POLYTECHNIC OF TURIN

The Department of Electronics and Telecommunications (DET)

Master of Science Thesis in Electronic Engineering (Embedded Systems)

ALGORITHM ANALYSIS FOR AN AUTOMOTIVE ADAPTIVE FRONT LIGHT SYSTEMS

Supervisor

Prof. Luciano LAVAGNO

Co. Supervisor

Dr. Massimiliano CURTI

Candidate

Soroush ARAB

September 2020

Abstract

One of the crucial components of a car is headlights. Lights are essential to safety on the road. They allow the driver to see the road ahead, even at night or in obscure-light conditions. They also ensure that others can see the vehicle. Generally, front vehicle lights point only forward, and drivers turn them on/off and switch on the high-beam headlamps manually. In this thesis, a model-based design approach was used to analyse and propose a prototype for an adaptive front lighting system that can automatically turn off high-beams and control the intensity of low-beams when detecting oncoming traffic. Furthermore, having a high visibility range provides safe driving conditions. The fundamental concept behind adaptive headlights is the capability of swivelling from side to side and up and down. This way, the driver can see throughout the turn.
The study was mainly carried out on STMicroelectronics development boards, which were collected to develop and simulate an automotive adaptive front light system for development purposes. These are labelled as "AutoDevKit boards" in this outline, which includes the "AEKD-AFL001", which represents logic and driving hardware for prototyping, testing, and development plans. The assortment involves two stepper motor control boards, and a four-channel LED driver board, a command board with MCU, a connector board with a FAN switchboard, and an integrated connector board for wiring arrangement. The configuration of boards for developing firmware has been done within "SPC5-STUDIO" as codes generator, expeditious source configurator, and Eclipse development environment for "SPC5 MCUs". Debugging and programming were done through the "SPC5-UDESTK" as an interface to run and test the firmware. The "ValueCAN 4-2" was used to monitor and transmit on CAN networks to generate hardware simulations for analysis. Additionally, the Vehicle Spy software was used as a single tool for diagnostics data acquisition and testing, while the Adaptive Front Light System CAN bus monitoring by creating a graphical simulation panel. The first step was to study each board for understanding, cognition design, functionality, communication protocols, and limitations. Next, drivers and communication protocols such as SPI and CAN were configured according to the aspects of the algorithm. Further, databases were created from the values of sensors and radars for transmitting and retransmitting the signal to simulate commands, and graphical panels were designed to demonstrate the results.
This algorithm was designed to turn on daytime running lamp automatically whenever the switch mode is "Drive", and it is dark enough to require lights. The automatic high-beam and low-beam lights help resolve two problems. They turn

off or reduce the intensity of bright lights as required to avoid blinding the occupants of oncoming cars. They also turn the high-beams on when the street ahead is not visible enough; they assist drivers who do not use them by turning on automatically in order to provide broader illumination. Similar to automated lights, the system is driver-selectable. The system uses a forward-facing radar which recognises lights—not just expected lights but also the tail lights of vehicles ahead and streetlights or other light sources that indicate the driver is in an area that does not require high beams. As soon as excessive lights are identified, the system turns the high-beam lights off, then switches them back on once the light fades. The low-beam front lights closely follow the rotation of the steering wheel; they change their position according to the direction of the steering wheel. The Follow-Me-Home feature keeps headlights active for a few minutes after the engine is turned off and the doors are closed in order to illuminate the path. The algorithm provides an alternative solution to predict and illuminate passengers' intended paths.

At the end of this research, can understand the vehicle headlight should not be a passive tool that can only be switched on/ off. It should be able to adapt to the environment to increase safety in low visibility conditions. I have illustrated the adaptability of the headlight for numerous duties; Allowing drivers to use high beams without glaring any other driver on the road. Allowing drivers to see better in curvature, and allowing better illumination of road lanes, sidewalks and dividers. It will be done through curve-adaptive lights. Provide essential safety for passengers by configurable and adaptive "Follow-Me-Home".

The main focus of this thesis was analysing an algorithm that operated with step motors. Technology is continuously moving forward, and one of the aims in the area of lighting systems is to minimise moving parts. One alternative is using adaptive lights in the form of LED pixels, which could reduce power consumption and motor delay.

Another modification to improve the system could be done through faster communication protocol such as Automotive Ethernet or CAN - FD. The data over CAN is transmitted in frames. The receiving nodes send an acknowledgement flag when they receive the frame. As this acknowledgement is sent into the transmitted frame, the sender receives an in-frame response after successful transmission. CAN - FD solves this challenge by applying two separate frames to transmit the real data and the acknowledgement data.

Keywords: AFL (adaptive front-lighting system), headlamps, Sensor, Automobiles, CAN communication protocol, Radars, Follow-Me Home,Intrepid Control Systems, STMicroelectronics, Teoresi group.

This thesis is dedicated to my parents.

"For their endless love, support and encouragement"

Acknowledgements

I would like to acknowledge my indebtedness and give my most empathetic thanks to my supervisor, Professor Dr. Luciano Lavagno, who made this work feasible. His friendly guidance and skilful advice have been invaluable during all stages of the thesis.
This work is the outcome of months of research, collaborations, and support from many people. Therefore, I would like to thank Teoresi group for allowing me to intern in this fantastic company.
I would also yearn to express my gratitude to Professor Massimiliano Curti for extended discussions and valuable suggestions which have contributed significantly to the enrichment of the thesis. I would appreciate Dr. Andrea Garino for his guidance and useful instructions.
And foremost, I have to thank my parents for their love and support throughout my life. Thank you both for giving me the strength to reach for the stars and chase my dreams. My lovely sister, deserve my wholehearted thanks as well.
To all my friends, thank you for your understanding and encouragement in my numerous moments of crisis. Your generosity makes my life a wonderful experience. I cannot list all the names here, but you are always on my mind.

Turin, September 20th, 2020
Soroush Arab

Table of Contents

List of Tables

List of Figures

Acronyms

Daylight
Daytime running lamp

AFL
Adaptive Front Light system

FMH
Follow-Me-Home

RPM
Revolutions Per Minute

DBC
DataBase Container

DSPI
Serial Peripheral Interface

DMA
Direct Memory Access

CAN
Controller Area Network

ST.W
Steering Wheel

SDL
Short Distance Light

FSM

Finite State Machine

IRQ

Interrupt Request line

SPC58

AEK-MCU-C4MLIT1

Chapter 1

Introduction

The 21st centenary is labelled as the period of technological evolution. With an extreme rise in population, automation is maturing the need of the hour to make life more convenient and straightforward. Due to the improvement and growth in the area of automation and embedded system, the perception of a smart car has become very common. Smart cars are modernising bents in traditional automobile manufacturers. Companies across the globe have been spending a considerable amount of sources on the production and design of smart cars. Every technological development needs to overcome certain obstacles, and hence, in this project design of an adaptative front light system for smart cars is proposed by using ST development boards and CAN communication protocol for simulation ultrasonic park sensor, Radars, dashboard lights, and commands panel.
One of the crucial segments of a car is headlights. Lights are essential to safety on the road. They guarantee that driver can see the road forward, even at night or during low-light conditions. Furthermore, they also ensure that others can see the vehicle.
Generally speaking, all vehicle lights pointed only forward, and the driver should turn them on/ off and run the high-beam headlamps all by themself.
Now, automatic headlamps are typical, and indeed many entry-level types can automatically change the high-beams when they recognise oncoming traffic. Moreover, amongst many features, bending the light happed when the driver is turning so he/ she can see throughout the corner. The fundamental concept of adaptive headlights is a pivot from side to side, or up and down. Without adaptive headlights, the road forward is not enough brightened till the driver is turned the corner, and he/ she is travelling direct again. Considering adaptive headlights swivel toward the area the driver is turning, he /she has a more remarkable view of what is ahead. The driver can see problems, such as a hindered automobile or someone deciding to cross the street, ere standard lights would have informed them. Some vehicles that had swivelling lights utilised a mechanical section to the steering wheel to

turn them. On new vehicles, they are affected by tiny step motors, that respond to data produced by sensors which define how quickly the car is going and whence far the driver has turned the wheel. On unusual cars, auxiliary cornering lights force come on to add additional lighting to the side of the street.[1] This project is considered automatic daylight appear on whenever the switch is in the "Drive" condition when it is obscure enough to need lights could utilise a photoelectric sensor that is typically installed on top of the dash.[2] Most automated lights are set by the company, although some let drivers adjust the light sensitivity to how dark it need be before they come on. Most do not begin when it is almost bright. Automatic high-beam lights help resolve two problems. They turn to off the more brilliant lights as required to evade blinding occupants of oncoming cars. At the same time, following they turn the high-beams on during the road forward is dark; they penitentiary help drivers who do not ever think to turn them on also when they can provide broader illumination forward the path. Similar automated lights, the system is driver-selectable. Most generally, “the driver leaves the high-beam switch on all the time and activates a secondary switch for the automatic function. The system uses a forward-facing radar, typically installed in or near the rearview mirror” [3]. The Radar recognised lights, not just expected lights but additionally taillights of vehicles ahead, and streetlights or other brilliance that indicates the driver is in the town and does not require high beams. As shortly as other lights are distinguished, the system changes the high-beam lights off, as well as, when switches them back on repeatedly once the light fades.
Technology is continually progressing forward, and one of the goals with lighting systems is to reduce as numerous moving parts as feasible. Audi has computer-controlled LED lights that able selectively exterminate section of the lights[4], pinpointing the beams so precisely that the driver can have the advantage of high-beam lights but also proposed that they do not harm other drivers. Other companies are also developing their lights. Hyundai is working on adaptive lights by pixels that form horizontal beams; segments of these can remain lit or turn off to perform the "bending" effect of adaptive headlights without physically moving the lights. Furthermore, these developments will be significant, indeed when automobiles drive themselves. Likewise, if they use lights and sensors and will not have to see each other, they will still have to be visible to walkers. Modern headlights employ arrays of LEDs that can be programmed to pinpoint wherever light moves. In the case of Audi’s Matrix Beam Lighting and BMW’s Dynamic high-beam, for instance, when a different car approaches, headlight arrays dim only the specific LEDs that glow into the approaching road.[5] These methods allow drivers to enjoy the advantages of light illumination without blinding the oncoming driver.[6] In the following, all the possible features analysed and explained how to implement according to the limitation of Autodevkit experimental boards.[7]

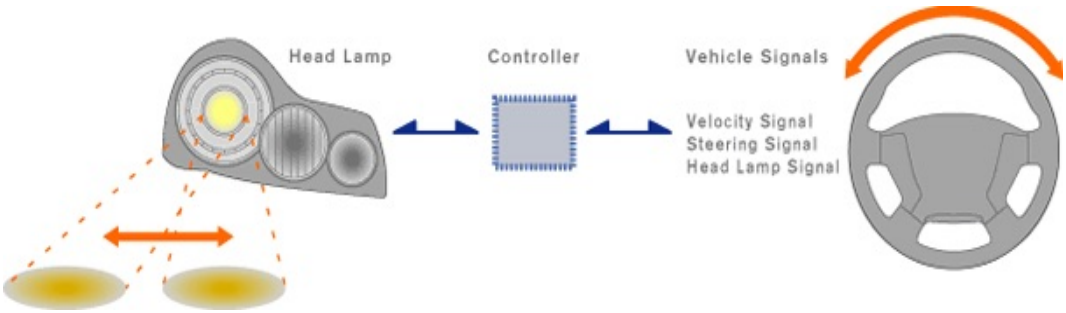

Figure 1.1: An Adaptive Front Light system

Figure 1.2: Follow Me Home feature

Chapter 2

Methodology

This thesis is done based on the model-based design that gives an effective method for establishing a standard framework for interaction throughout the design method while supporting the development cycle [8]. The model-based design of Adaptive Front Light system, was followed these four steps by primary and secondary data. The primary data was collected by myself for this specific purpose, as well as; the secondary data was collected from books, articles, sources from STMicroelectronics company, and Teoresi Group.

1. Modelling a plant and aims according to the market of automotive
2. Analysing a system for the plant
3. Simulating the plant and controller
4. Integrating all these states by using the controller.

2.1 Proposed design:

- Design a system that can detect the rotation of the steering wheel and, according to that, change the direction of lighting systems (Low-Beams).
- Adapt the lighting systems according to the different conditions, traffic jam and high way.
- Design possible features such as Follow Me Home, Welcome Light, warning light system.
- Design a graphical panel to demonstrate and simulate the system.

The block diagram of the proposed Adaptive Front Light System is shown in Figure 2.1. To get a clear viewed on the road, and obstacles on the way at night time along the curved road, it is essential to turn the light along that direction. To do that, AEK-MCU-C4MLIT1 has received a direction analogue value generated by a sensor in the steering wheel and converted to digital by a SAR ADC converter. According to the direction, AEK-MCU-C4MLIT1 is sent a command within an SPI communication protocol to the appropriate motor driver for rotation. Controlling LEDs happens within AEK-MCU-C4MLIT1 that receives a command by CAN communication protocol. The AEK-LED drivers get an SPI command to switch LEDs or change the lighting intensity by PWM pulse width.

2.2 Hardware requirement:

- "**AEK-MCU- C4MLIT1:** MCU discovery board for SPC5 Chorus 4M automotive microcontroller with CAN transceivers.
- **AEK-MOT- SM81M1:** Stepper motor driver board for automotive applications.
- **AEK- LED-21DISM1:** Digitally controlled LED driver board.
- **AEK- CON-5SLOTS1:** 5 slots connector board.
- **AEK-CON- AFLVIP2:** Connector board.
- **ValueCAN 4-2:** To create hardware simulations for network analysis."[3]

2.3 Software requirement:

- "**SPC5-STUDIO:** Code generator, expeditious source configurator, and Eclipse development environment for SPC5 MCUs.
- **SPC5-UDESTK:** An interface to run and test the firmware for debugging and programming.
- **Vehicle Spy:** A tool for diagnostics data acquisition, testing, and the Adaptive Front Light System CAN bus monitoring.
- **AEKD-AFL001:** AutoDevKit adaptive front lighting kit.
- **STSW-AFL001:** AutoDevKit adaptive front lighting kit firmware.
- **Microsoft Visual Studio**."[3]

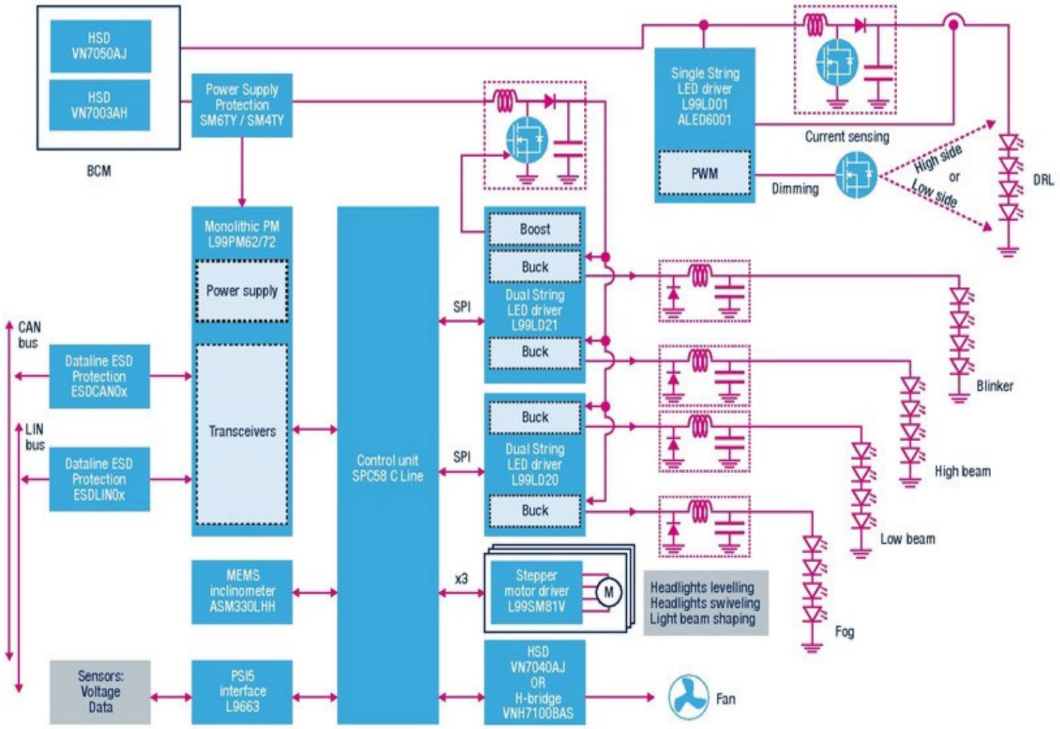

Figure 2.1: Block Diagram of the proposed design

Part I

Background

Chapter 3

Hardware overview

3.1 The system control AEK-MCU-C4MLIT1 board

The AEK-MCU-C4MLIT1 board is created for Automotive and Transportation utilisation and additional applications requiring automotive protection and security levels, also thanks to the Hardware Secure Module (HSM) embedded on the SPC58EC80E5 MCU, which renders it compliant with the EVITA Medium standard. The different interfaces, such as Ethernet controller, CAN-FD, FlexRay, LIN and UART, ADC, and JTAG port do the SPC58EC-DISP, a tool to assess the microprocessors and to improve and then debug applications. An integrated programmer debugger allows debugging and programming the microcontroller. The same section allows enabling a USB Virtual COM port, Figure 3.1.

Overview
"The 32-bit SPC58EC80E5 microcontroller has these features:

- Two main dual-issue 32-bit CPUs built on Power Architecture technology, working up to 180MHz.
- 4224 KB (4096 KB code Flash + 128 KB data Flash) on-chip flash memory.
- 384 KB on-chip general-purpose SRAM (+ 128 KB local data RAM: 64 KB included in each CPU).
- 8 DSPIs available: 3 DSPIs are allocated in AFL to connect the LED driver board and the two stepper motor driver boards.
- 8 (Programmable Interrupt Timer) PIT channels available.
- 2 x 32 eMIOS channels available.

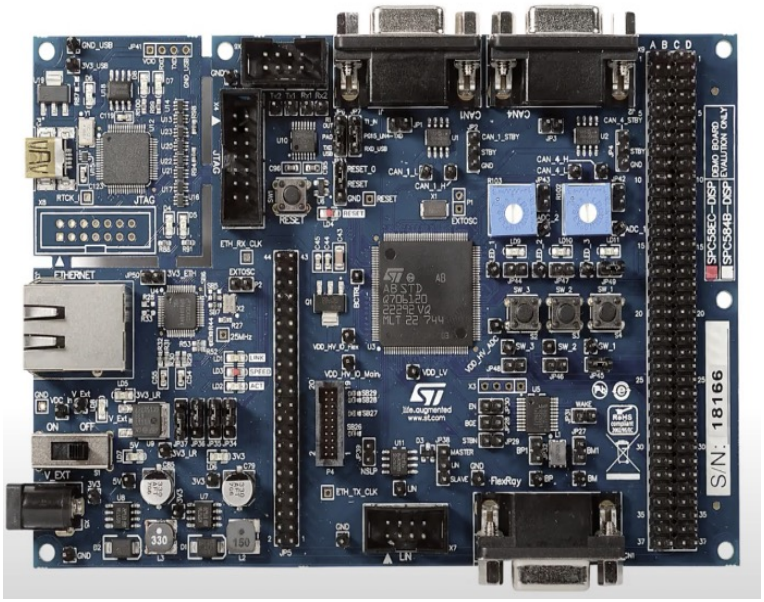

Figure 3.1: SPC58EC-DISP top side

- Up to 96 channels SARADC available: 1 channel allocated for steering wheel position.

- 8 MCAN available: 1 MCAN used to connect and control an instrumental dashboard." [9]

3.2 Power supply

The voltage input range is 12VDC, and the output voltage is set to 5 V, and 3.3 V. Figure 3.3 and the buck converters are shown in Figure 3.4 used to generate 5 V and 3.3 V, respectively.

3.3 Integrated section of Programming and Debugging

This section permits the user to program the microcontroller, to build and to debug the software applications. The core of that is a single-chip USB to dual serial / parallel ports which the device controls the JTAG signals while the second part implements a UART communication channel.

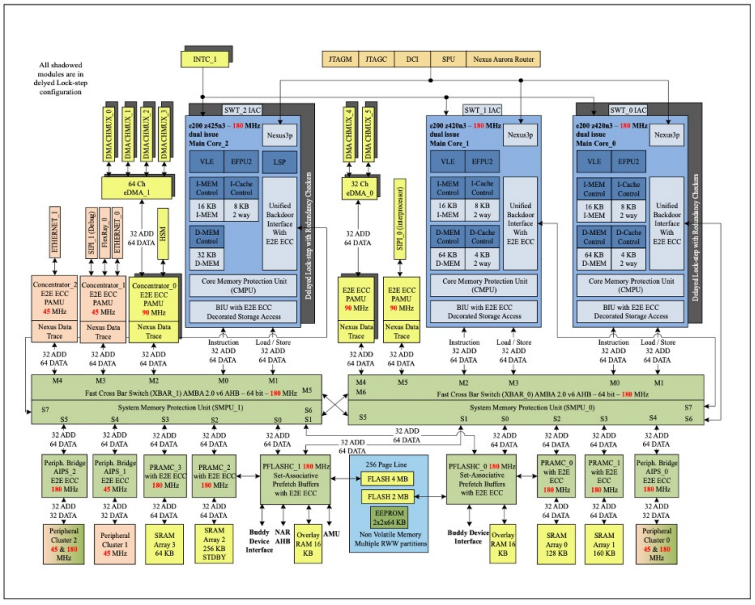

Figure 3.2: SPC58ECx Block diagram

3.4 CAN HD and ISO CAN-FD

Two-channel for CAN are available, and both are based on a fast transceiver; the implemented is compatible with CAN-FD operation standard.

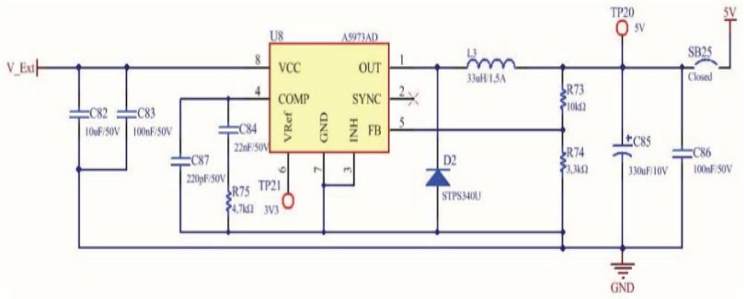

Figure 3.3: Buck converter [12 -5] V

3.5 The LED driver AEK-LED-21DISM1 LED board

The AEK-LED-21DISM1 LED driver board is developed for high illumination automotive LED front lighting applications. It can manage four LED strings

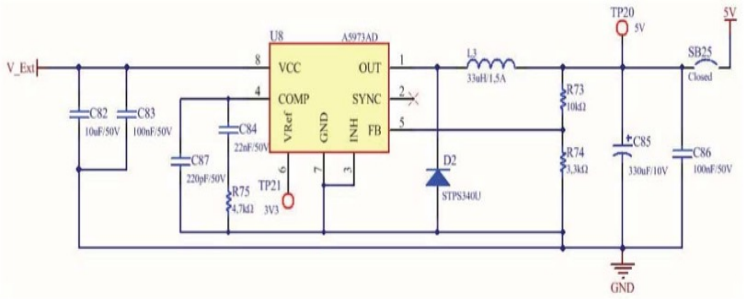

Figure 3.4: Buck converter 3.3V

within two embedded LED drivers with a boost controller and two embedded buck converters.[10] The boost controller component in all LED driver includes a high current gate driver for an external N-channel MOSFET and delivers a fixed output voltage to two integrated buck converters. The boost controller of two devices can be stacked to provide dual-phase achievement for the high power applications, with an interleaving pattern for developed input current ripple and current load splitting. The buck converter integrates an N-channel MOSFET driven through a bootstrap circuit. The board works with SPI communication with an outer microprocessor by a 12 male pin connector, and a single Limp Home Mode implemented in the driver ensures that active communication with the MCU is regularly monitored. Was used the AutoDevKit library for the SPC5-Studio to install board firmware drivers that are regular with SPC5-family microprocessors.

Overview

The "AEK-LED-21DISM1 LED driver board has these features:

- Two embedded L99LD21 high power LED drivers capable of supplying four independent channels.
- Output current up to 1.69 A for every channel
- Input working voltage limit from 5.5 V to 24 V
- Programmable through SPI, allowing accurate LED current setting
- Protections and diagnostics for output short-circuit and open-load, overtemperature, and battery Undervoltage.
- Automotive Limp-Home Mode support
- Board designed according to the AutoDevKit initiative." [11]

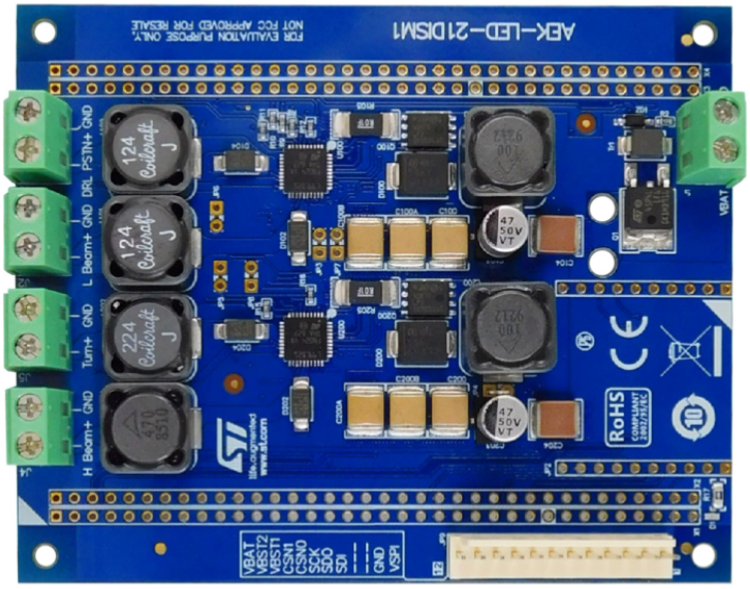

Figure 3.5: AEK-LED-21DISM1 LED driver board

The performance of the board is observed and controlled within the control, status registers in the L99LD21 LED drivers, which can be set or read within the external microcontroller within the fast SPI interface.

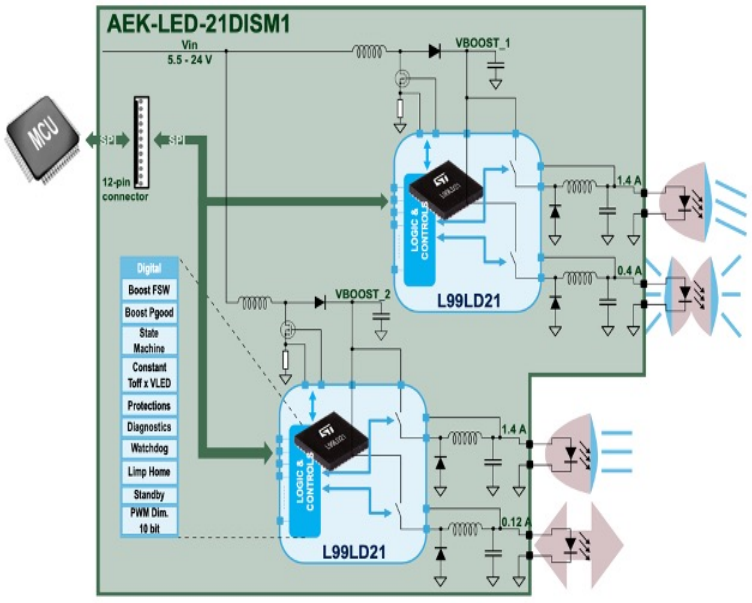

Figure 3.6: AEK-LED-21DISM1 block diagram

3.6 The stepper motor driver AEK-MOT-SM81M1 board

The AEK-MOT-SM81M1 evaluation board is designed to run one bipolar stepper motor. The range of stepping resolution is four full steps to 64 micro-steps and with coil voltage monitoring for stall detection. The stepper motor driver L99SM81V manages timers, counters, a mentioning table and situation register that are manipulated by an external MCU via SPI.[12] The driver also controls the

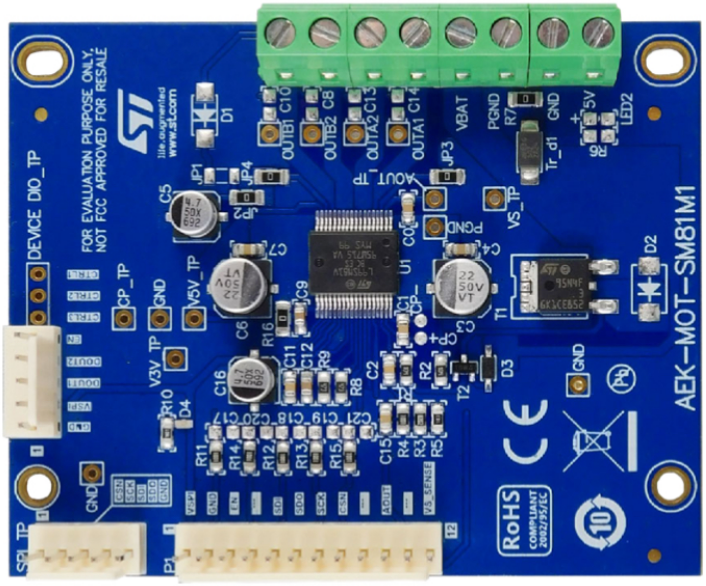

Figure 3.7: AEK-MOT-SM81M1 motor driver board

output voltages over the phase terminals in order to detect and flag critical motor stall events that can compromise motor performance and control.

Overview
The functionality based on "the L99SM81V programmable stepper motor driver are:

- with micro-stepping and hold functions
- BEMF monitoring for stall detection
- programmable configuration via SPI
- 5 V internal linear voltage regulator

- Board reverse battery protection with STD95N4F3 MOSFET, which can be substituted with
- two optionally mounted diodes and a jumper
- Input working voltage level from 6 V to 28 V
- Output current up to 1.35 A"[13]

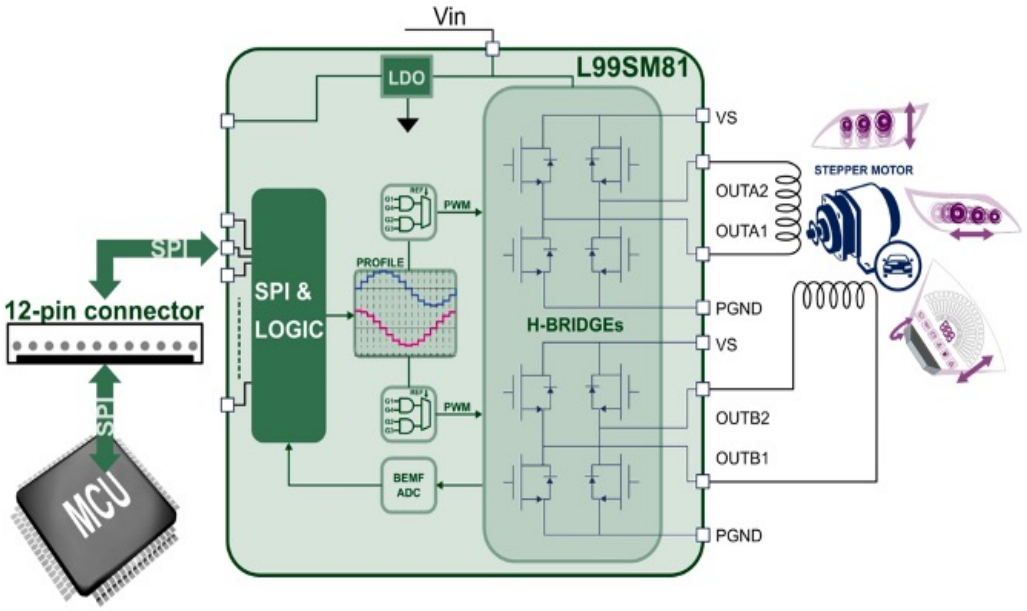

Figure 3.8: The stepper motor diagram

3.7 The fan EV-VN7050AS control board

In the AFL system, a cooling fan is used for the LED headlight. The board is based on the VN7050AS single-channel high-side driver. The device is generally powered at 12 V but has a running range of 4-28 V. The VN7050AS also integrates high-level stability functions such as load current limitation, active overload management through power limitation, and over-temperature shutdown. "The device's current limitation value is 30 A (typical) and has meagre standby power. The board can accept a sense resistor for load current sense."[14]

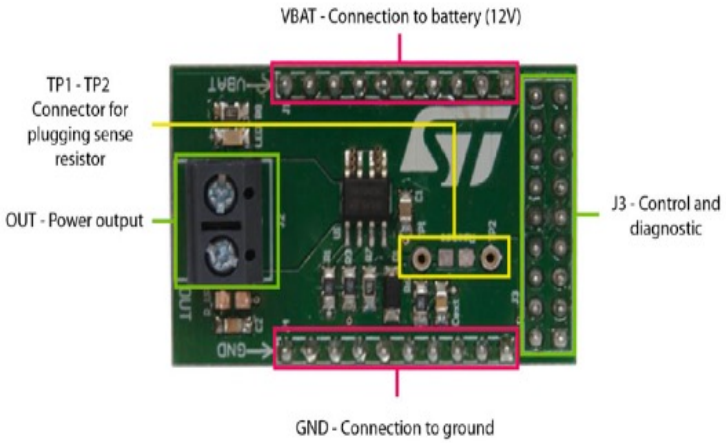

Figure 3.9: EV-VN7050AS high-side driver board

3.8 The connector AEK-CON-AFLVIP2 board

The AEK-CON-AFLVIP2 board is designed for connection within the boards in the Adaptive Front Light system by ST. "The female connector for the MCU control board is located on the side of the connector board. The 12 V supply input for the whole system can also be supplied within this board."[15] The design of the connector board allows the increasing extra functions to the AFL system within the 4x37 male pin connector, which has the same electrical connections as those on the MCU board.[16]

3.9 ValueCAN 4-2 comunication protocol

The ValueCAN 4-2 is Intrepid Control Systems general-purpose interface device, affording entrance to many channels of CAN communication protocols. The ValueCAN 4-2 could be performed to monitor and transmit on CAN and CAN FD systems. For hardware simulations, it can also produce network analysis. The ValueCAN 4-2 support CAN FD; it has an extraordinary performance, a robust case and connectors.

Overview
The "ValueCAN 4-2's network Interfaces and Features are:

- Two dedicated dual-wire CAN / CAN FD channels (ISO 11898-2:2015).
- Both dual wires CAN channels have CAN FD support.
- Support for ISO CAN FD and NON-ISO CAN FD.

Figure 3.10: AEK-CON-AFLVIP2 connector board

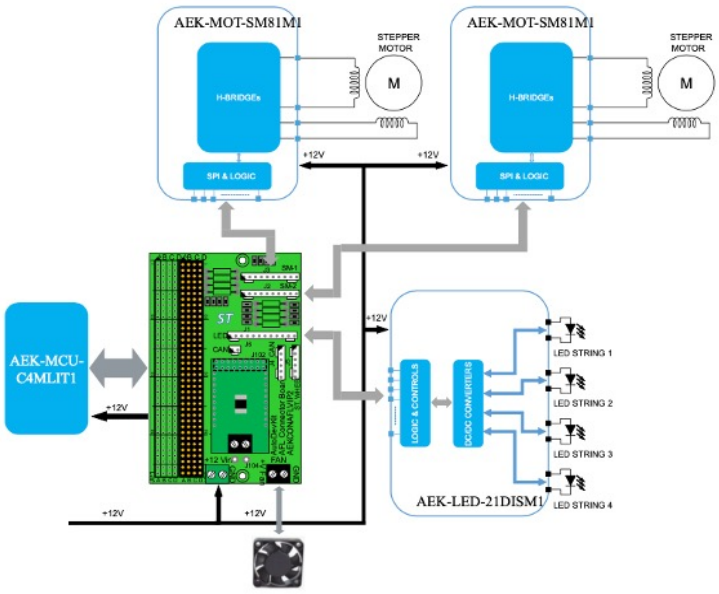

Figure 3.11: AEK-CON-AFLVIP2 block diagram

- Software-programmable CAN termination resistance.
- Real-time clock for 64-bit message timestamping."[17]

Figure 3.12: ValueCAN 4-2 top view

The ValueCAN 4-2 Is Connected within a USB Type-A to a USB port on the PC. It is feasible to use a powered USB hub to connect the ValueCAN 4-2, but performance varies due to the quality of the hub and its ability to provide power. The ValueCAN was operated by related software to simulation signals; is labelled as Vehicle Spy. Another side of the connector cable is connected through CAN inputs pins into the AEK-MCU-C4MLIT1 board.

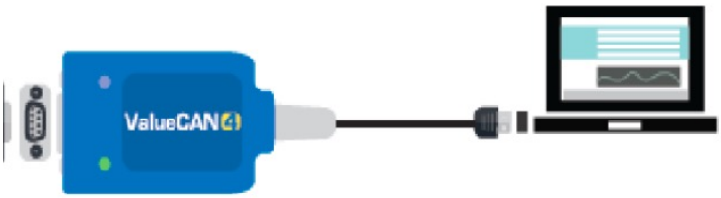

Figure 3.13: Connector diagram with ValueCAN OBD-II cable

3.10 Automotive Front Light kit

The Front Light system is assembled by with LED lights, motors and cooling fan. Can accommodate adaptive front light (AFL) adjustment systems for simulation and development purposes.

A simple adaptive front lighting system can receive feedback data from sensors indicating the position or orientation of the motor vehicle steering wheel. This feedback message can be emulated with a potentiometer signal plugged to connector J5 on the AEK-CON-AFLVIP2 connector board, where various voltages will represent different steering wheel positions.

Figure 3.14: Front Light kit

Number	Operation
1	"Perspex headlight"
2	"Turn indicator LED string"
3	"High-beam LED light"
4	"Low-beam LED light"
5	"Daytime running light LED string"
6	"Adaptive X-axis position stepper motor"
7	"Adaptive Y-Axis position stepper motor 8. cooling fan"
8	"PX2 connector"
9	"JPX3 connector"

Table 3.1: AFL components list [3]

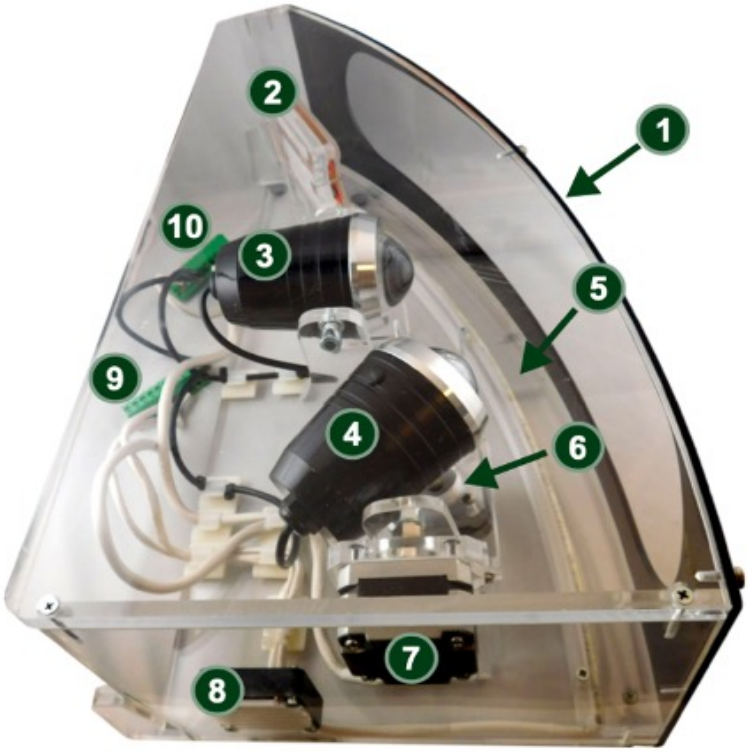

Figure 3.15: Adaptive front lighting headlight assembly

Chapter 4

Software overview

4.1 SPC5-STUDIO overview

SPC5-STUDIO is an integrated development environment (IDE) based on Eclipse. It includes a norm workspace and a plug-in system environment. “SPC5-STUDIO tries to increase developer productivity of embedded applications based on SPC5 microprocessors with a single tool for evaluation, development, prototype and production. SPC5-STUDIO includes an application wizard to clarify project creation and configuration; it automatically gives a solution component dependencies and generates support files.”[15]

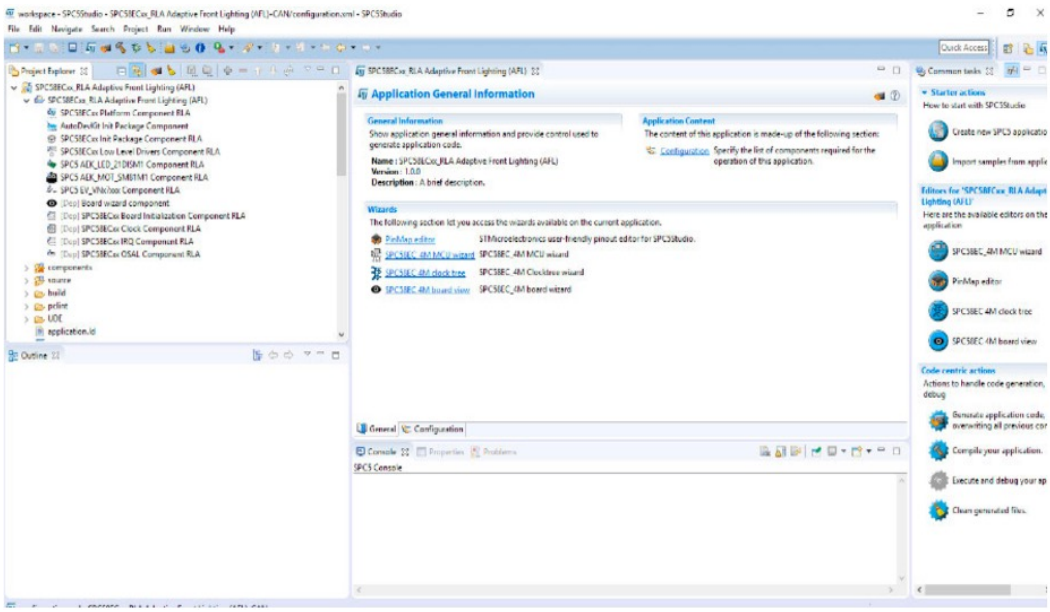

Figure 4.1: Project workspace on SPC5-STUDIO AFL

The application wizard unites the first components into the project, with the critical elements employed by SPC5-STUDIO to build the final application source code. Standard of layered architecture, the sets in one segment are provided to other components. Moreover, the configuration of each component is supported by an automatic GUI. “Register Level Access (RLA) components are low-level drivers with straightforward access to the MCU and peripherals such as CAN, Ethernet, DSPI, ADC, PIT and GTM. The RLA segments can be added and configured within a time-saving GUI.”[15]

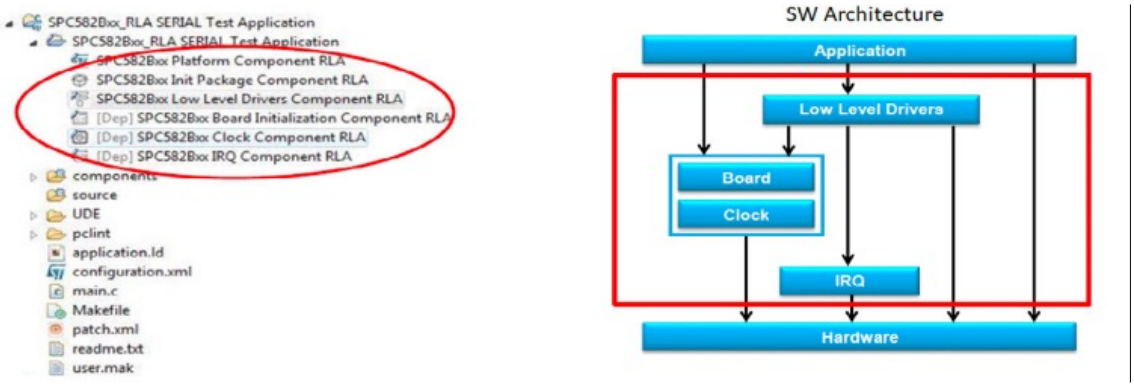

Figure 4.2: SPC5-STUDIO AFL RLA drivers

The FreeRTOS is a real-time open-source operating system, that is agreeable with the rest of the environment. SPC5-STUDIO also contains direct software examples for each peripheral in the MCU, which developers can use to become familiar with the particular code involved. Other “features of SPC5-STUDIO are:

- The ability to integrate other software products from the Eclipse standard market place
- The possibility to support industry-standard compilers
- Support of multi-core microcontrollers
- A PinMap editor to facilitate MCU pin connections”[9]

4.2 SPC5-UDESTK-SW overview

The SPC-UDE/STK (UDE) can be used as stand-alone debug environment. After the application has been loaded successfully, by the Debug menu can run or step through the procedures. Some features for programming and debugging are setting breakpoints, watch the CPU registers, Watch tips.

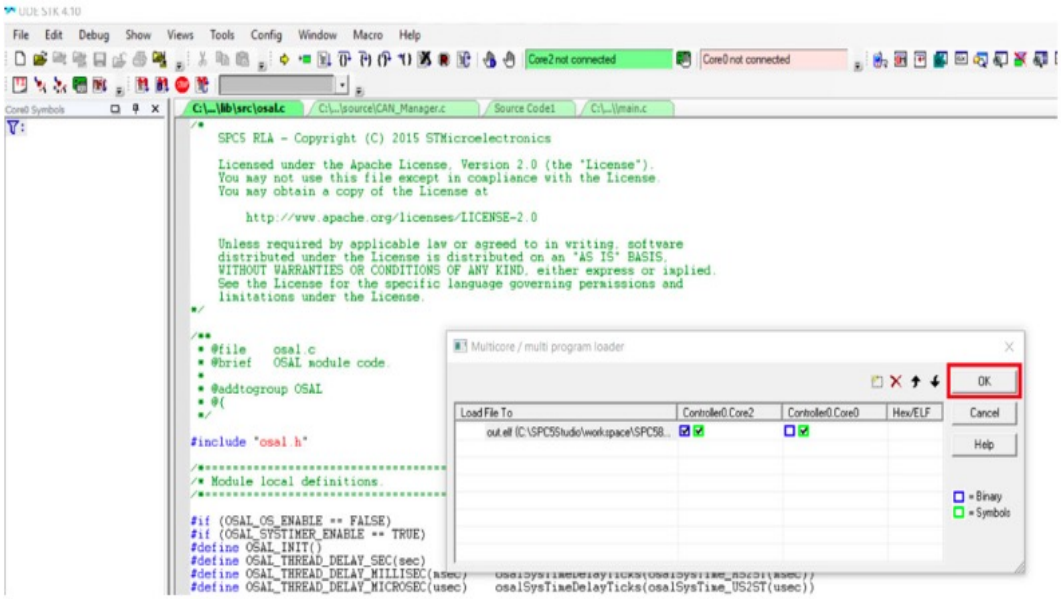

Figure 4.3: SPC-UDE/STK program loader

4.3 Vehicle Spy overview

Vehicle Spy Enterprise is a tool that lets to perform diagnostics, node/ECU Simulation, data acquisition, automated testing and vehicle network bus monitoring. Some characteristic that vehicle spy is provided are:

- Bus Monitoring
- Simulation
- Data Acquisition
- Diagnostics

In this thesis, radars, sensors, feedback dashboard lights, commands to controlling systems were defined by Vehicle Spy and transmitting and retransmitting was happed through ValueCAN 4-2.

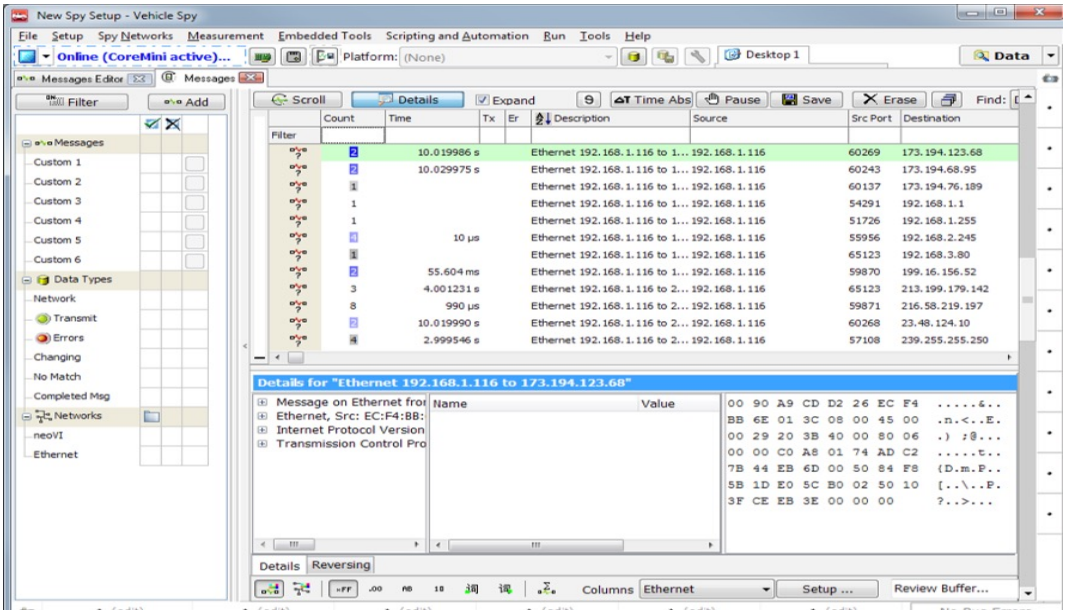

Figure 4.4: Vehicle Spy Messages window

Part II

Adaptive Front Light Systems

Chapter 5

Initial configuration

By the following procedure, as shown in the figure23 components are added to the project in the SPC5 Studio. “These components in following that are essential for the project and should add;

- SPC58Cxx Init Package Component: to initialisation and the configuration of the selected board.
- SPC58Cxx Clock Component RLA: for the configuration of the MCU clock tree
- SPC58Cxx IRQ Component RLA: to set and configure interrupt Request QUEUE
- SPC58Cxx OSAL Component RLA: operating system abstraction”[15]

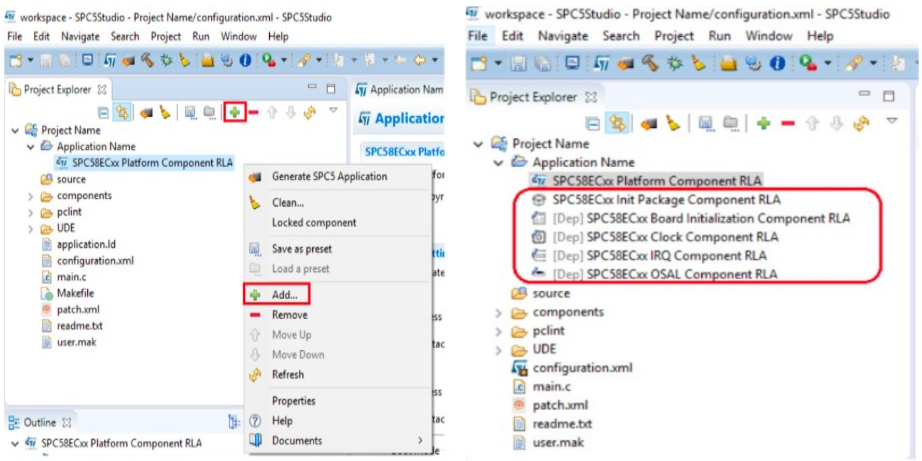

Figure 5.1: Adding component

5.1 Pin mapping

The PinMap editor supports pin functionality selection by presenting the potential configurations for any selected pin. Pins in SPC5 microcontrollers are known as Ports because they can have various functions depending on "the configuration settings in individual MCU registers:

- Peripherall pins such as CAN, ADC, eMIOS.
- System function pins such as RESET (cannot be modified).
- Special function pins such as supply voltage pins."[15]

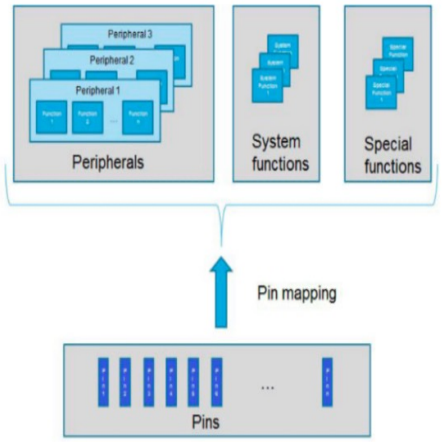

Figure 5.2: Pin mapping

The pin configuration was performed with the help of a graphical illustration of the chosen MCU. The following procedure is shown how a pin was configured. A peripheral was selected according to that available pine was chosen then type and direction of the signal were configured.

5.2 Component configuration

Figure 5.3 Pin configuration Configuration of each component and peripheral was done through the graphical interface of the SPC5 studio.

5.3 Set low-level driver IRQ

The AFL system has several interrupts of various kinds, which can be connected with events linked to peripherals such as DSPI, LIN, PIT, or to other components.

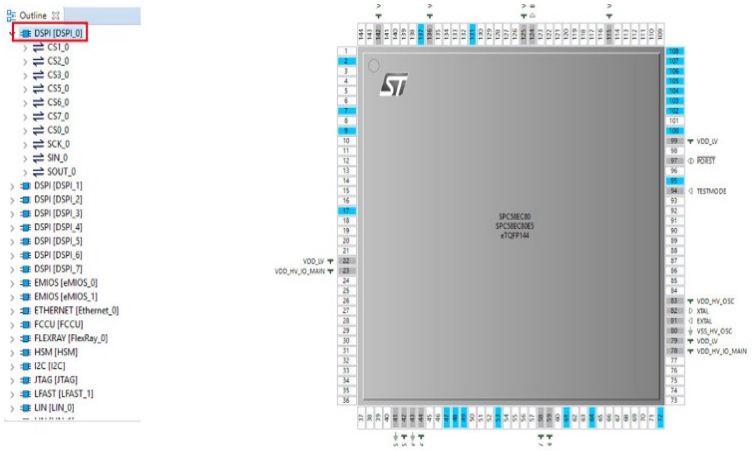

Figure 5.3: Pin configuration; Selecting a peripheral

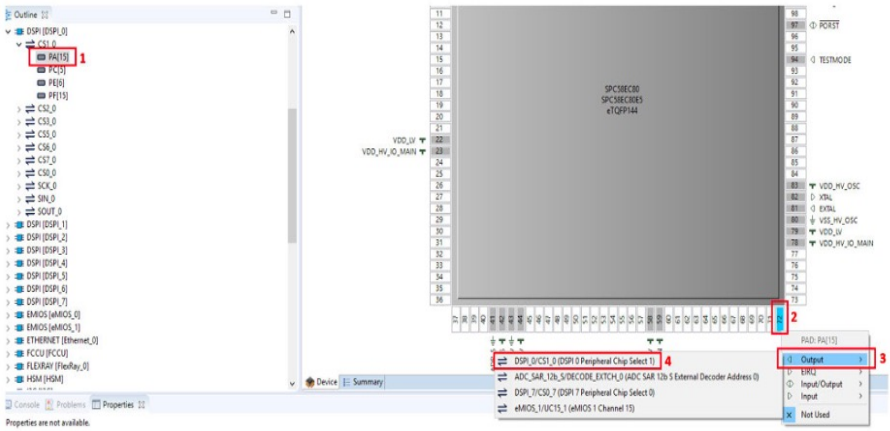

Figure 5.4: Pin configuration

These interrupts should be reprioritized order to avoid possible conflicts. After configuration, the source code was generated.

5.4 DSPI configuration

Basic configuration was done by the setting of SPI mode and enabling TX FIFO.

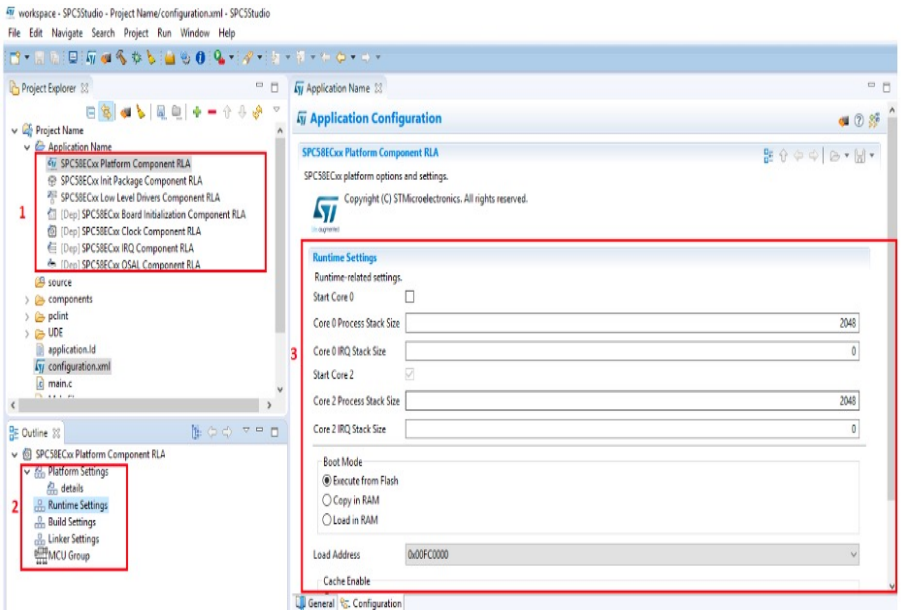

Figure 5.5: Component configuration

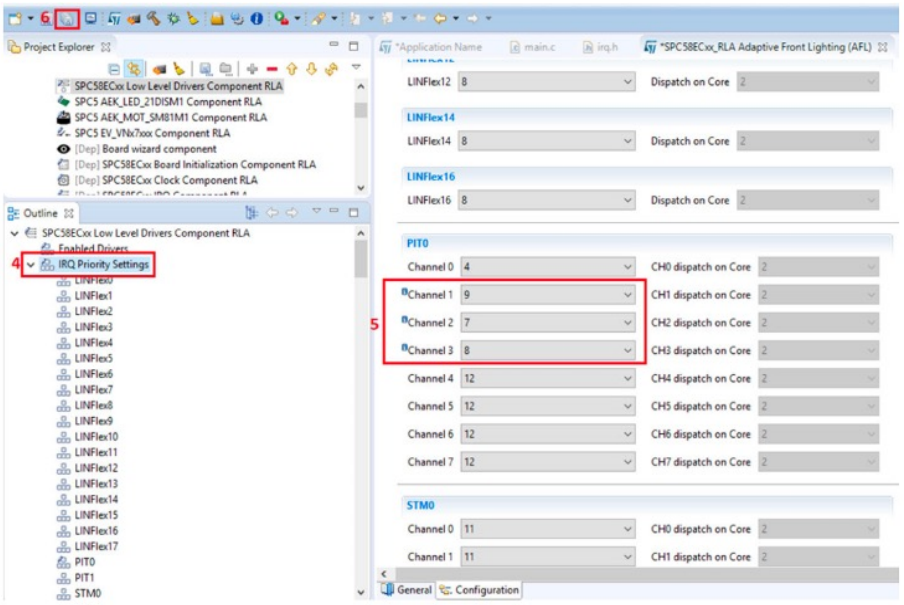

Figure 5.6: Low-level IRQ driver configuration

Listing 5.1: DSPI configuration

```
1 DSPI .MCR.B.MDIS = 0x0; // Enable DSPI clocks
2 DSPI.MCR.B.HALT = 1; // stop DSPI transmission
```

```
3   DSPI.MCR.B.DCONF = 0;   // DSPI mode
4   DSPI.MCR.B.MSTR = 0x1; // SPC is master
5   DSPI.MCR.B.DIS_TXF = 0x0; // enable TX FIFO
6   DSPI.MCR.B.DIS_RXF = 0x0; // enable RX FIFO
7   DSPI.MCR.B.CLR_TXF = 0x1; // clear TX FIFO
8   DSPI.MCR.B.CLR_RXF = 0x1; // clear RX FIFO
9   DSPI.MCR.B.PCSIS0 = 0x0;   // PCS[0] is active low ~ I2S_WS
      signal
```

Clock configuration was done by continues clock in the following way, The setting of clock required frequency a sample rate of 44.1 kHz.

Listing 5.2: DSPI configuration

```
1  DSPI  .MCR.B.CONT_SCKE = 0x1; // continuous clock
2  SCK DSPI.CTAR0.B.LSBFE = 0x0; // MSB(LSB) transmitted first
3  DSPI.CTAR0.B.FMSZ = 15; // command frame size 16 bits
4  //note: two SPI frames has to be sent for each I2S channel
5  DSPI.CTAR0.B.CPOL = 0x1; // normal SCK polarity
```

5.5 DSPI command and data arrangement

Commands and data are written to PUSH register as the first step for DSPI data transfer. Both 32 bits channels will be transmitted in four writes to PUSH register. The connected chip select has to be permitted to stop the PCS break during 32 bits communicate.

Listing 5.3: DSPI command and data arrangement

```
1    //SPI command COUNT = 1, CTAS = 0, EOQ = 0, no PCS (I2S_WS=0)
2     DSPI_PUSHR = 0x 0x8000dddd; //Frame0      16bit
3     DSPI_PUSHR = 0x 0x8000dddd; //Frame1      16bit
4
5     //SPI command COUNT = 1, CTAS = 0, EOQ = 0, PCS0 selected (I2S_WS
      =1)
6       DSPI_PUSHR = 0x 0x8001dddd; //Frame2      16bit
7       DSPI_PUSHR = 0x 0x8001dddd; //Frame3      16 bit
```

5.6 DMA configuration

Before writing to PUSH register, the data and request were merged. It was done off-line in RAM or FLASH. In this case, we were needed four transfers to send one sample of both channels. The DMA was also prepared for new data for inter-buffer. In this case, eight transfers were needed to send one sample of both channels. For transfers, several configurations were possible. An example of such a DMA

configuration is here. Block of data 0 to data n has to be frequently transferred to DSPI by a command for channel L and command for channel R. For every channel the 20 bits data are split into two 16 bits words. The 4x32 bits inter-buffer is used to consolidate data and commands. In this case, the DMA 12 transfer is triggered by DSPI when TX FIFO is not full. After the command with data3 is transferred to PUSH register the DMA 13 is linked to transfer new 4 data to inter-buffer. Both buffers are circular, so there is no need for CPU intervention.

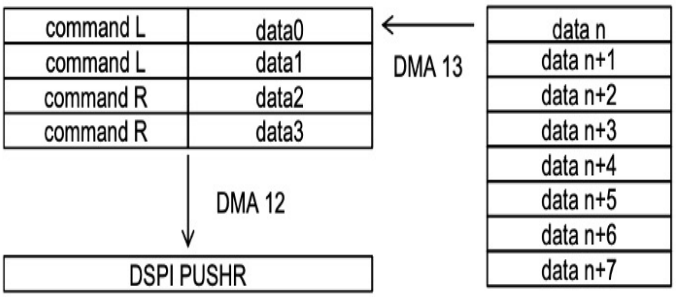

Figure 5.7: Diagram of DMA configuration

5.7 CAN configuration

The SPC58 hosts 8 CAN controllers. These controllers are designed into two subsystems to optimize device. All the embedded CAN controllers present in the same subsystem will share resources like RAM, memory and clock.

Overview
The CAN subsystem consists of the following main blocks:

- "Modular CAN cores: the registers of the CAN module can be reached using the generic slave interface (GSI). The peripheral GSI module permits act as a request from the various master. 29
- CAN-RAM arbiter: is an additional logic for arbitration between the requests for RAM access from the various CAN controllers.
- SRAM: the CAN subsystem will interface with an external RAM applying this interface.
- ECC Controller: it contains the logic to estimate and validate the correction code on the SRAM memory cells."[17]

The message section is intended to be outside of the module a single-ported Message RAM. Depending on the collected device, various M-CAN controllers can share the same Message RAM. The handling of messages refers to each function concerning is implemented by the RX Handler and the TX Handler. The RX Handler controls message taking filtering, the transfer of received messages from the CAN to the Message RAM, and it also receives message status data. The TX Handler is ready for the transfer of transmitting messages from the Message RAM to the CAN core, and it also transmits status data. After the ECC functionality implements the Message RAM, was initialized Message RAM after hardware reset by writing 0x00. This avoids that reading of uninitialized RAM segments will activated interrupt IR.BEC (Bit Error Corrected) or IR.BEU (Bit Error Uncorrected).

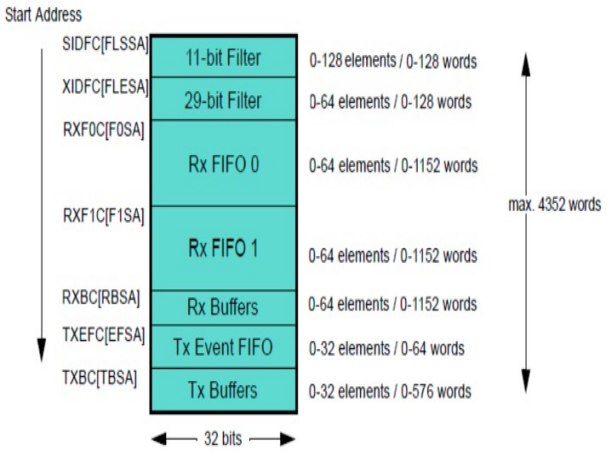

Figure 5.8: Message RAM configuration

5.8 PIN configuration

It is possible to combine one at the time a list of functions to the PIN within the inner configuration of the MCU registers, therefore, the port is a logical absorption of the pin in a sense. A particular plug-in named the Pin Map Editor in the SPC5-Studio: this is an intuitive graphical interface implemented to the microcontroller's pin arrangement. The Pin Map Editor controlled the configuration procedure step by step. Additionally, it assists by damaging the options not compatible with the picked driver.
In following is a summary of the pin configuration that was done:
1. Was Selected the driver, CAN1;
2. Selected the signal on the driver. in this project RX and TX for the CAN;
3. For any signal, was selected the pin to configure.

4. Was selected the signal direction: input, output or bidirectional;

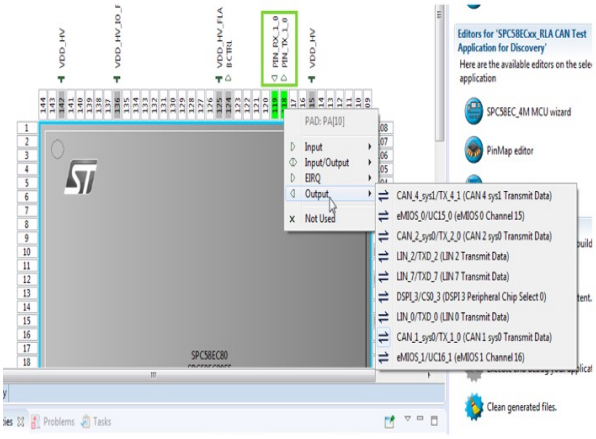

Figure 5.9: Pin configuration selection

5.9 Low-level driver configuration

The low-level driver is relevant to the configuration and choice of the elements to add. The chosen element, in this statement, is the CAN controller. The CAN subsystem consists of the modular CAN (M-CAN) modules beside an integrated smart CAN RAM controller. The CAN RAM controller consists of extra logic for adjustment among the requests for the RAM access by the different CAN controllers. Shared memory is located between the four M-CAN controllers in each subsystem.

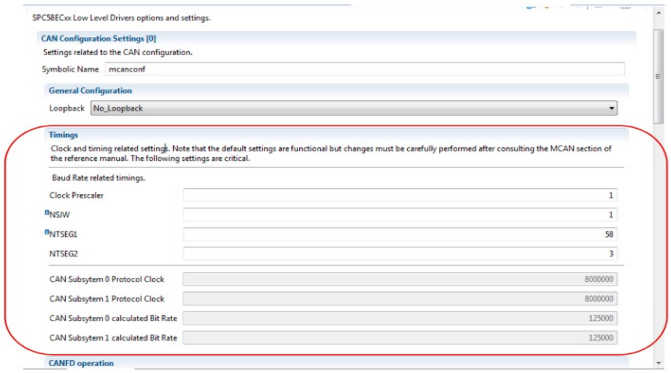

Figure 5.10: Timing configuration

Chapter 6

Curve-adaptive headlights

6.1 Steering Wheel sensor

The car string wheel is rotated 180-degree as shown in the figure 2 regard to the position of that sensor is set, the variation of the voltage to the AEK-MCU-C4MLIT1 by AEK-CON-AFLVIP2 J4 (ST.W) and J5 (CAN) connectors. The sensor here is a 10kohm 344 potentiometer. It is not a great choice but is acceptable for simulation kind of these projects.

Features

- "Standard Resistance Range: Linear 1 K ohms to 1 megohm
- Total Resistance Tolerance: Linear Tapers ±20 (percentage)
- Independent Linearity: ±5 (percentage)
- End Resistance: 2 (percentage) or 2 ohms maximum (whichever is greater)
- Effective Electrical Angle 270 ° ±15 ° (In the AFL project is locked to 180°)
- Contact Resistance Variation: 1 (percentage) or 1 ohm (whichever is greater)
- Operating Temperature Range: -40 °C to +125 °C (-40 °F to +257 °F)" [9]

The SAR ADC convertor is converted an input analogue signal to the digital respect to the precision of the resistance potentiometer seven different positions of the steering wheel is hired for a different routine. The steering wheel can be rotated 90 degrees from starting point, for instance, from starting from position

Figure 6.1: AEK-CON-AFLVIP2 J4 (ST.W)

1 to 4. The table below is described a distinct range of the digital output value that is converted from analogue value by SAR Analog to Digital converter, the first position of the steering wheel has an output value between [0 - 228]. On each rotation position from left to right or vice versa, the motor driver regarding variation can adjust the position of Low beam light in the X-axis, and this option is available also when the vehicle is in the stopping condition to provide stabler condition even coincidentally wants to see a blind spot at night.

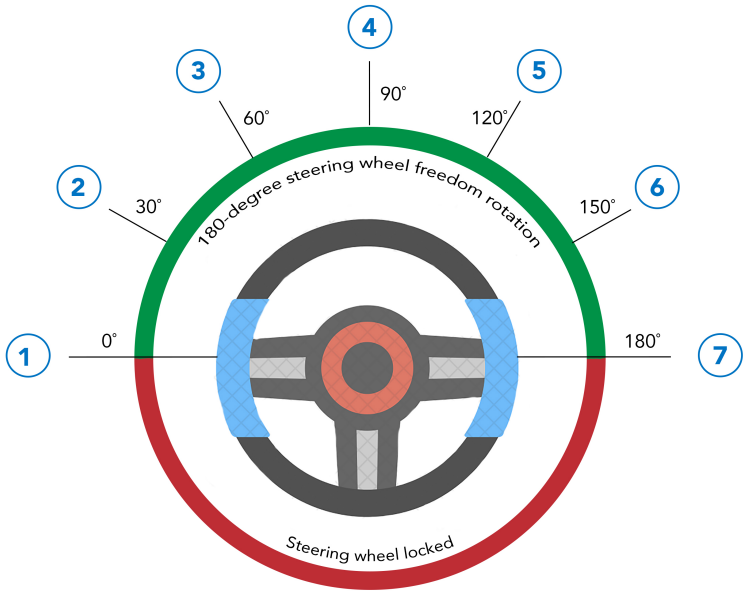

Figure 6.2: Steering Wheel different possible position

A feature that known as car comfort option is auto blinking systems that will be very useful when driver forgot to turn on blinker light when the steering wheel is rotated on the left (in our case) In each position from 3 to 1, the blinker will

be activated, and by turning back to direct position, it will be deactivated, and of course, driver able turn On/ Off the blinker in any desired time that is explained in CAN signals section of this report.

Position	Operation	SAR ADC Value
1	"Blinker blinking"	"0 – 228"
2	"Blinker blinking"	"250 – 700"
3	"Blinker blinking"	"800 – 1800"
4	–	"2000 – 2500"
5	–	"2700 – 3700"
6	–	"3800 – 4000"
7	–	"4095 – 40228"

Table 6.1: Positions and operations [3]

A low beam LED position is calibrated in the centre position (direct light) by the function that is dedicated to doing that.

Listing 6.1: Positions and operations

```
1
2  if (prev_positioning != positioning )
3              {
4                   if (step == 0)
5                       step = 6 - positioning;
6                   else
7                       step = prev_positioning - positioning;
8                   prev_positioning = positioning;
9      //Turn left, switch on blinker
10                  if (positioning >= 1 && positioning <= 5 )
11                  {
12                      pit_lld_channel_start(&PITD1, PIT0_CHANNEL_CH2);
13
14                  }
15
16
```

6.2 Stepper motors

Two stepper motors, one for up-down and one for lateral angular displacement of light, The fundamental components of an adaptive front lighting system are the headlight equipment with LED lights and motors directional positioning, and the driver logic and the control to operate the system. The ST AutoDevKit leads these components in two highly convenient kits. The principal is the AEKD-AFLPANEL1 kit with all the essential control and function boards and preloaded firmware, and the second is the AEKD-AFLLIGHT1 headlight assembly, which is connected to the board within two simple connectors.

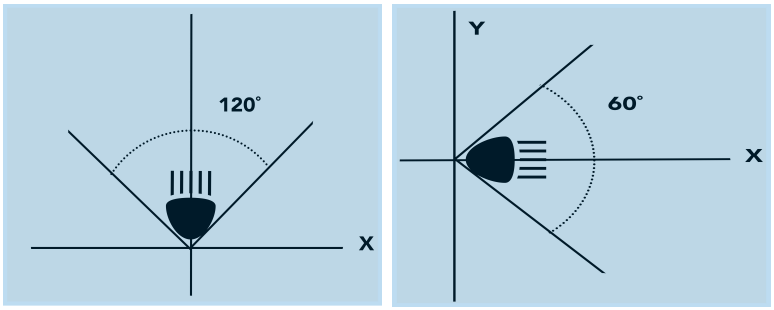

Figure 6.3: Stepper Motors freedom degree

The algorithm provides a freedom situation for stepper motors to rotate 120 and 60 degrees in the X and Y axes, respectively. As shown in the table below, step motors can be rotated in 1.80 deg by ± 5 accuracies.

Voltage	**3.50**
AMPS/PHASE	1
Step Angle	1.80 deg
Step Accuracy (Non-Accum)	± 5

Table 6.2: Stepper motors characteristics

The SAR converter converts the output of the potentiometer by reliable precision for each rotation of ST. Wheel an angle is dedicated to a step phase number that

is presented input voltage, which is converted to a digital value.

STEP	A	B	A\	B\		CCW
1	+	+	-	-	↓	↑
2	-	+	+	-		
3	-	-	+	+		
4	+	-	-	+	CW	

Figure 6.4: Full step

Through the SPI communication protocol commands to set the position of step motors are written to the register file to setting AEK-MOT-SM81M1drv motor driver also the resolution of the driver is set as well.

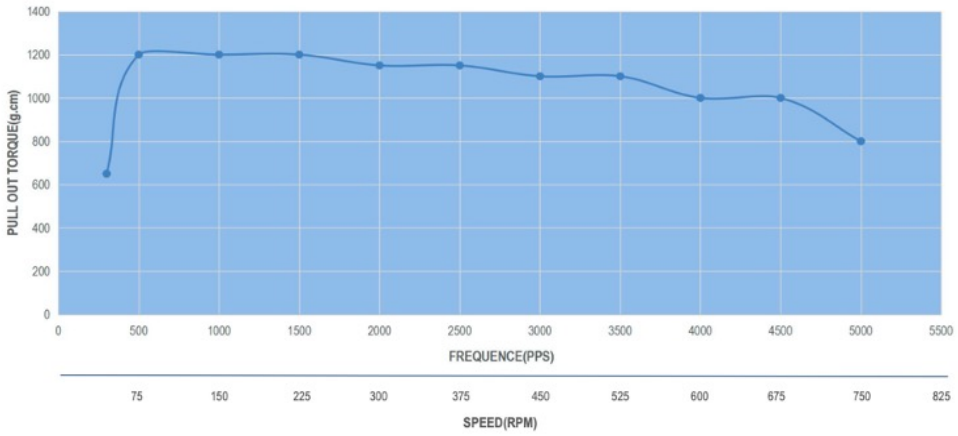

Figure 6.5: Torque curve

6.3 Read the register through the SPI peripheral

- Param driver: SPI driver pointer
- Param config : SPI config pointer
- Param regAddress: Address of the register
- Param DataReceived: Data received pointer

Listing 6.2: Read the register through the SPI peripheral

```
void AEK_MOT_SM81M1drv_ReadSPIRegister(SPIDriver* driver, SPIConfig*
    config, unsigned char regAddress,
            char* DataReceived) {
        unsigned char parity = 0;
        Spi_DataType rawData[3];
        Spi_DataType SPI_cmd[1];
        Spi_DataType SPI_payload[1];
        uint32 tmpData_A;
        uint32 tmpData;
        uint8_t rxbuf[3];
        unsigned long int temp = 0;

        tmpData = updateParityBitMotor((uint32) (regAddress |
    ST_SPI_RD) << 16);
        intToArrayMotor(tmpData, rawData);

        spi_lld_start(driver, config);
        spi_lld_exchange(driver, 3, rawData, rxbuf);
        spi_lld_stop(driver);

        tmpData_A = MCR1_START;
        tmpData = updateParityBitMotor(
                (uint32) (((regAddress & ST_SPI_ADDR_MASK) |
    ST_SPI_WR) << 16)
                        | (tmpData_A & ST_SPI_DATA_MASK));
        intToArrayMotor(tmpData, rawData);
        spi_lld_start(driver, config);
        spi_lld_exchange(driver, 3, rawData, rxbuf);
        spi_lld_stop(driver);

        tmpData_A = MCREF_CURR;
        tmpData = updateParityBitMotor(
                (uint32) (((0x08 & ST_SPI_ADDR_MASK) | ST_SPI_WR) <<
    16)
                        | (tmpData_A & ST_SPI_DATA_MASK));
        intToArrayMotor(tmpData, rawData);
        spi_lld_start(driver, config);
        spi_lld_exchange(driver, 3, rawData, rxbuf);
        spi_lld_stop(driver);

    }


```

6.4 Write the register through the SPI peripheral

- Param driver SPI driver pointer
- Param config SPI config pointer
- Param regAddress Address of the register
- Param DataReceived Data received pointer
- Param mode should be written in the register

Listing 6.3: Write the register through the SPI peripheral

```
1   void AEK_MOT_SM81M1drv_WriteSPIRegister(SPIDriver* driver,
    SPIConfig* config, unsigned char regAddress,
2           char* DataReceived, int mode) {
3       unsigned char parity = 0;
4       Spi_DataType rawData[3];
5       Spi_DataType SPI_cmd[1];
6       Spi_DataType SPI_payload[1];
7       uint32 tmpData_A;
8       uint32 tmpData;
9       uint8_t rxbuf[3];
10      unsigned long int temp = 0;
11
12      if (mode == 1)
13          tmpData_A = MCR1_START;
14      if (mode == 2)
15          tmpData_A = 0x00;
16      if (mode == 3)
17          tmpData_A = MCR2_FREG;
18      if (mode == 4)
19          tmpData_A = MCREF_CURR;
20      if (mode == 5)
21          tmpData_A = MCR1_POS3;
22      if (mode == 6)
23          tmpData_A = MCR1_POS4;
24
25      tmpData = updateParityBitMotor(
26              (uint32) (((regAddress & ST_SPI_ADDR_MASK) |
    ST_SPI_WR) << 16)
27                      | (tmpData_A & ST_SPI_DATA_MASK));
28      intToArrayMotor(tmpData, rawData);
29      spi_lld_start(driver, config);
```

```
30          spi_lld_exchange(driver, 3, rawData, rxbuf);
31          spi_lld_stop(driver);
32
33      }
34
```

6.5 Write the register to set the step resolution

- Param driver SPI driver pointer
- Param config SPI config pointer
- Param regAddress Address of the register
- Param DataReceived Data received pointer
- param mode Step resolution
- param pos Step positioning
- param dir Step direction

Listing 6.4: Write the register to set the step resolution

```
1  void AEK_MOT_SM81M1drv_WriteSPIRegisterMOV(SPIDriver* driver,
       SPIConfig* config,
2              unsigned char regAddress, char* DataReceived, STEP_MODE
       mode,
3              long int pos, int dir) {
4          char parity[2];
5          Spi_DataType rawData[3];
6          Spi_DataType SPI_cmd[1];
7          Spi_DataType SPI_payload[1];
8          uint24 tmpData_A = 0x00;
9          uint32 tmpData;
10         uint8_t rxbuf[3];
11         int temp = decimalToBinary(pos);
12
13         //write register
14         tmpData_A |= MCR1_ME;
15         switch (mode) {
16         case HALF:
17             tmpData_A |= MCR1_HALF;
18             break;
```

```
19          case FULL:
20              tmpData_A |= MCR1_FULL;
21              break;
22          case MINI:
23              tmpData_A |= MCR1_MINI;
24              break;
25          case MICRO2:
26              tmpData_A |= MCR1_MICRO2;
27              break;
28          case MICRO3:
29              tmpData_A |= MCR1_MICRO3;
30              break;
31          case MICRO4:
32              tmpData_A |= MCR1_MICRO4;
33              break;
34          case MICRO8:
35              tmpData_A |= MCR1_MICRO8;
36              break;
37          }
38          if (dir == 1)
39              tmpData_A |= MCR1_DIR;
40          //Position
41          tmpData_A |= pos << 1;
42
43          tmpData = updateParityBitMotor(
44                  (uint32) (((regAddress & ST_SPI_ADDR_MASK) |
    ST_SPI_WR) << 16)
45                          | (tmpData_A & ST_SPI_DATA_MASK));
46          intToArrayMotor(tmpData, rawData);
47          spi_lld_start(driver, config);
48          spi_lld_exchange(driver, 3, rawData, rxbuf);
49          spi_lld_stop(driver);
50
```

6.6 Turn the Stepper Motor in direction and angle defined

- param dev Device selected
- param grade Angle of rotation
- param direction Direction of the rotation
- param time Delay between commands

Listing 6.5: Turn the Stepper Motor in direction and angle defined

```
1       void    RotationGrade(AEK_MOT_SM81M1_DEVICE dev,
2               int grade, TURN_DIRECTION direction, int  time)
3   {
4       int step;
5       step = grade / 7;
6       if (direction == RIGHT) {
7           TurnRight(dev, step, time);
8       }
9       if (direction == LEFT) {
10          TurnLeft(dev, step, time);
11      }
12  }
13
```

Chapter 7

Communicate Protocol Operation

In the adaptive front light systems, some operation is defined as simulated by CAN communication protocol:

- Welcome light
- Doors open /close warning light
- Turn on /off Low Beam light
- Turn on /off High Beam light
- Turn on /off Daylight
- Turn off all active light at once
- Driving situation
- Traffic jam situation during the night
- The average driving situation during the night (low traffic)
- The parking brake (stopping situation)
- Follow Me Home

7.1 Graphical Interface

Signals for these peculiarities are listed in above are defined as DBC by VehicleSpy and are sent by ValueCAN 4. Signals are divided into two kinds of assortments, Transmitting signals to commanding or to sending a value that sensed by sensors or radars; on the other hands, some signals are applied to receiving signals to explain the current situation on the car dashboard, for transmitting and receiving are dedicated 19 and 5 signals respectively.

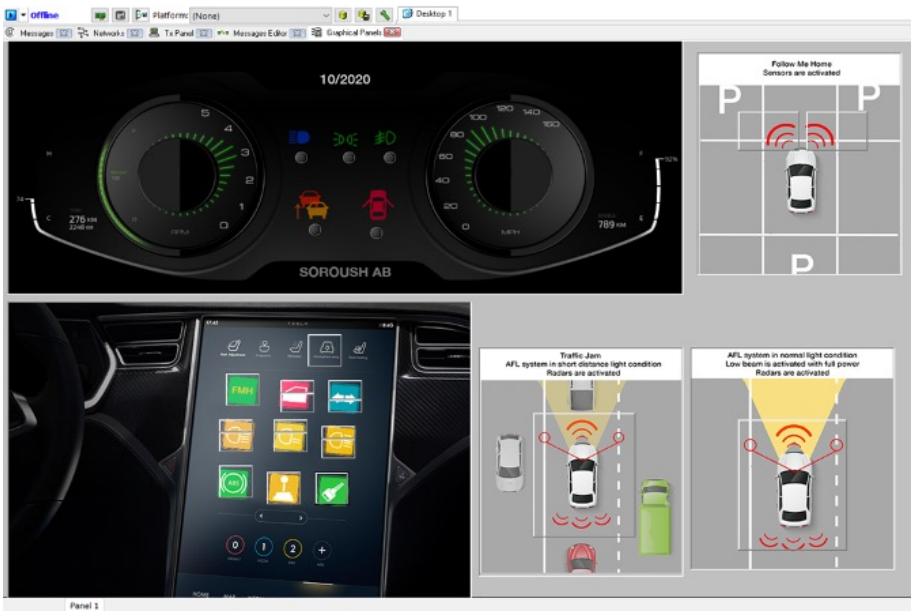

Figure 7.1: Vehicle SPY graphical monitoring panel

Thanks to the graphic panels of the Vehicle Spy, these signals are designed to demonstrate alteration conditions and situations more straightforwardly by the user-friendly panel that is designed to commanding and controlling alterations. In figure above is shown an overview of all graphical control panels designed in the Vehicle Spy.

7.2 Dashboard for receiving signal

The Dashboard graphical control panel is designed to demonstrate signals in the Vehicle Spy, which are received by ValueCAN 4 from AEK-MCU- C4MLIT1. When messages appear, the dedicated symbol of that will be turn on till that signal is available.

Figure 7.2: Dashboard

Number	Operation
1	RPM dashboard gauge
2	High Beam dashboard light
3	Blinker dashboard light
4	Low Beam dashboard light
5	Car distance dashboard light
6	Doors warning dashboard light
7	Speed dashboard gauge

Table 7.1: Dashboard symbol light

7.3 Touch screen control panel for transmitting signals

The control panel designed is used to command and convey signals in a user-friendly way by clicking on each icon, as described in the table, which represents related signals.

Figure 7.3: Control panel

7.4 Compatible light intensity (Short Distance Light situation)

The short distance light is designed to present a comfortable condition to driving during the night at the traffic jam, and nowadays cars light have become brighter and brighter From halogen lights to Xenon systems and LEDs technically speaking for safety there is immeasurable progress, on the other hand, traffic jam for the people in the big city has become a routine regarding the size, high, and variety of kind of vehicle intensity direct light during driving how could be annoying, adaptive front light systems is designed to be capable of changing the power of light in the different situation such as the normal situation in the highway and city roads or at traffic jams, by receiving data from two radars the algorithm can adapt itself. In the figures below, two different signals of radars are defined as simulation. By clicking on any of them, the finite state machines are programmed to handle a new subroutine and adjust Lowbeam to that condition more details of how the systems work is described in the next chapter.

Number	Operation
1	The parking brake (Stopping situation)
2	Driving situation
3	Welcome light
4	High Beam light On /Off
5	Daylight On /Off
6	LowBeam light On /Off
7	Flow me home
8	Doors Open /Close
9	Blinker
10	Turn Off all running light

Table 7.2: Control Panel Icon

7.5 Follow-Me Home

The follow-me headlamps hold your headlights to stay on a few moments longer after you Turn off the engine and close the doors to give you several seconds of lighting from the lamps to illuminate to you while you make your way to a door or your destination. This algorithm provides an alternative solution to predict and recognize direction of the way that drivers desire to go and then to light that side. Thanks to the number of the sensors that are sounded, the car AFL system can sense the direction of drivers and passengers are passing. To simulation, this feature within the Vehicle Spy should do these particular steps:

- Send a signal as stopping situation (brake signal), because this feature during the driving is not feasible and does not work.
- Send an FMH signal request
- and finally by clicking on each side of the car in the figure below (red symbol) will be able to see the response from the AFL system.

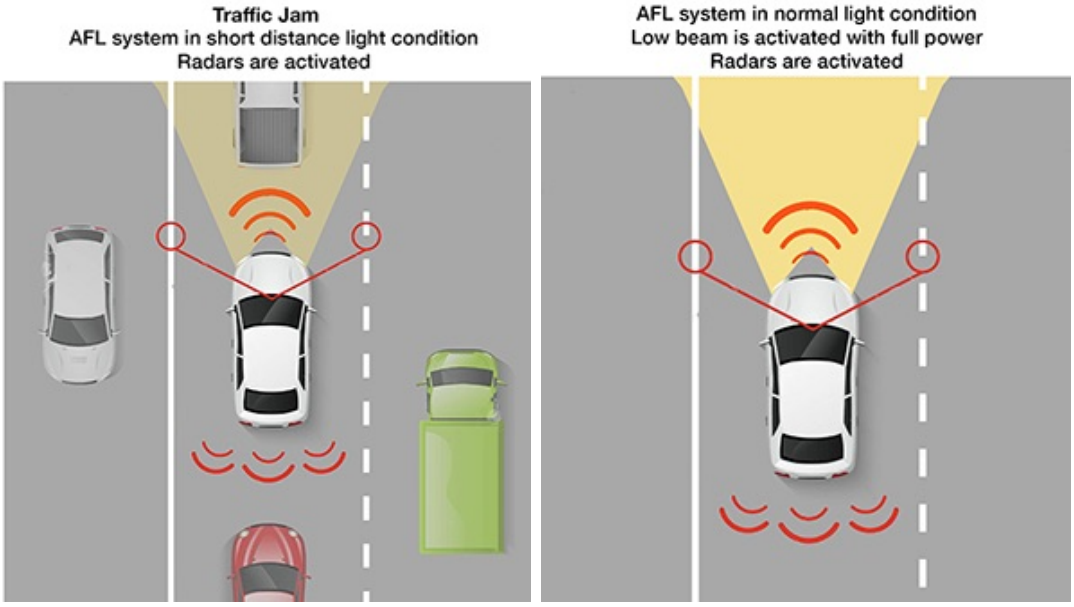

Figure 7.4: Traffic jam simulation

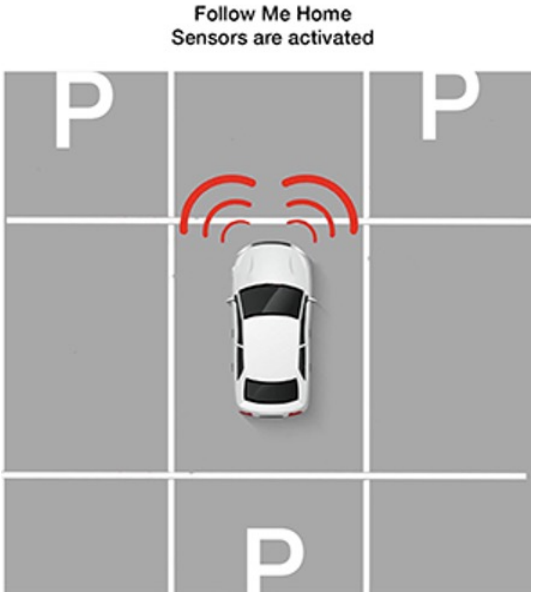

Figure 7.5: Traffic jam simulation

Chapter 8

CAN Message Characteristics

A controller area network (CAN) is fitting for the numerous high-level modern protocols embracing CAN and ISO 11898 as their physical layer. Its cost, performance, and upgradeability provide for considerable flexibility in system design. There are two types of messages frame forms; standard and extended. The only difference between the two is the length of the identifier. Standard is 11 bits, and extended is 29 bits in length. An IDE bit in the frame checks this selection, dominant for standard and recessive for extended. CAN has four frame types

Data Frame

This is a frame holding data for transmission.

Standard Frame

A standard frame structure looks like the following, with field specifications in the table below.

SOF	Identifier	RTR	IDE	R	DLC	DATA	CRC	ACK	EOF	IFS

Each frame and bits are defined for Transmitting signal and receiving a signal by particular arbiter ID; The TX Buffers is dedicated for message transmission. Each dedicated TX Buffer is configured with a particular Message-ID also. The FIFO Buffers are controlled and organized in the logic that the first one arrives in is the first one that comes out.

Field name	Length (bits)	Description
Start of Frame (SOF)	1	Denotes the start of frame transmission.
Identifier	11	An identifier for the data which also represents the message priority.
Remote Transmission Request (RTR)	1	Dominant. See Remote Frame subsection.
Identifier Extension (IDE)	1	Sets standard (0) or extended format (1).
Reserved bit (r0)	1	Reserved bit.
Data Length Code (DLC)	4	Number of bytes of data (0–8 bytes).
Data	0–64 (0-8 bytes)	Data to be transmitted.
CRC + delimiter	16	Cyclic Redundancy Check.
ACK slot + delimiter	2	Transmitter sends recessive and receiver asserts dominant on successful reception.
End Of Frame (EOF)	7	Marks the end of a CAN frame.
Interframe Space (IFS)	7	Contains the time required by the controller to move a correctly received frame to its proper position in a message buffer area.

Figure 8.1: Standard frame

8.1 Representation CAN driver

By arrangement, Can call back and interrupt service routine transmission and receiver are available to send or receive data from external nodes, in this case, Vehicle Spy into the AEK-MCU- C4MLIT1 board.

Listing 8.1: Representation CAN driver

```
struct CANDriver {
      /**
       Current configuration data.
       */
      const CANConfig *config;
      /**
       Pointer to the CAN registers.
       */
      volatile struct spc5_mcan *mcan;
      /**
       Shared RAM information.
       */
      uint32_t shared_ram_start_address;
      /**
       Max data size for RX, FIFO and TX buffer
       */
      uint8_t max_data_size;
        };
```

```
19
```

8.2 CAN transmission frame

The following structure configures feedback signals that represent in the previous chapter to demonstrate an Activation or Deactivation commands on the dashboard of the car, and these signals will be recognizable this configuration in the Vehicle Spy. All signals are defined in following with an exact length of the frame to transmitting from AEK-MCU-C4MLIT1 to the external nodes, and this signal is called when an event happened.

Listing 8.2: CAN transmission frame

```
1       typedef    struct {
2       uint8_t  OPERATION;        //operation type(NORMAL or CANFD)
3       uint8_t  RTR;             //Frame type
4       union {
5         uint8_t TYPE;         //Id type. STD or XTD
6         uint8_t IDE;          //Id type. STD or XTD
7       };
8       union {
9         uint32_t SID;        //Standard identifier
10        uint32_t EID;         //Extended identifier
11        uint32_t ID;          //identifier
12      };
13      union{
14        uint8_t  DLC;                 //Data Length
15        uint8_t  LENGTH;              //Data length
16      };
17      union {
18        uint8_t  data8[SPC5_CAN_MAX_DATA_LENGHT];   //Frame data.
19        uint16_t data16[SPC5_CAN_MAX_DATA_LENGHT/2U]; //Frame data.
20        uint32_t data32[SPC5_CAN_MAX_DATA_LENGHT/4U]; //Frame data.
21      };
22    } CANTxFrame;
```

All signals are defined in following with an exact length of the frame to transmitting from AEK to the external nodes, and this signal is called when an event happened.

High Beam dashboard light: The High Beam Indicator becomes glowed on your car's dashboard, which, as the name refers, indicates that your car's high beam headlights are on. In most of the vehicles, it is a blue symbol.
Blinker dashboard light: Any flashing light can be called a blinker, but it usually refers to the change signal on a car. Ere you quickly turn left/ right, will

Signals	Operation	HEX
1	High Beam dashboard light	0x02
2	Blinker dashboard light	0x04
3	Low Beam dashboard light	0x01
4	Car distance dashboard light	0xA
5	Doors warning dashboard light	0x08

Table 8.1: CAN transmission frame

be on. A light that blinks on / off is a blinker, and it may be used to signal your plans to change direction in a car or to send some kind of signal such as doors are open.
Low Beam dashboard light: The Low Beam Indicator becomes glowed on your car's dashboard, which, as the name refers, indicates that your car's Low beam headlights are on. In most of the vehicles is a green symbol.
Car distance dashboard light: Has a light symbol to demonstrate the short distance between vehicle, the roads will be scanned by radars when the gap between your car and next car is smaller than standard this light will be turned On, In any different situation AFL system behaved regarding that.
Doors warning dashboard light: When doors of cars have opened, this symbol glows, and flasher light start to blinking also, the symbol of Doors remained On even the driver turn it Off the blinker.

Listing 8.3: CAN packing messages

```
1    void    setCanMessage(char *nameCommand)
2    {
3        if ((strcmp(nameCommand, ACTIVATION_LOW_BEAM) == 0))
4        {
5            SET_LIGHT_ON(can_mess.data32[0],MASK_LOW_BEAM);
6            SET_LIGHT_ON(can_mess.data32[1],MASK_LOW_BEAM);
7        }
8        else if (strcmp(nameCommand, ACTIVATION_HIGH_BEAM) == 0)
9        {
10           SET_LIGHT_ON(can_mess.data32[0],MASK_HIGH_BEAM);
11           SET_LIGHT_ON(can_mess.data32[1],MASK_HIGH_BEAM);
12       }
```

```
13
14          .
15          .
16          .
17
18          else if (strcmp(nameCommand, DEACTIVATION_DOORSLIGHT) == 0)
19          {
20
21              SET_LIGHT_OFF(can_mess.data32[0], MASK_DOORS);
22              SET_LIGHT_OFF(can_mess.data32[1], MASK_DOORS);
23
24          }
25          else if (strcmp(nameCommand, ACTIVATION_DISTANCELIGHT) == 0)
26          {
27              SET_LIGHT_ON(can_mess.data32[0], MASK_DISTANCE_LIGHT);
28              SET_LIGHT_ON(can_mess.data32[1], MASK_DISTANCE_LIGHT);
29          }
30          else if (strcmp(nameCommand, DEACTIVATION_DISTANCELIGHT) ==
    0)
31          {
32
33              SET_LIGHT_OFF(can_mess.data32[0], MASK_DISTANCE_LIGHT);
34              SET_LIGHT_OFF(can_mess.data32[1], MASK_DISTANCE_LIGHT);
35          }
36      }
```

These messages are received and are readable in the Vehicle Spy in detail also in the graphical panel.

Figure 8.2: RX message symbol light on the dashboard

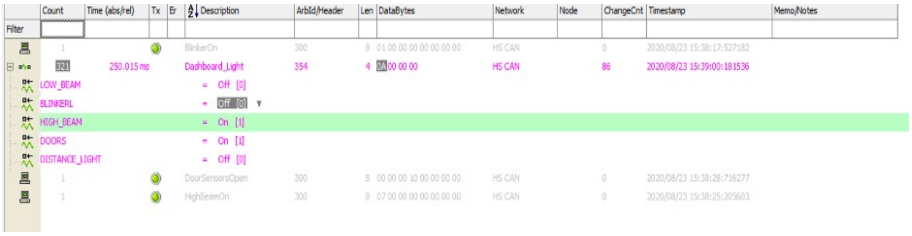

Figure 8.3: RX messages

8.3 CAN received frame

The control panel has different operations, explained at starting the chapter; for these operations, 19 signals are defined. Transmitting has happened from an external node (in this project is the Vehicle Spy). The length of the message and the type of them are defined in the following. FIFO filters messages; therefore, each intended messages are available by the following structure.

Listing 8.4: CAN received frame

```
typedef    struct {
uint16_t TIME;            Time stamp.
uint8_t  OPERATION;       operation type(NORMAL or CANFD)
uint8_t  RTR;             Frame type.
union {
  uint8_t TYPE;           Id type. STD or XTD
  uint8_t IDE;            Id type. STD or XTD
};
union {
  uint32_t SID;           Standard identifier
  uint32_t EID;           Extended identifier
  uint32_t ID;            identifier
};
union{
  uint8_t  DLC;         brief Data Length
  uint8_t  LENGTH;      Data length.
};
union {
  uint8_t  data8[SPC5_CAN_MAX_DATA_LENGHT];          Frame data.
  uint16_t data16[SPC5_CAN_MAX_DATA_LENGHT/2U];      Frame data.
  uint32_t data32[SPC5_CAN_MAX_DATA_LENGHT/4U];      Frame data.
};
```

The function is *can-lld-receive*: "*can-lld-receive(CAND2, CAN-DEDICATED-RXBUFFER, rxf)*;"[17] this function was used by SPC58ECx to receive the messages from a node. A message when it arrives at SPC58EC-DISP, the MCU will get an interrupt. If interrupts are not masked, the MCU will terminate executing the

current instruction and will jump to the beginning instruction of the Callback function.
The CANRxFrame type is assigned to rxf variable: CANTxFrame. The function used to receive is similar to the transmission one.

Listing 8.5: CAN received frame

```
1     typedef    struct s_RxCanCustomDataType
2   {
3     uint8_t LightOperationMsg: 5;      //Byte1
4     uint8_t FollowMeHome: 1;           //Byte2
5     uint8_t Speed: 8;                  //Byte3
6     uint8_t DoorSensor: 8;             //Byte4
7     uint8_t SensorDistance_L:8;        //Byte5
8     uint8_t SensorDistance_R: 8;       //Byte6
9     uint8_t Radar2Distance: 8;         //Byte7
10    uint8_t Radar1Distance: 8;         //Byte8
11  } RxCanCustomDataType
```

As shown in the code below, eight bytes are assigned for theses signals to transmitting from Vehicle Spy to the AEK-MCU-C4MLIT1 board. More further erudition about these signals is explained in the following.

Arb ID	DLC	B1	B2	B3	B4	B5	B6	B7	B8
300	8	LightOpMes	FMH	Speed	Door	SenDist	SenDist	Radar	Radar

8.4 Light operation messages

The first byte of transmission message is assigned to light operation to commands LED driver to Turn to on/ off lights such as Blinker, Low Beam, High Beam, or Turn off all lights at once together. Welcome light is a pattern light designed by the participation of lights to display the situation of lock or unlock the doors of the automobile.

8.5 Follow-Me Home

The second byte of the message is dedicated to the Follow Me Home request, and this request will store on the variable. It will be compared to the current situation if the vehicle is in the parked condition by low RPM Follow Me the related function

	Operation	Status	Decimal	Hex
1	"All Lights"	Off	9	0x9
2	"Blinker"	Blinking	1	0x1
3	"Blinker"	Off	2	0x2
4	"DayLight"	On	3	0x3
5	"DayLight"	Off	4	0x4
6	"LowBeam"	On	5	0x5
7	"LowBeam"	Off	6	0x6
8	"HighBeam"	On	7	0x7
9	"HighBeam"	Off	8	0x8
10	"WellComeLight"	–	18	0x12
"Total number of bits: 5"				

Table 8.2: Light operation messages

will call Follow Me Home commands. Any time this function is called, it will be compared; a valid value comes from sensors that sounded cars. According to the correct direction, the light will be illuminated for a certain amount of times, and it will be turned back to the home position in the centre by calling the calibration function.

8.6 Speed

Speed of the vehicle is an essential parameter during the design of the AFL system; many features are related to this value, such as Follow Me Home and Driving Light that programmed when driver start to driving, Day Light turn to On.

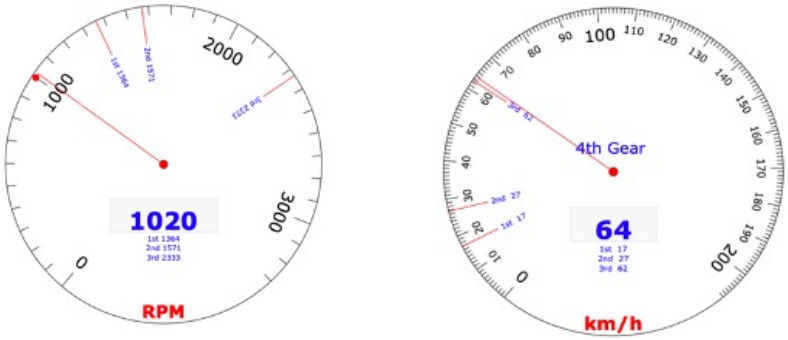

Figure 8.4: RPMs during driving

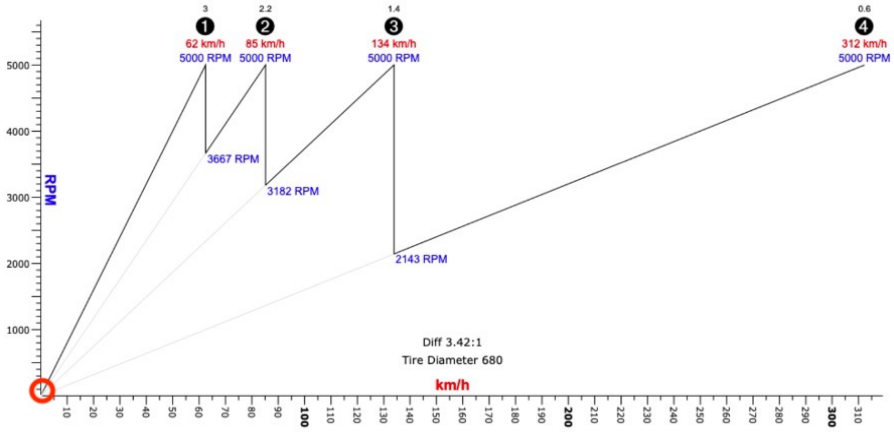

Figure 8.5: RPMs during driving

8.7 Ultrasonic sensor (Distance sensor)

Nowadays, ultrasonic sensors are employed in modern cars to scan the ambient during parking a car or driving. In this project, Park Sensor aides to detect passengers, and it helps Follow Me Home feature to find the correct direction and move light for illumination. Therefore, HRXL-Max Sonar sensors are a perfect decision to use for the simulation; two of them are enough for the front of the car and able to scan a wide range. The HRXL-Max Sonar-WR sensor gives high precision and high-resolution ultrasonic vicinity detection and arranging in air. This sensor line emphasizes resolution, this type that capable of measuring to 5-meter. This ultrasonic sensor detects objects from 1-mm and ranges to objects from 30-cm to maximum range. Objects closer than 30-cm are typically reported as 30-cm. The interface output formats are pulse width, analogue voltage.[**13**] The

MB735 sensors have an RS232 data form (with 0V to Vcc levels), and the sensors have a TTL output. The output is an ASCII capital "," supported by four ASCII character digits describing the range in millimetres, followed by a carriage return (ASCII 13). The maximum range reported is 4999 mm (5-meter models), A range value of 5000 or 9999 matches to no target being detected in the field of view.[18] The serial data format is 9600 baud, 8 data bits, no parity, with a one-stop bit (9600-8-N-1).

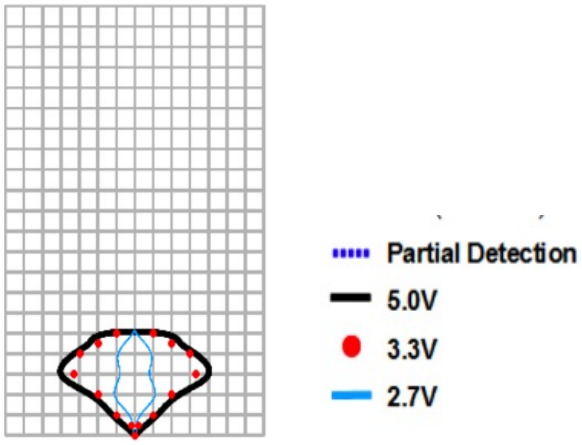

Figure 8.6: RPMs during driving

8.8 Radar Distance

Scan the roads and ambient by Radar present an excellent potential to gather information about the roads and change the position of light systems according to that. The SRR radar systems are considered for simulation. This Radar can recognisee vehicle from different objects, as well as it has an unremarkable object detection by affix. The rugged SRR 2 sensor from A.D.C. estimates independent the distance and speed (Doppler's law) to objects without reflector in one measuring cycle due base of PCM (Pulse Compression Modulation) with high-speed ramps, with real-time scanning of the app. 33/sec.[19] A unique feature of the device is the simultaneous measurement of intervals up to 50 m, relative quickness, and the angle relation among two or some objects.

Sixteen bits are dedicated to transmitting data via CAN communication protocol from Radar to explain the condition of roads, and according to that, the intensity of light will be change. Tow different scenario is defined when roads are free of

Intensity	LUX
Low	800 - 850
Custom	900 - 950
High	1700 - 1800

Table 8.3: Intensity light control

traffic such as highway; in this case, the operation of light will be regular with high intensity. In the second scenario, when Radar sends data that explains a traffic jam, the light systems will be adapted; it means; if the headlight is On by chance, it will be turned to Of; furthermore, the intensity of the Low Beam light will be changed to Low.

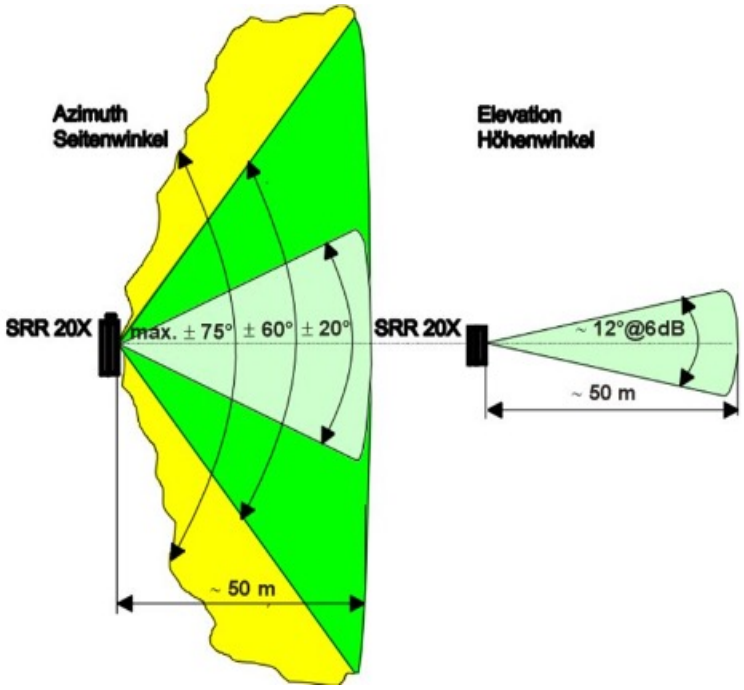

Figure 8.7: The SRR radar systems distance simulation

By calling the function below inside the *Set-ShortDistanceLight()*, the intensity of light will be change to Low.

Listing 8.6: To activate low beam light

```
1       void    ActivateDRL_LowBeam_Low()
2    {
3        AEK_LED_21DISM1drv_DINActiveBuckSPC(AEK_LED_21DISM1_DEV0,
     REG_CR3, rxbuf, DEV1,          BUCK1);          //LowBeam
4        AEK_LED_21DISM1drv_IntensityBuckSPC(AEK_LED_21DISM1_DEV0, LOW
     , DEV1, BUCK1);
5        CAN_Manager_Activation_LowBeam();
6    }
```

Listing 8.7: To activate Short Distance Light

```
1       void    Set_ShortDistanceLight()
2    {
3        if ((getRadar1Distance() != RESET)
4                && (rxData1.Radar1Distance <= Shortdistance))
5        {
6            rxData1.Radar1Distance = RESET;
7            ShortdistanceRadar1 = true;
8            CAN_Manager_Activation_DistanceLight();
9            DeactivateDRL_LowBeam();
10           ActivateDRL_LowBeam_Low();
11           DeactivateDRL_HighBeam();
12           osalThreadDelayMilliseconds(10);
13       }
14       else ShortdistanceRadar1 = false;
15
16       if (((getRadar1Distance() != RESET)
17               && (getRadar2Distance() != RESET))
18               && ((rxData1.Radar1Distance <= Shortdistance)
19               || (rxData1.Radar2Distance <= Shortdistance)))
20       {
21           rxData1.Radar2Distance = RESET;
22           rxData1.Radar1Distance = RESET;
23           ShortdistanceRadar2 = true;
24           CAN_Manager_Activation_DistanceLight();
25           DeactivateDRL_LowBeam();
26           ActivateDRL_LowBeam_Low();
27           DeactivateDRL_HighBeam();
28           osalThreadDelayMilliseconds(10);
29       }
30       else if (((getRadar1Distance() != RESET)
31                   && (getRadar2Distance() != RESET))
32                   && ((rxData1.Radar1Distance > Shortdistance)
33                   || (rxData1.Radar2Distance > Shortdistance)))
34           {
35               rxData1.Radar2Distance = RESET;
36               rxData1.Radar1Distance = RESET;
37               ShortdistanceRadar2 = false;
```

```
38                  ShortdistanceRadar2 = false;
39                  CAN_Manager_DeActivation_DistanceLight();
40                  DeactivateDRL_LowBeam();
41                  ActivateDRL_LowBeam();
42                  osalThreadDelayMilliseconds(10);
43              }
```

Chapter 9

Finite State Machine

A finite state machine can possess various states, and it can change from one state to another state based on internal or external input. This input could be a timer expiry signal, hardware, or software interrupt.[20] In the finite state machine, the procedure to change one state to another state is called transition.

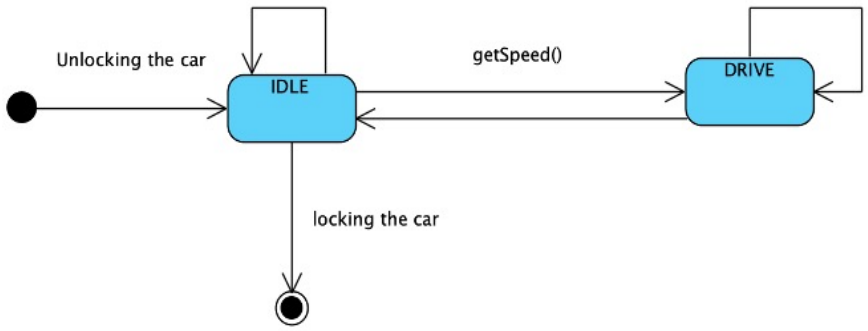

Figure 9.1: States of the AFL system

In this project, two leading states are defined; Idle and Drive; stats are switched between each other according to the speed of the vehicle.
in the Idle state these features are available:

- Welcome light (lock /unlock)
- lighting systems; activate/deactivate low beam, high beam, daylight, blinker
- follow me home
- change position of light by rotating steering wheel.

In the Drive state this feature is available;

- Change the position of light by rotating the steering wheel.
- Lighting systems; activate/deactivate low beam, high beam, daylight, blinker.
- Change the intensity of light according to the situation of the car on the road.

When the vehicle is on Drive mode, the daylight will be activated automatically. As shown in below the table of FSM, each state has an initialization, an entry function, a during function, and an exit function.

Listing 9.1: Table of FSM

```
1        const    FsmState sg_fsmStates[] =
2    {
3 /* FSM_STATE_NONE*/       { NULL, NULL, NULL, NULL},
4 /* FSM_State_IDLE*/       { fsmStateIDLE_Init,fsmStateIDLE_Entry,
5                           fsmStateIDLE_During, fsmStateIDLE_Exit
6                           },
7 /* FSM_State_DRIVING*/    { fsmStateDRIVING_Init,fsmStateDRIVING_Entry,
8                           fsmStateDRIVING_During,fsmStateDRIVING_Exit
9                           },
10   }
```

Chapter 10

Database

A database is an ordered arrangement of data, and A database generator can sort, edit, or serve the information on the database. Platforms link database information together. If a database platform is chosen, all connected database files are loaded into Vehicle Spy. This includes .DBC files for CAN, and diagnostic data saved to the platform. Vehicle Spy 3 can take in immature message data, and Using the Receive table of the Message editor; message decoding tables are created. A database is available, and this data can be imported, saving time by not needing this information transcribed in by a user.

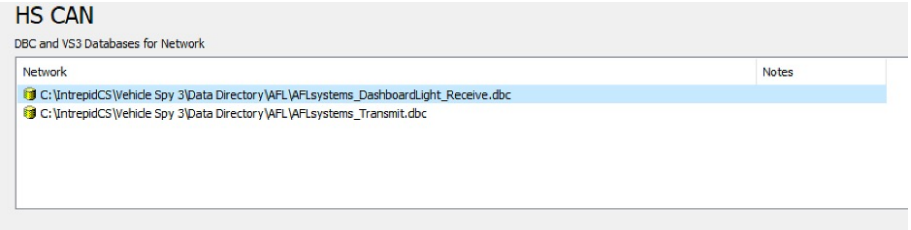

Figure 10.1: AFL systems DBC

Chapter 11

Conclusion and future improvements

At the end of this research, can understand the vehicle headlight should not be a passive tool that can only be switched on/ off. It should be able to adapt to the environment to increase safety in low visibility conditions. Furthermore, an algorithm or a design for adaptive front light systems should not be confined to a single task. It should be able to do many different duties to assist the driver in various road environments.The current conventional light systems can not provide illumination in the correct direction on curve ways. Due to this constrain, a necessity to understand an alternative technology solution.

This thesis introduces the new system which is based on radars and sensor as input to adjust the horizontal rotation of front light, and this afresh proposed Adaptive front lighting system (AFS) assists in improving driver's visibility at night time hence obtaining enhance safety. This Adaptive Front Light design presents unique low beam control over space and time.
We have illustrated the adaptability of the headlight for numerous duties:
Allowing drivers to use high beams without glaring any other driver on the road. It will be done by scanning ambient through radars, and comprehensive decision algorithm could be changed the intensity of light according to environmental condition.
Allowing drivers to see better in curvature, and allowing better illumination of road lanes, sidewalks and dividers. It will be done through curve-adaptive lights. This prototype can speedily react to the road environment in 1 to 2.5 milliseconds thanks to high-speed communication protocols.
Provide essential safety for passengers by configurable and adaptive Follow-Me Home. It helps and prevents from any miss-happenings in numerous day-to-day parking situations.

The future work mainly focuses on to create a comprehensive AFS system which can be suitable for complex road conditions including related to this thesis. Road corner, highway, rural road and urban road. This system relies on data obtained from various sensors, and the radars consider just the next car. A step forward can be achieved by combining computer vision-based image processing algorithms. Instead of the only fixed ultrasonic module also for FMH feature, we can add the radar type mechanism to scan the ambient. By this idea, a neighbouring and backside vehicle can also be traced. A second dimension which can be introduced and realized is the shape of the road. This can be achieved through navigation data.
This analysis was done through Auto Dev kit and did not consider the design of the system; further, the next step could be modifying and improve the components. The critical part of future improvement could be using matrixes of LEDs instance of motor drivers. This alternative is using adaptive lights in the form of LED pixels, which could reduce power consumption and motor delay. Moreover, adaptively headlight will be done faster and more efficient by having several controllable LEDs could turn to off some of them to avoid annoying other drivers or change the intensity of the rest. As a result, the front light system can help drivers in any various condition, at rainy/ snowing night or rural road with the possibility of wild animals cross the roads.
Another modification to improve the system could be done through faster communication protocol such as Automotive Ethernet or CAN - FD. The data over CAN is transmitted in frames. The receiving nodes send an acknowledgement flag when they receive the frame. As this acknowledgement is sent into the transmitted frame, the sender receives an in-frame response after successful transmission. CAN FD solves this challenge by applying two separate frames to transmit the real data and the acknowledgement data. Upper bit rate is used to transmit the data itself, and the nominal bit-rate is allocated to manage data (acknowledgement).
The CAN FD solution can support larger bandwidth than 1 Mbit/s and therefore it could manage data payloads higher than 8 bytes.
The three main differences and improvement of using CAN FD instead CAN are;
1. Increased data rate: From a max of 1 Mb/s to up to 8 Mb/s, the switch is obvious. The increased data rate is beneficial in programming the applications like ADAS, as large data packages are needed to be transmitted.
2. Large Payloads: CAN FD carries 64 bytes of the data field as opposed to 8 bytes in CAN. This massive leap in payload size translates into faster and more efficient in-vehicle network communication among the vehicle ECUs. The end-of-line software upgrade also grows faster affair with CAN FD. With 64 bytes of payload carried, there is no necessity for the splitting of the log messages as well. This simplifies the treatment of the data, thus contributing to inefficient data

transmission.

3. Message Format: The message frame format has supported some significant changes in CAN FD bus standard. The primary being the capability to transition the control data, arbitration and acknowledgement with another bit rate (usually 500 Kb/s) and send the actual data at a higher bit rate. Extra improvement in CAN FD is the introduction of CRC (Cyclic Redundancy Checks), that takes care of the larger frames by using CAN Stuff bits into account.

Bibliography

[1] de Charette - R. Tamburo R.- Barnum - P. C. - Rowe - A. - Kanade - T. - Narasimhan. «S. G.: Fast Reactive Control for Illumination Through Rain and Snow. In: IEEE In- ternational Conference on Computational Photography (ICCP), Seattle, Washington (2012)». In: () (cit. on p. 2).

[2] How It Works: Automatic and January 9). Daddyhood. adaptive headlights. (2019. In: (). URL: https://daddyhood.net/how-it-works-automatic-and-adaptive-headlights-10934.html (cit. on p. 2).

[3] AEK-MCU-C4MLIT1. (n.d.). STMicroelectronics. In: (). URL: https://www.st.com/en/evaluation-tools/aek-mcu-c4mlit1.html (cit. on pp. 2, 15, 32).

[4] A High-Frequency Digitally Controlled LED Driver for Automotive Applications With Fast Dimming Capabilities - IEEE Journals Magazine. (n.d.). Digitally Controlled LED Driver. In: (). URL: https://ieeexplore.ieee.org/abstract/document/6732892 (cit. on p. 2).

[5] HELLA Inc.: HELLA Develops Unique Matrix LED Headlamp System With Audi. Press Release (2014)). In: () (cit. on p. 2).

[6] National Highway Traffic Safety Administration:
bibnamedelimb Traffic Safety Facts 2011: A Com- pilation of Motor Vehicle Crash Data from the Fatality Analysis Reporting System and the General Estimates System. (2011). In: () (cit. on p. 2).

[7] Wang O. and MA (2010) others H. P. A.: A Context-Aware Light Source. In: IEEE International Conference on Computational Photography (ICCP) Cambridge. In: () (cit. on p. 2).

[8] June 9). Model-based design. Wikipedia contributors. (2020. In: (). URL: Wikipedia.%20https://en.wikipedia.org/wiki/Model-based_design (cit. on p. 1).

[9] AEK-MCU-C4MLIT1 datasheet - STMicroelectronics' AEK-COM-BLEV1 evaluation board. (n.d.). AEK-MCU-C4MLIT1. In: (). URL: https://www.digchip.com/datasheets/8877338-aek-mcu-c4mlit1-aek-mcu-c4mlit1.html (cit. on pp. 6, 18, 30).

[10] AEK-LED-21DISM1. (2020). STMicroelectronics. In: (). URL: https://www.st.com/en/evaluation-tools/aek-led-21dism1.html (cit. on p. 8).

[11] AEK-LED-21DISM1 datasheet - STMicroelectronics' AEKD-BLINDSPOTB1 set of assembled. (2019). AEK-LED-21DISM1. In: (). URL: https://www.digchip.com/datasheets/8878252-aek-led-21dism1-digitally.html (cit. on p. 8).

[12] motor driver EVAL board | RS Components. (n.d.). AEK-MOT-SM81M1. AEK-MOT-SM81M1 | AEK-MOT-SM81M1. In: (). URL: https://benl.rs-online.com/web/p/power-motor-robotics-development-tools/1988652/ (cit. on p. 10).

[13] AEK-MOT-SM81M1 - L99SM81V Stepper Motor Driver Evaluation Board. In: (). URL: https://www.evelta.com/aek-mot-sm81m1-l99sm81v-stepper-motor-driver-evaluation-board (cit. on p. 11).

[14] STM-aek-led-21dism1. (n.d.). Stm-Aek-Led-21dism1. In: (). URL: https://www.mouser.com/new/stmicroelectronics/stm-aek-led-21dism1-led-driver-board/ (cit. on p. 11).

[15] SPC5 Software Development Tools. (n.d.). STMicroelectronics. In: (). URL: https://www.st.com/en/development-tools/spc5-software-development-tools.html (cit. on pp. 12, 17, 18, 22, 23).

[16] AEK-CON-AFLVIP2. (n.d.). STMicroelectronics. In: (). URL: https://www.stmicroelectronics.com.cn/ja/evaluation-tools/aek-con-aflvip2.html (cit. on p. 12).

[17] SPC58EC CAN bus configuration. (n.d.). STMicroelectronics. Retrieved 2020. In: (). URL: from%20https://www.st.com (cit. on pp. 13, 27, 51).

[18] Water Level Measurement Product on Alibaba.com. (n.d.). High Resolution Ultrasonic. High Resolution Ultrasonic Level Sensor Ip67 Weather Resistant Bin Tank Water Level Sensor 0 5 Meters Mb7389 - Buy Ultrasonic Level Sensor Water Level Sensor. In: (). URL: https://www.alibaba.com/product-detail/High-Resolution-Ultrasonic-Level-Sensor-IP67_60641399650.html (cit. on p. 55).

[19] RAD-Pluto documentation. (n.d.). RAD-Pluto. In: (). URL: https://cdn.intrepidcs.net/guides/rad-pluto/Introduction-and-Overview.html (cit. on p. 55).

[20] February 6). How to implement finite state machine in C. AticleWorld. https://aticleworld.com/author/pritosh/. (2020. In: (). URL: `https://aticleworld.com/state-machine-using-c/` (cit. on p. 59).