    - With multiple motors actuated together (smoother and faster)
  - Force feedback system for the surrogate drivers
  - The sensor mount stabilization setup
    - Use the tilt sensors
    - Compensate the tilt with balancing motion of the sensor mount
  - High-speed storage for the system data (GDDR6 or alike)
    - To store TBs of data getting generated for diagnostics and analysis
- Mechanical actuation system
  - Robotic arm with embedded motors for motion
    - Power lines run through the limbs, but signals are wireless
  - REE free robotic arms
    - Without magnets
- Network interface
  - Data exchange with V2X interfaces including payment gateways
- A simulation environment for algorithm testing
  - Our custom test suite including custom test conditions and environments
  - Based on the open-source options
- OS for the controller
  - Open-source based but security centered
  - Supports special routines for self-testing and calibrating the modules
  - Supports OTA and secure boot-loader with an optional recovery mode
- Platooning for vehicles
  - With basic autonomy for the following vehicles
  - Useful for the driving robots and the surrogate drivers
- Camera system and lens integration hardware including the electronics development

- Image data processing and judging the distance and shape of the object from the stereo input and the autofocus data.
- 3D model of the environment based on the inputs and keeping track of all the objects in the virtual world.
- Actuators—mechanical hardware and electronics
- Sensors for the actuator state and feedback system
- Primary control system for identifying the robot's location and maneuvering around the obstacles to reach the destination
- Secondary control system for the actuator control based on the set value.
- GPS and IMU sensor units, internal/external to the robot that locates the vehicle on a global map.
- Target location identification and route planning
- Training and calibration for vehicle types and other customizable parameters
- Central console and information display with all the parameters of the vehicle and the robot.