Technische Universiteit
**Eindhoven**
University of Technology

# Active Lane Centering System

**Authors:**
Anshuman Singh
Davide Occello
Raymond Wouters
Sharad Bhadgaonkar

Automotive Systems Design
Stan Ackermans Institute
Eindhoven University of Technology

January 30, 2017

# Active Lane Centering System

## HIGH LEVEL DESIGN DOCUMENT

Eindhoven University of Technology
Stan Ackermans Institute / Automotive Systems Design

The design that is described in this report has been carried out in accordance with the TU/e code of scientific conduct

**Partners:**

**Stakeholders:**
Gijs Dubbelman
Gerardo Daalderop
Rameez Ismail
Peter Heuberger

TECHNISCHE UNIVERSITEIT EINDHOVEN

## *Abstract*

Automotive System Design
Department of Mathematics and Computer Science

**Active Lane Centering System**

by ASD Group ALC

This report presents a High Level Design Document of an Active Lane Centering System (ALC). This document represents the deliverable for the second phase of the project, which was mainly focused on the high level design of the ALC system with a top down approach, and later on the analysis and improvement of the existing hardware architecture. The high level design was derived using the CAFCR approach, which derives a conceptual architecture from customer objectives. This design was combined with the insights coming from the previous phase on functional safety in order to produce a safer concept architecture. The second part of this phase involved the analysis of the hardware components of the existing architecture and of the possible automotive grade communication protocols. The insights deriving from this analysis allowed us to propose a modified version of the architecture, which aims to improve the safety, the performance and the practical implementation of the system. The proposed architectures follow the development process of the final ALC system. One will be implemented in the following phase of this project, and other two architecture proposals which will be more efficient and compact for future implementations.

# Contents

Chapter 1

# Introduction

Advanced driver assistance systems are one of the fastest-growing segments in automotive electronics [2]. These are systems developed to automate/adapt/enhance the vehicle systems for safety and better driving. Safety features are designed to avoid collisions and accidents by offering technologies that alert the driver to potential problems, or to avoid collisions by implementing safeguards and taking over control of the vehicle. A large percentage of accidents occur due to unintentional lane change [1]. Active Lane Centering System (ALC) is a system designed to avoid such accidents or collisions by actively maintaining the vehicle at the center of the lane. If unwanted drift from lane center is detected, ALC will perform an appropriate control action aimed to steer the vehicle back to the desired trajectory. Hence, broad level goals for ALC system are to improve the safety of the driver by keeping the car in the safest zone of the lane and reducing the driver workload by automating the most common lateral control actions.

## 1.1 Project Objective

The second phase of the ALC project, focused on the delivery of a high level design of the ALC system. The design was broken down into different phases, which follow the CAFCR Framework as a guideline. The major project objectives which were achieved during this phase are the following:

- Definition of the Customer Objectives
- Definition the Application areas of the system
- Derivations of Functional Requirements
- Improvement of the Concept functional architecture
- Analysis of the available Hardware Components
- Benchmark of communication technologies
- Proposal of High Level Architecture Designs

All the information which was acquired in the previous phase was used in this phase to deliver a safer Architecture.

# Chapter 2

# System Design

## 2.1 Introduction to CAFCR Framework

The CAFCR is a method which is used to transform customer objective and key drivers in an actual system design. This method consists of five different views that include: customer view, application view, functional view, conceptual view and realization view. The customer view deals with the desires of the customer in terms of key drivers. The application view describes the needs of the customer in terms of how the customer would like to realize his goals. These two views provide the justification for the design of the system in the other three phases, or in other words, **why** the system should have certain functions. The functional view describes the system from an external perspective and shows **what** the system should do .
The last two phases describe **how** this functionality is realized. The CAFCR method is key to transform the customer's objectives into a possible solution which ensures a good system design which complies with the customer needs.

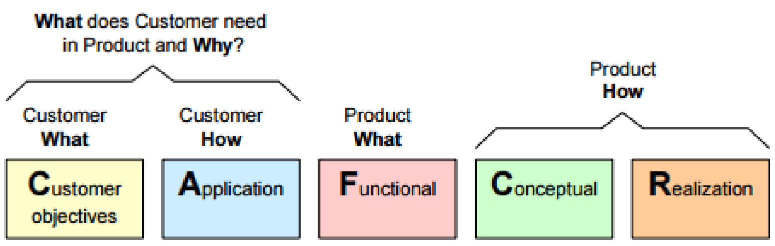

FIGURE 2.1: CAFCR Framework

## 2.2 Customer Objectives

The customer view from the CAFCR framework is used to capture the key drivers of different stakeholders. The combined customer objectives are listed in in Fig. 2.2 and are ranked as performance (top-to-bottom). These can be divided in: safety, comfort, maintainability and user friendly. Safety is a common objective among the stakeholders, Comfort and User friendliness ensure a system which will be pleasurable to use on top of the safety requirements. Finally, the maintainability originates from the desire to improve the ALC in the future, as this is a project that aims to deliver a prototype.

## 2.3 Application view

The customers objectives are further clarified in the application view, which are again shown in Fig. 2.2. This view also deals with the demands from other users, which are in this case the regulation/law , driver, car manufacturer and dealer. For now, the car manufacturer and dealer are assumed to be out of scope, since they do not impose strict functional requirements to the system yet. Two examples regarding the application view will be explained in the following lines. First of all, to ensure safety, the system should be robust, designed in compliance with the standards and the sensor measurements should be reliable. Secondly, driver comfort can be ensured by reducing the workload of the driver, ensure smooth operation and feedback the ALC operating status back to the driver.

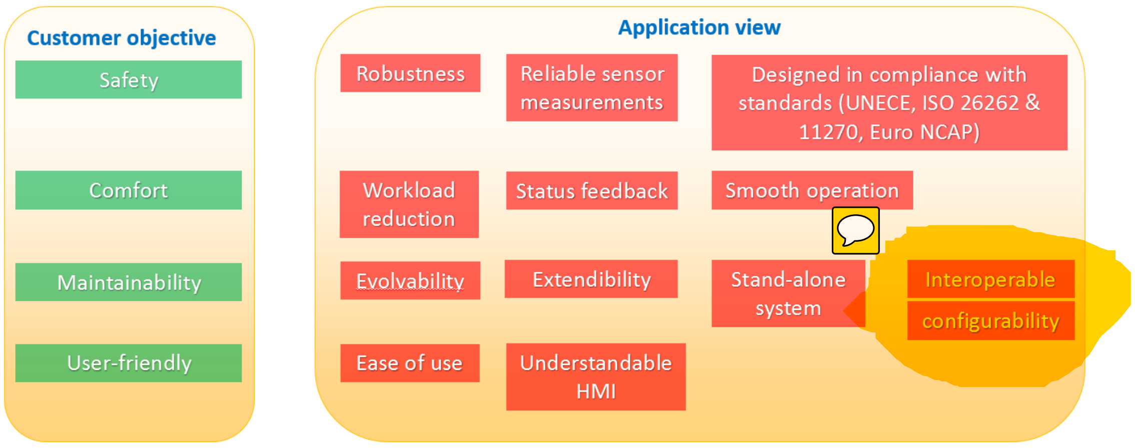

FIGURE 2.2: C and A view within the CAFCR Framework

## 2.4 Functional View

The functional view of the CAFCR approach proposes to see the system as a black box. This means to view it from an external perspective, without knowing how it works (Ref. Figure 2.3) and determining what the system should do, rather than how it should do it.
This view is used to derive the effective functional and non-functional requirements based on the input/output behavior, and by considering interfaces, restrictions, boundaries, exceptions and regulations.

### 2.4.1 Functional Requirements

Each customer objective was broken down in a multitude of desired characteristics of the system in the previous phases as can be seen in Figure 2.2. The following requirements are written in the effort of fulfilling these characteristics.
Subscript F and NF in each requirement are read as functional and nonfunctional requirements respectively.

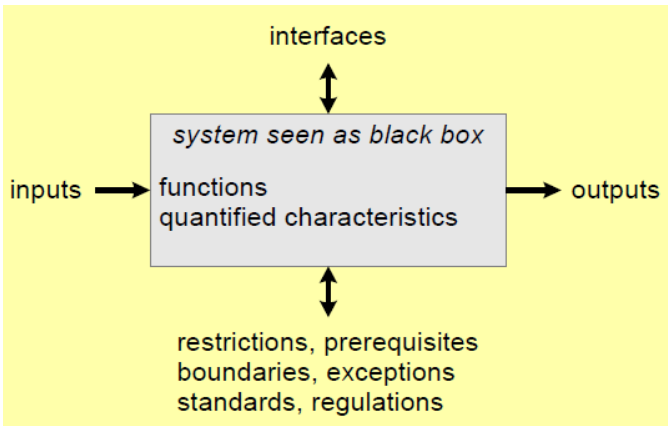
interfaces
*system seen as black box*
inputs
functions
quantified characteristics
outputs
restrictions, prerequisites
boundaries, exceptions
standards, regulations

FIGURE 2.3: Black box view of Systems

**Safety**

1. **F**: ALC shall be default enabled (ON) at every key cycle once Engine is on.
2. **F**: LCA shall act to keep the vehicle at the center of the lane within a tolerance of 0.15 meters.
3. **F**: Additionally, ALC shall trigger a warning at latest when unintentional lane change is detected.
4. **F**: The driver shall be able to overrule the system at any point of time.
5. **F** LCA shall be deactivated in less than 0.5 second (AVG REng time) when :
    - Manually disabled (off) by the user
    - LCA is active currently and driver does the counter steering (opposite to assist torque), with a torque value of more than 0.3 Nm.
    - LCA is active and driver doesn't intervene to the steering wheel within 5 seconds.
    - Driver activates the Turn signal.
    - Driver brakes (more than 50%).
    - Engine is off
6. **F**: ALC shall warn the driver when driver doesn't intervene within 5 seconds when ALC is actively steering the vehicle to the center.
7. **NF**: ALC shall be capable to detect the related component's functional failure and disable ALC with simultaneous warning to the driver.
8. **NF**: ALC shall warn the driver if the system is disabled by the driver.
9. **F:** (Regulation) LKA system shall be available only if vehicle possess Electronic Stability Control system in compliance with regulatory requirements.
10. **F**: (Regulation) ALC shall be operational at least under below conditions while performing unintended lane change.
    - Lane width between 3 to 3.7 m

- Dashed line on one side having width of 0.1 to 0.25
- Solid line on other side with 0.1 to 0.25
- Dry weather conditions
- No precipitation
- Horizontal visibility till 1 km
- Ambient temperature between 5 to 40 deg
- Natural ambient illumination excess of 2000 lux for day light with no strong shadow
- Uniform solid paved surface with consistent slope and no irregularity within a lateral distance of 3.0 m to either side. The minimum peak braking coefficient shall be 0.9
- Wind speed less than 10 m/s
- Slope of the surface between 0 and 1 deg
- Original fitment of tires according to make, model, size, speed and load operating specified by the manufacturer with correct pressure.
- Slope of the surface between 0 and 1 deg
- Default wheel alignment measure set by the OEM

11. **F**: (Regulation) The system must have an accuracy of:
    - 0.1 km/h in longitudinal speed
    - 0.03 m in longitudinal and lateral position
    - 0.1 degrees in heading angle
    - 0.1 deg/sec in yaw rate
    - 0.1 m/sec2 in longitudinal acceleration
    - 1 deg/sec in steering wheel velocity

12. **F**: (Regulation) Once disabled (OFF), the ALC can be enabled only if the time between OFF and new ON exceeds 1 second.

13. **F**: It shall be possible to disable (OFF) the ALC during runtime, but only when ALC is currently inactive.

14. **NF**: The system shall perform cyclic diagnostic tests on its communication interfaces and HW in order to detect latent faults.

15. **NF**: The system should receive data from the vehicle ECU regarding:
    - Vehicle speed
    - Steering angle
    - Steering torque
    - Yaw rate
    - Longitudinal and lateral accelerations
    - Brake pedal position
    - Steering wheel buttons and levers state
    - Wheel speed information

16. **NF**: The system should be able to control the following vehicle parameters:
    - the steering wheel angle
    - steering control mode (safe / high performance mode)
    - HMI display, icons and if available haptic and acoustic signals

**Comfort**

1. **F**: The ALC system shall reduce the workload of the driver by actively steering the vehicle to the center.
2. **F**: The driver must be able to enable/disable the ALC system using a hardware lever/button.
3. **NF**: The switching of control between ALC and the driver shall be smooth.
4. **F**: The steering to counter lateral deviation, shall be in smooth controlled manner and with minimal overshoot.
    - Lateral acceleration < 2 m/s2 while cornering,
    - Lateral acceleration < 0.5 m/s2 while driving straight
    - Lateral jerk < 5 m/s3 overall,
    - Longitudinal deceleration < 3 m/s2
    - If Longitudinal deceleration > 1 m/s2 then, longitudinal speed reduction < 18 km/h
5. **NF**: ALC enable (ON) and disable (OFF) button shall be easily reachable the driver.
6. **NF**: Warning signals shall not distract the Driver.

**Robust**

- **F**: The ALC system shall reduce the workload of the driver by actively steering the vehicle to the center.

**Maintainability**

-

**User Friendly**

-

### 2.4.2 Use cases

## 2.5 Conceptual view

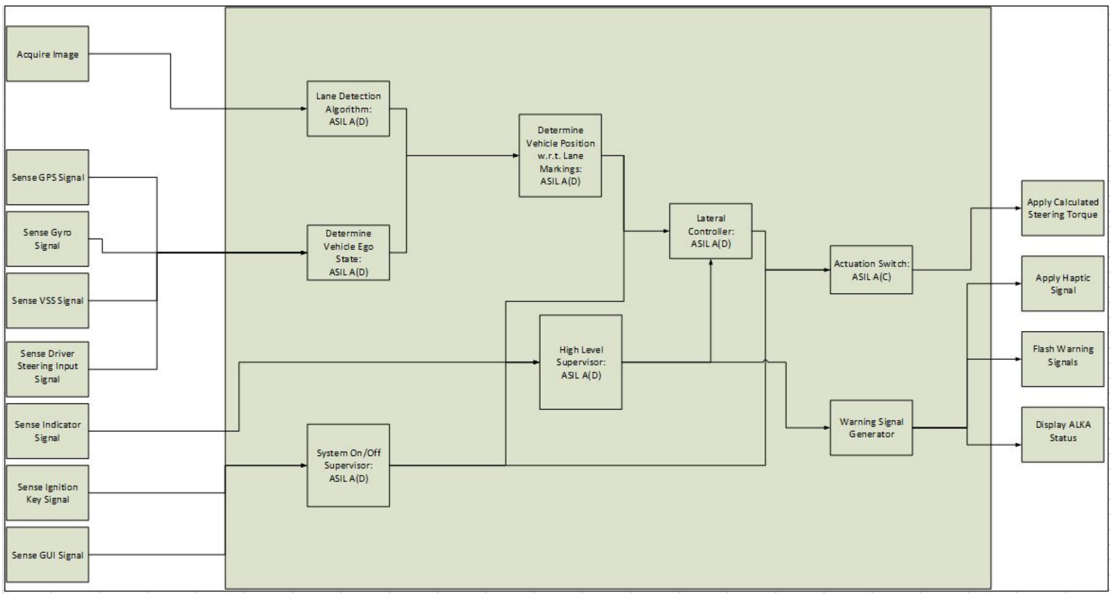
Acquire Image
Sense GPS Signal
Sense Gyro Signal
Sense VSS Signal
Sense Driver Steering Input Signal
Sense Indicator Signal
Sense Ignition Key Signal
Sense GUI Signal
Lane Detection Algorithm: ASIL A(D)
Determine Vehicle Ego State: ASIL A(D)
Determine Vehicle Position w.r.t. Lane Markings: ASIL A(D)
Lateral Controller: ASIL A(D)
High Level Supervisor: ASIL A(D)
System On/Off Supervisor: ASIL A(D)
Actuation Switch: ASIL A(C)
Warning Signal Generator
Apply Calculated Steering Torque
Apply Haptic Signal
Flash Warning Signals
Display ALKA Status

FIGURE 2.4: Basic Functional Architecture of the system

# Chapter 3

# Hardware Architecture Design

This chapter aims to describe the design of the hardware architecture of the ALC system, following all the steps which were performed to propose the final solution.

The design started from the Original Architecture (Figure 3.4), which was analyzed in detail from the performance, safety and practical realization point of view. This analysis lead to the proposal of alternative HW architectures which are compatible with the requirements. A selection between these will be done in the next phases.

## 3.1 Component specification

The aim of this section is to briefly describe the HW components of the architecture that are key to the implementation of ALC. Figure 3.4 shows the original architecture with all the components which are available for the implementation phase of the project.

The selected components are the following:

- Processors:

  Real-time target PC (SpeedGoat)

  NXP Bluexbox

- Sensors and Actuators:

  TNO MoveBox

  Mobileye C2-270

  HDR Mono Camera

  XSens Mti-200 VRU

The next few paragraphs will briefly describe the most important among these.

### 3.1.1 Speedgoat

Speed Goat provides a range of real time target computers for rapid vision, DSP and control prototyping. The hardware along with I/O modules and drivers can be used with Simulink Real time package for HIL simulations and embedded application deployment. The target computer used will be the Performance real-time target machine from speedgoat which is meant for lab use and has powerful computing power. Speedgoat provides support for a wide range of I/O modules and communication protocols. The harware specification, I/O modules and communication protocols are listed in A

### 3.1.2 NXP Bluebox

Bluebox is a mobile computing platform which can be used for development of ADAS functionalities in accordance with automotive functional safety. This provides a single computing solution for testing and developing ADAS applications which need to process sensor data from different sources and decide control action for the vehicle in real time. It enables autonomy up to SAE level 4 for self-driving cars. The Bluebox has two different processors namely S32V234 automotive vision/sensor fusion processor and a LS2085A embedded compute processor. Both the processors run independent embedded Linux systems. Therefore the main host communication interface is the Linux terminal which can be accessed via UART or Ethernet. The Ethernet is a faster and more reliable connection than UART and is therefore preferred over UART. The Bluebox has a number of different kinds of ports to connect to different kinds of inputs like CAN and other sensor data. The hardware specification for the Bluebox is listed in A.

### 3.1.3 TNO MOVE BOX

The TNO MOVE box, is an interface between the CAN network of the Toyota Prius 3rd generation and the CAN network connected to the external processors and sensors.

The main functions of the of the Movebox are the following:

- Connection between internal and external network
- Safety

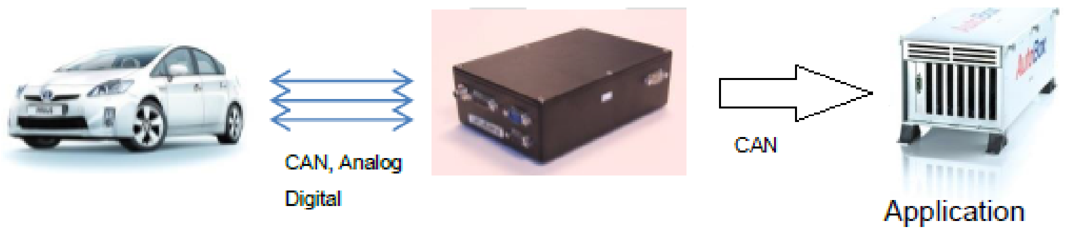

FIGURE 3.1: Movebox description

#### Connection between internal and external network

The interface is capable of transferring a large amount of sensor data from the car network to the external network, and also control actions from the external network to the car (See figure 3.1).

This is a very important task as it enables the testing of the ALC on the selected car without having to know the proprietary CAN messages of the Toyota Prius, but simply knowing the messages of the Movebox. The following list is a selection of vehicle data which will be used in the project.

**Sensor data:**

- User requested steering torque
- Steering angle

- Velocity
- Four independent wheel speeds
- Brake pressed
- Steering wheel buttons and lever
- Lateral acceleration
- Yawrate

**Control Actions:**

- Control of HMI display icons (e.g. warning, Adaptive Cruise Control interface, Lane Keeping Assist interface)
- Controlling steering wheel angle
- Controlling Steering control mode

**Safety**

The Movebox ensures that the connection between the two networks adheres to certain safety requirements. The way the system ensures safety is by providing a set of hard limitations for control parameters and by transitioning to a safe state (user in control) when problems arise.

The following safety functions are relevant for the lateral control:

- Steering torque is limited to 1.5 Nm
- Steering angle is limited to +/- 540 degrees, further limits are given according to the vehicle speed
- Steering wheel rotational speed is limited to 200 rad/s
- The enabling of lateral control is possible only under 25 km/h (it is suggested also to enable it at low steering angles)
- If an error is detected MOVE will be directly switched off and a warning will inform the driver.
- There is an emergency button which disconnects the external network from the internal one

### 3.1.4 Mobileye C2-270

The Mobileye is described on the user manual as a "single-camera-based safety solution for collision prevention and mitigation". In our application, it acts as a multipurpose sensor which detects the car distance from the lanes, and information about lane curvature. Mobileye does not guarantee 100% accuracy in the detection driving lanes, but it is accurate enough to start the development of the system, while the "vision algorithm" is being developed and as a redundancy in case of system failure.

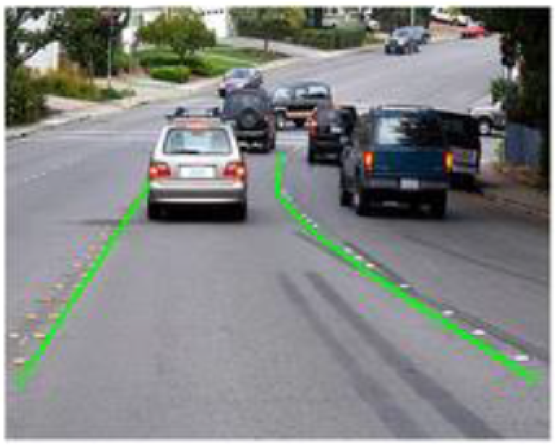

FIGURE 3.2: Lane Detection by Mobileye

## 3.2 Communication strategies

All electronic embedded systems used to control vehicle functions (specifically ADAS) need communications networks and protocols to manage all the process information [1]. Generally, these vehicle control functions receive the input information from network of sensors (camera, wheel speed sensor etc.) and send commands to the control stage or the application software which further interacts with actuators or HMI elements to execute commands. Aspects of communication networks like assurance of message delivery, safety and security of messages, time of delivery, cost, EMF noise resilience, overall robustness and reliability and routing therefore become an important criteria while choosing the right communication network and protocol. Depending on different requirements of complex automotive functions, many communications networks and protocols are developed. Society of for Automotive Engineers (SAE) defined a classification for automotive communication protocols based on data transmission speed, functions that are distributed over the networks. This classification can be referred in [2]. Some of the most commonly used communication protocols and standards in now a day vehicles being CAN, LIN, FlexRay, MOST, SAE J193 [1]. Conventional computer networking technologies (such as Ethernet and TCP/IP) are rarely used [11]. This section explains in short the widely used CAN communication. It also elaborates on Ethernet and Mobile Industry Processor Interface (MIPI). Table ?? further provide the differences between different vehicle communication networks.

### 3.2.1 CAN

One of the first and most enduring control networks, the CAN bus, is the most widely used network[3]. CAN became an ISO standard in 1994 and is now a defacto standard in Europe for data transmission in automotive applications due to its low cost, its robustness and the bounded communication delays [2]. CAN is a multi-master serial bus standard for connecting Electronic Control Units [ECUs] also known as nodes [4]. Each node requires a Central processing unit (microprocessor, or host processor), CAN controller and a Transceiver. The host processor decides what the received messages mean and what messages it wants to transmit. CAN controller which is often part of microcontroller stores the received serial bits from the bus until an entire message is available. It also transmits the bits serially onto the bus when the bus is free.

Transceiver mainly converts the data stream from CAN bus levels to levels that the CAN controller uses and vice versa.

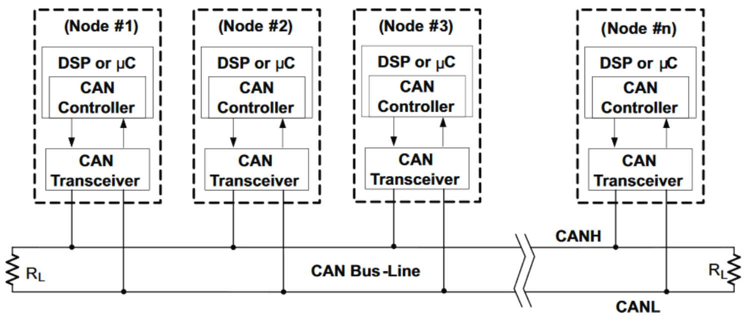

FIGURE 3.3: Details of a CAN bus

Typically the CAN runs at speeds with 125 Kbps (low speed CAN with applications like body control electronics like seat, window etc) and with 500 Kbps (high speed CAN with applications like chassis control systems, lane keep systems). The typical message lengths are message lengths (CAN frames) are typically 50 to 100 bits in length. In CAN, any node may start a transmission when the bus is idle. To guarantee that the transmission of top priority message always remains first, CAN uses decentralized, reliable, priority driven CSMA/CD (Carrier Sense multiple Assess/ Collision Avoidance Detection) control method. CSMA means that each node on a bus must wait for a prescribed period of inactivity before attempting to send the message. CD means that the collisions of the messages are resolved by bit wise arbitration based on pre-programmed priority. The higher priority always wins the bus access. CAN offers an error detection mechanism which detects data transfer errors, interrupts and erroneous transmission with an error flag that indicates the re-transmission of the affected message. It also contains a mechanism for automatic fault localization including disconnection of the faulty controller. More explanation about CAN can be found in [5]. It must be also noted that CAN is a low-level protocol and does not support any security features intrinsically. There is also no encryption in standard CAN implementations, which leaves these networks open to man-in-the-middle packet interception. In most implementations, applications are expected to deploy their own security mechanisms [4].

### 3.2.2 Ethernet

Although Ethernet being rarely used in cars, the first motivation for Ethernet is that is a low cost and a mature technology that offers much more bandwidth that is available today. This is generally of interest for infotainment and active safety systems [2]. The original 10BASE5 Ethernet uses coaxial cable as a shared medium, while the newer Ethernet variants use twisted pair and fiber optic links in conjunction with hubs or switches. Over the course of its history, Ethernet data transfer rates have been increased from the original 2.94 megabits per second (Mbit/s) to the latest 100 gigabits per second (Gbit/s). Systems communicating over Ethernet divide a stream of data into shorter pieces called frames which are transferred to nodes on various parts of the network. Each frame contains source and destination addresses, and error-checking data so that damaged frames can be detected and discarded; most often, higher-layer protocols trigger retransmission of lost frames. Also like the CAN bus, Ethernet is bidirectional, and the speed possible on any individual link decreases as the number of nodes on the system increases. Still, Ethernet can transport data over a link 100 times faster than a CAN bus [7]. Ethernet would be an ideal choice to replace the CAN bus, but Ethernet's cost per node is higher. Hence, it can be used to augment the CAN bus like for mid-bandwidth communications rather than to replace [7]. Although it can be used for data-intensive requirements like cameras and infotainment systems the use is limited as it comes with greater complexity. This is because, typically the video needs to be compressed and decompressed at source and destination to avoid exceeding the bandwidth of Ethernet. However to fully meet the automotive requirements, multiple new specifications and revisions to specification are being done in the IEEE 802.3 and 802.1 groups focusing EMF/RFI emission susceptibility, latency requirement, synchronization and network management requirements [8].

### 3.2.3 MIPI: CSI-2

The bandwidths of today's host processor-to-camera sensor interfaces are being pushed to their limits by the demand for higher image resolution, greater color depth and faster frame rates [9]. But more bandwidth is simply not enough with performance targets that are also being pushed to their limits for the complex embedded systems like camera based ADAS applications. It is needed to have a standard, robust, scalable, low-power, high-speed, cost-effective camera interface that supports a wide range of imaging solutions. The Mobile Industry Processor Interface (MIPI) Alliance is a global, open membership organization that develops interface specifications for the mobile ecosystem including mobile-influenced industries. The Camera Serial Interface (CSI) is a specification of the (MIPI) Alliance. It defines an interface between a camera and a host processor. CSI-1 was the original standard MIPI interface for cameras. It emerged as an architecture to define the interface between a camera and a host processor [10]. The MIPI CSI-2 and MIPI CSI-3 are the successors of the original MIPI camera interface standard, and both these standards are continue to evolve. Latest MIPI CSI-2 was released in September 2014 and it offers higher interface bandwidth and greater channel layout flexibility than its predecessor. It also supports packetized transmission which helps in line management, error detection and error correction. More details can be found in [9].

**Comparison table : different bus network** Below is the comparison between different automotive communication network.

TABLE 3.1: My caption

| Bus | CAN | LIN | FlexRay | MOST | Ethernet |
|---|---|---|---|---|---|
| **Speed** | Up to 1Mbps | 20 kbps | 10 Mbps | Upto 23 Mbps | Upto Gbps |
| **Cable type** | Twisted pair | Single wire | Optical Fiber Dual-Wire | Fiber Optic/ Coax | One or more twisted pair |
| **Cost** | $$ | $ | $$$ | $$$$ | $$ |
| **applications** | Soft real time | Low cost low speed | Hard real time | Multimedia | Camera systems |
| **application examples** | Power train, engine control, Driving assistants | Elelectric seats, power window, rain sensor | Break-by-wire, Steer-by-wire, Emergency systems | Infotainment, navigation | IP Cameras, Infotainment |
| **Control** | Multi-master | Single-master | Multi-master | Time-master | Malti-master |
| **Access control** | CSMA/CDA | Polling | TDMA FTDMA | TDMA CSMA/CA | CSMA/CDA |
| **Error Detection** | CRC Parity bits | Checksum Parity bits | CRC Bus Guardian | CRC System - Service | CRC |
| **Redundancy** | None | None | Yes (2 channels) | None | None |

## 3.3 Architecture Proposals

In this section, the hardware architecture for the ALC system is discussed. The original architecture, as can be seen in Figure 3.4 shows all the hardware components which are relevant for the ALC system. However not all these components are available at the moment and therefore only a selection of these components will be used during the next phase of implementation.

### 3.3.1 Proposed Architecture

The proposed architecture shows the layout of how all the hardware components will be connected when the ALC system is being developed and tested in real time. The Bluebox has the capability to process sensor data from different sources and decide control actions for the vehicle in a particular scenario. However the ‘number cruncher’ part of the Bluebox is still unavailable and therefore a real time target computer will be used during the next phase. Figure 3.5 shows the layout of components in proposed architecture.

### 3.3.2 Implemented Architecture

The implemented architecture shows the layout of how all the hardware components will be connected when the ALC system is being developed and tested in real time during the implementation phase of the project. The vision part of the functional architecture i.e. lane detection will be executed on the Bluebox and the high level, low level

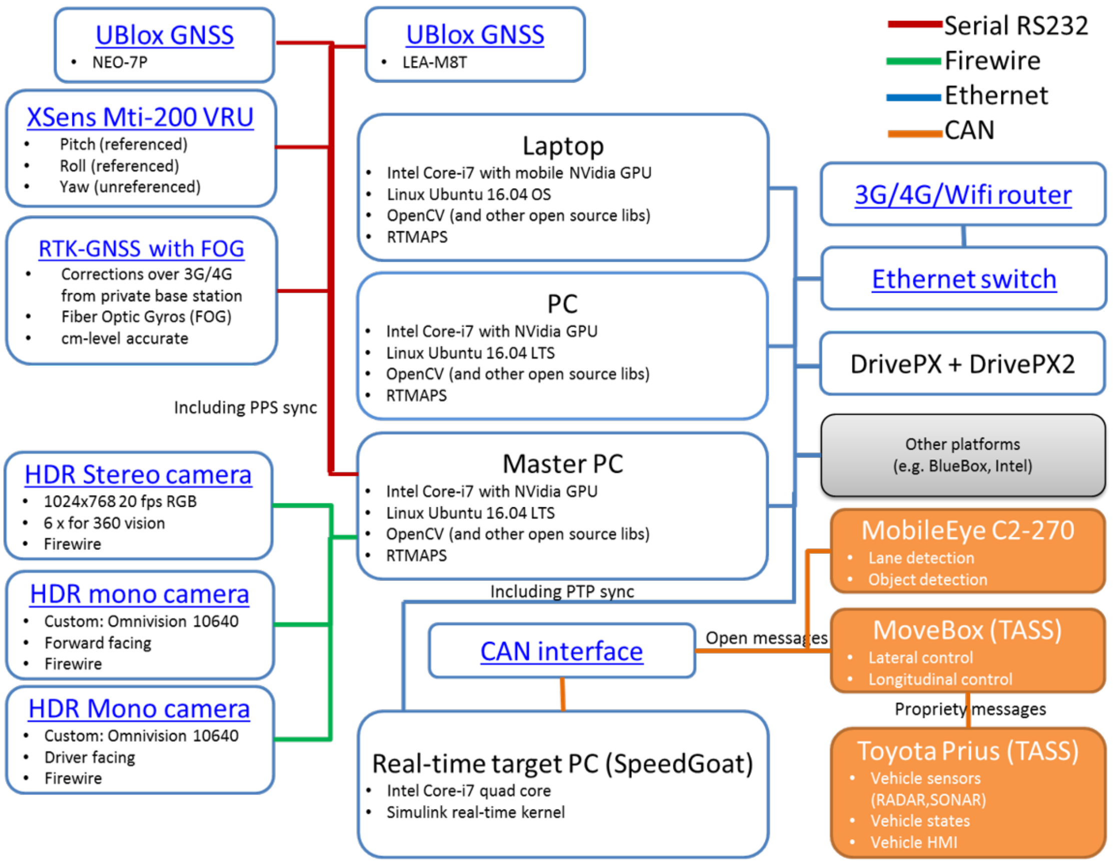

FIGURE 3.4: Available Hardware

control and actuation will be executed on the speedgoat real time target computer. The communication between them is via Ethernet. Figure 3.6 shows the layout of components in implemented architecture. An alternative solution will be where the Bluebox communicates directly with vehicle via CAN bus which is shown by a dotted line in the diagram.

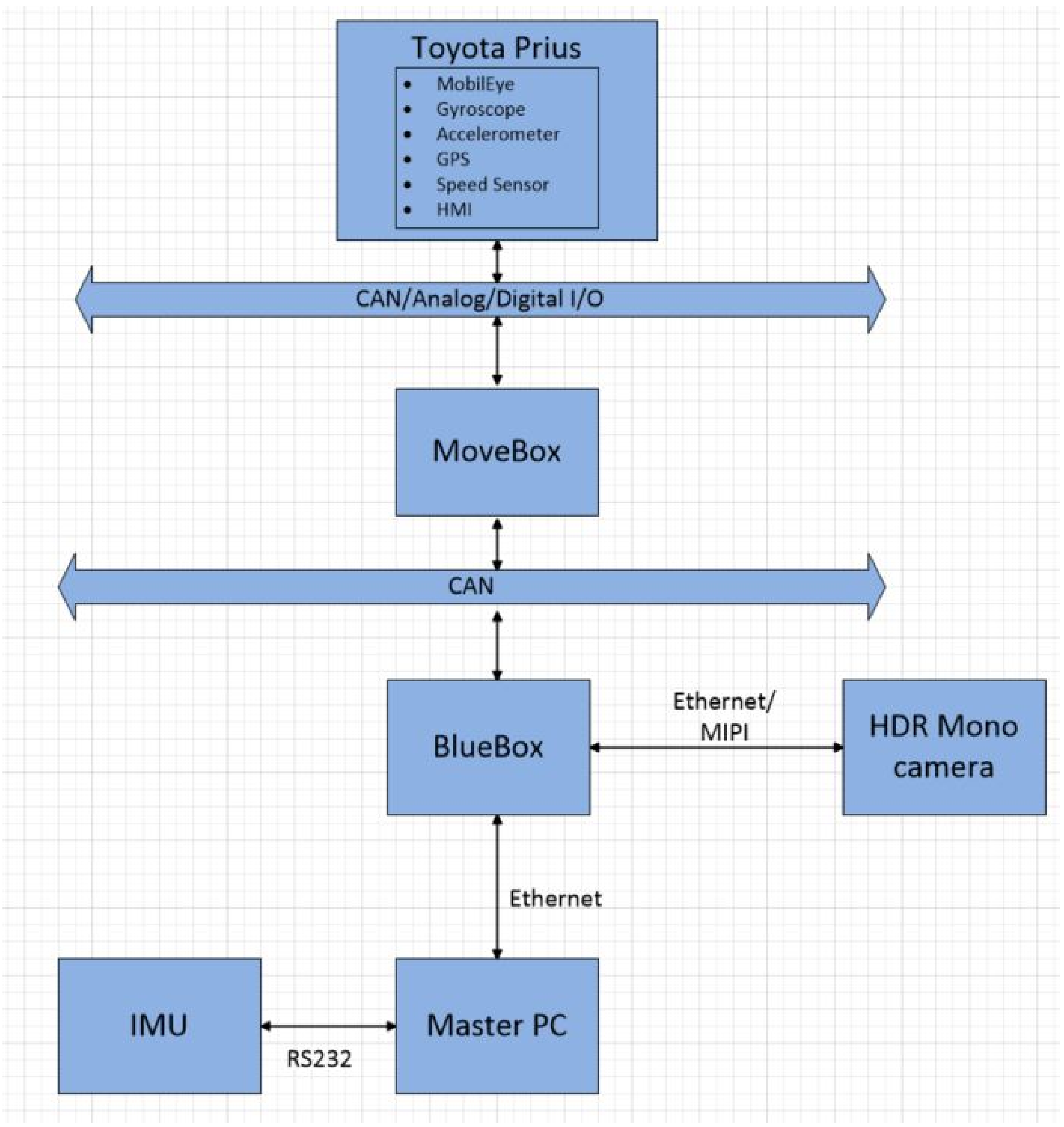

FIGURE 3[illegible] Proposed Hardware Architecture

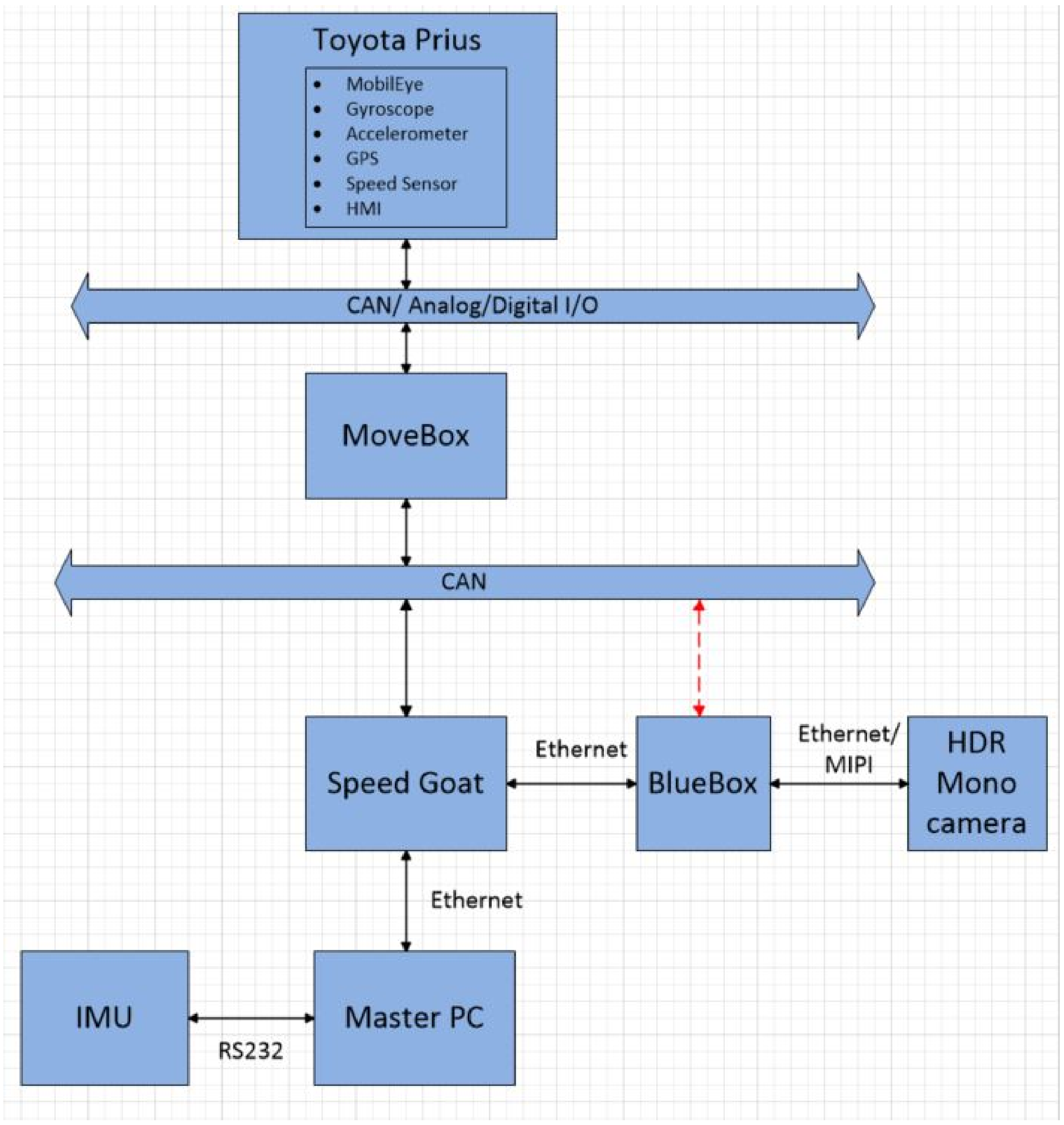

FIGURE 3.6: Implemented Hardware Architecture

# Chapter 4

# Conclusion and Future Work

# Bibliography

[1] Euro NCAP. “TEST PROTOCOL – Lane Support Systems. Version 1.0”. In: *EUROPEAN NEW CAR ASSESSMENT PROGRAMME*. November 2015.

[2] Ian Riches. “Strategy Analytics: Automotive Ethernet: Market Growth Outlook.” In: *Keynote Speech 2014 IEEE SA - Ethernet and IP at Automotive Technology Day*. 2014.

# Appendix A

# Hardware Specifications

## A.1 Speedgoat

**I/O Modules:**

- Analog I/O: AD, DA, DMA, 16-24 bit, with optional configurable FPGAs
- Digital: TTL, RS422, LVDS, MGTs, with optional configurable FPGAs
- PWM generation and capture
- Encoder measurement and simulation (quadrature, SSI, SSI2, EnDat 2.2, BiSS)
- Temperature measurement and simulation: thermocouples, RTD (PT100/PT1000), NTCs
- Strain gauge and pressure measurement and simulation
- IEPE/ICP vibration measurements
- Resistor, potentiometer, and reed relays (SPDT, DPST, SPST)
- LVDT, RVDT, Synchro, and Resolver measurement and simulation
- Cam and crank measurement and emulation
- Shared/Reflective memory (ScramNet, PIORC5565)
- Fault/signal insertion switches

**Communication Protocols:**

- CAN, CANopen, LIN, SAE J1939
- Serial UART (RS232, RS422, RS485), and SDLC/HDLC
- SPI Master and SPI Slave
- I2C Master and I2C Slave
- Real-Time UDP, and Real-Time Raw Ethernet
- EtherCAT Master and EtherCAT Slave
- EtherNet/IP Scanner (Master) and EtherNet/IP Adapter (Slave)
- PROFIBUS Master and PROFIBUS Slave

- PROFINET Master and PROFINET Slave
- Ethernet POWERLINK Master and Ethernet POWERLINK Slave
- Modbus TCP and Modbus RTU
- XCP Master and XCP Slave over CAN and Ethernet
- FlexRay
- MIL-STD-1553 and ARINC-429
- IRIG and Precision Time Protocol (PTP), IEEE 1588
- USB WebCam and CameraLink

**Hardware Specifications:**

**Housing:**

- Enclosure: 4U 19"-compatible aluminium chassis
- Color: Silver powder-coated, natural aluminium
- External dimensions:
  - Height: 177.8mm (4U)
  - Width: 440mm, 480mm (including rack mounts)
  - Depth (standard): 360mm (400mm including handles) Depth (deep option): 440mm (480mm including handles)
- Weight: 12kg (excluding I/O modules, cables, and terminal boards)
- Power supply: 400W, 100-240V, 50-60Hz, fan-less, zero-noise
- Fans: Two at rear (outtake), high quality, low-noise Papst fans
- Handles: 2 for desktop use , 2 for rack installation
- Certification: CE and FCC certified

**Mainboard and CPU:**

- Processor: Intel Core i3 3.5GHz
- Form factor: ATX
- Chipset: Intel C216
- Bus: PCI, 32-bit/33MHz
- Memory: 4096MB DDR3 RAM
- Graphics: Intel HD Graphics 400P onboard
- USB: 4 x USB 3.0 and 1 x USB 2.0 at front, 6 x USB 2.0 internal

- Ethernet: 2 x Gigabit at front
- Serial Ports (for baud rates up to 115kb/s only): 1 x RS232/422/485 at front,1 x RS232/422/485 and 4 x RS232 internal
- Keyboard and mouse: 1 x PS/2 at front
- BIOS: American Megatrend Inc. (AMI)
- Number of slots for I/O modules: 3 PCI, 4 x PCIe and 1 x Mini PCIe

**Drives:**

- Main Drive: 1 x 60GB SSD

**Power:**

- Power inlet: AC 100-240V, 50/60Hz, at rear
- Power switch at rear
- Secondary power switch at front
- Reset button none (secondary power switch)
- Power LED at front (combined with secondary power switch)

**Environment:**

- Temperature: 0 deg to +60 deg (operating)
- Humidity: 10-90 %, non-condensing

**Software:**

- OS / RTOS: FreeDOS / Simulink Real-Time kernel
- Development computer: Utilities for kernel transfer, I/O drivers and Simulink test models

## A.2 NXP Bluebox

**Hardware Specifications:**

**Vision Processor : S32V234**

- 4 x ARM Cortex- A53 CPU, up to 1.0 GHz core speed
- ARM Cortex-M4 at 133 MHz for IO control and AutoSAR OS
- Accelerators: Dual APEX-2 image processing engine, ISP, 3D GPU
- Main Memory: 2 x 2 GB LPDDR2
- 1 x NOR flash min. 512 Mbit – Hyperflash
- PCI Express: 1 x PCIe Gen2

- Ethernet: 1 x GB Ethernet
- Additional I/O Interfaces: VIU, SDHC, UART, HDMI, FlexRay, FlexCAN, SIPI, LIN, Gyroscope + Digital compass, JTAG and Trace port

**Embedded compute processor: LS2085A**

- LS2085A CPU: up to 8 x ARM Cortex- A57 CPU, up to 2.0 GHz core speed
- Accelerators: Advanced I/O Processor, Security Engine
- Main Memory: 2 x 72b (ECC) DDR controllers, up to 2.1 GT/s
- 128 MB NOR flash, 8-bit 2 GB SLC NAND flash
- PCI Express
- Ethernet
- USB 3.0
- SATA

# Appendix B

# Appendix B: Glossary

| | | |
|---|---|---|
| ALC | : | Active Lane Centering |
| LKA | : | Lane Keep Assist |
| ALKA | : | Active Lane Keep Assist |
| LDW | : | Lane Departure Warning |
| Euro NCAP | : | European New Car Assessment Programme |
| ASIL | : | Automotive Safety Integrity Level |
| FSG | : | Functional Safety Goal |
| SG | : | Safety Goal |
| FSR | : | Functional Safety Requirement |
| TSR | : | Technical Safety Requirement |
| DFSR | : | Decomposed Functional Safety Requirement |
| HARA | : | Hazard Analysis and Risk Assessment |
| FMEA | : | Failure Mode Element Analysis |
| LSS | : | Lateral Support System |
| VUT | : | Vehicle Under Test |
| HMI | : | Human Machine Interaction |