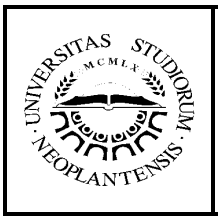


УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА

УНИВЕРЗИТЕТ У НОВОМ САДУ
ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА
НОВИ САД
Департман за рачунарство и аутоматику
Одсек за рачунарску технику и рачунарске комуникације

ЈЕДНО РЕШЕЊЕ УСМЕРАВАЊА У ОКВИРУ ЗОНАЛНИХ КОНТРОЛЕРА У AUTOSAR ПРОГРАМСКОЈ ПОДРШЦИ

Завршни (bachelor) рад

Кандидат: **Милица Војновић**
Број индекса: **РА 59/2020**

Ментор рада: **Проф. др Илија Башичевић**

Нови Сад, јул 2024.

УНИВЕРЗИТЕТ У НОВОМ САДУ ● **ФАКУЛТЕТ ТЕХНИЧКИХ НАУКА**
21000 НОВИ САД, Трг Доситеја Обрадовића 6

КЉУЧНА ДОКУМЕНТАЦИЈСКА ИНФОРМАЦИЈА

Редни број, **РБР**:			
Идентификациони број, **ИБР**:			
Тип документације, **ТД**:		Монографска документација	
Тип записа, **ТЗ**:		Текстуални штампани материјал	
Врста рада, **ВР**:		Завршни (Bachelor) рад	
Аутор, **АУ**:		Милица Војновић	
Ментор, **МН**:		Проф. др Илија Башичевић	
Наслов рада, **НР**:		Једно решење усмеравања у оквиру зоналних контролера у AUTOSAR програмској подршци	
Језик публикације, **ЈП**:		Српски / ћирилица	
Језик извода, **ЈИ**:		Српски	
Земља публиковања, **ЗП**:		Република Србија	
Уже географско подручје, **УГП**:		Војводина	
Година, **ГО**:		2024.	
Издавач, **ИЗ**:		Ауторски репринт	
Место и адреса, **МА**:		Нови Сад; трг Доситеја Обрадовића 6	
Физички опис рада, **ФО**: (поглавља/страна/ цитата/табела/слика/графика/прилога)		7/36/0/2/33/0/0	
Научна област, **НО**:		Електротехника и рачунарство	
Научна дисциплина, **НД**:		Рачунарска техника	
Предметна одредница/Кључне речи, **ПО**:		AUTOSAR, CAN протокол, Етернет протокол, Vector алати, конвертор протокола	
УДК			
Чува се, **ЧУ**:		У библиотеци Факултета техничких наука, Нови Сад	
Важна напомена, **ВН**:			
Извод, **ИЗ**:		Реализација једног решења за усмеравање порука у зоналним контролерима у AUTOSAR програмској подршци. Циљ је развити ефикасно и поуздано решење које омогућава комуникацију између различитих подсистема возила путем CAN магистрале и Етернет мреже, узимајући у обзир захтеве сигурности и перформанси система.	
Датум прихватања теме, **ДП**:			
Датум одбране, **ДО**:			
Чланови комисије, **КО**:	Председник:	Проф. др Мирослав Поповић	
	Члан:	Проф. др Небојша Пјевалица	Потпис ментора
	Члан, ментор:	Проф. др Илија Башичевић	

UNIVERSITY OF NOVI SAD ● **FACULTY OF TECHNICAL SCIENCES**
21000 NOVI SAD, Trg Dositeja Obradovića 6

KEY WORDS DOCUMENTATION

Accession number, **ANO**:			
Identification number, **INO**:			
Document type, **DT**:		Monographic publication	
Type of record, **TR**:		Textual printed material	
Contents code, **CC**:		Bachelor Thesis	
Author, **AU**:		Milica Vojnović	
Mentor, **MN**:		Ilija Bašičević, PhD	
Title, **TI**:		A realization of routing within the zone controllers in the AUTOSAR software	
Language of text, **LT**:		Serbian / cyrillic	
Language of abstract, **LA**:		Serbian	
Country of publication, **CP**:		Republic of Serbia	
Locality of publication, **LP**:		Vojvodina	
Publication year, **PY**:		2024.	
Publisher, **PB**:		Author's reprint	
Publication place, **PP**:		Novi Sad, Dositeja Obradovica sq. 6	
Physical description, **PD**: (chapters/pages/ref./tables/pictures/graphs/appendixes)		7/36/0/2/33/0/0	
Scientific field, **SF**:		Electrical Engineering	
Scientific discipline, **SD**:		Computer Engineering, Engineering of Computer Based Systems	
Subject/Key words, **S**/**KW**:		AUTOSAR, CAN protocol, Ethernet protocol, Vector tools, gateway	
UC			
Holding data, **HD**:		The Library of Faculty of Technical Sciences, Novi Sad, Serbia	
Note, **N**:			
Abstract, **AB**:		Implementation of a solution for routing messages in zone controllers in AUTOSAR software.The goal is to develop an efficient and reliable solution that enables communication between different subsystems of the vehicle via the CAN bus and Ethernet network, taking into account the security and performance requirements of the system.	
Accepted by the Scientific Board on, **ASB**:			
Defended on, **DE**:			
Defended Board, **DB**:	President:	Miroslav Popović, PhD	
	Member:	Nebojša Pjevalica, PhD	Menthor's sign
	Member, Mentor:	Ilija Bašičević, PhD	

Захвалност

Захваљујем се проф. др Илији Башичевићу на указаној сарадњи и стручној подршци приликом израде овог дипломског рада. Захваљујем се и Дарку Спрему на менторству током рада на пројекту.

Посебну захвалност бих изразила према својој породици, пријатељима и колегама на подршци приликом израде рада, као и током целих студија.

САДРЖАЈ

Списак слика

Списак табела

Скраћенице

AUTOSAR (AUTomotive Open System Architecture) – Отворени стандард за израду програмске подршке у аутомобилској индустрији. Истовремено назив глобалног партнерства водећих компанија у индустрији.

CAN (Controller Area Network) – Мрежа контролера, комуникациони протокол

ECU (Electronic Control Unit) – Електронска контролна јединица

ISO (International Organization for Standardization) – Међународна организација за стандардизацију

OSI (Open System Interconnection) – Отворено повезивање система

LIN (Local Interconnect Network) – Локална мрежа унутрашњих веза, комуникациони протокол

MOST (Media Oriented Systems Transport) – Систем преноса медијски оријентисан, комуникациони протокол

CSMA/CD (Carrier Sense Multiple Access / Collision Detection) – Метода вишеструког приступа са ослушкивањем носиоца уз откривање сукобљавања

AMP (Arbitration on Message Priority) – Арбитража о приоритету поруке

CAN FD (Controller Area Network Flexible Data-rate) – Мрежа контролера са променљивом брзином, комуникациони протокол

CAN XL (Controller Area Network eXtended data-field Length) – Мрежа контролера са проширеним пољем, комуникациони протокол

IEEE (Institute of Electrical and Electronics Engineers) - Институт инжењера електротехнике и електронике

SWC (Software Component) – Компонента програмске подршке

BSW (Basic Software) – Основна програмска подршка

RTE (Runtime Environment) – Извршно окружење

PDU (Protocol Data Unit) – Јединица податка протокола

IP (Internet Protocol) - Интернет протокол

TCP (Transmission Control Protocol) – Протокол управљања преносом података

UDP (User Datagram Protocol) – Протокол за размену корисничких датаграма

ID (Identification) - Идентификатор

IPv4 (Internet Protocol version 4) - Верзија 4 интернет протокола

Ipv6 (Internet Protocol version 6) - Верзија 6 интернет протокола

SOF (Start Of Frame) - Почетак оквира

RTR (Remote Transmission Request) - Захтев за удаљени пренос

SRR (Substitute Remote Request) – Замена удаљеног захтева

IDE (Identifier Extension Bit) - Бит проширења идентификатора

DLC (Data Length Code) - Дужина података

CRC (Cyclic Redundancy Check) - Циклична провера сувишности

ACK (Acknowledge) - Потврда

EOF (End Of Frame) – Крај оквира

IFS (Inter-Frame Spacing) - Размак између оквира

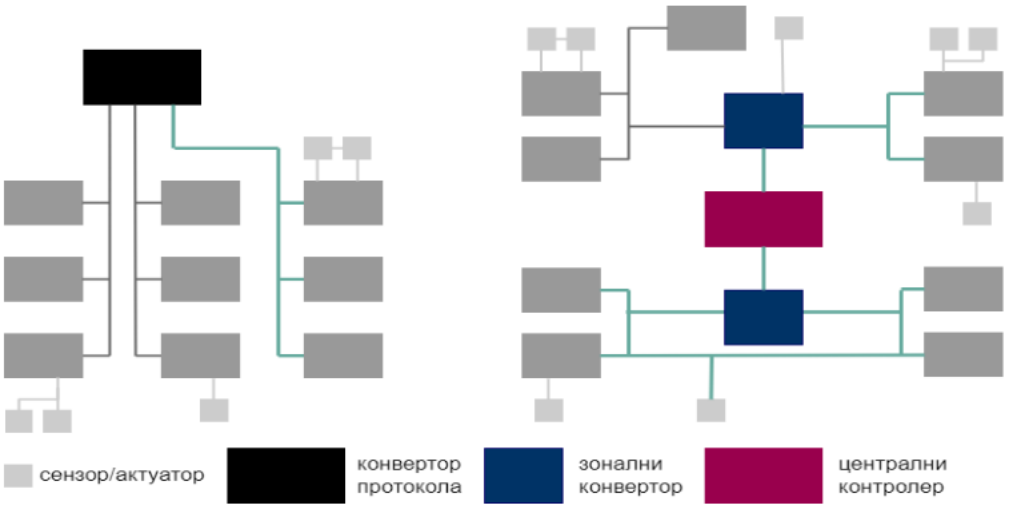
SFD (Start Frame Delimiter) - Граничник почетка оквира

MAC (Media Access Control) - Контрола приступа медијима

1. Увод

Аутомобилска индустрија се развија великом брзином, па тако и аутомобилске технологије које стоје иза ње. Једна од најбитнијих области развоја технологија не само у овој индустрији, већ и у другима, јесте област комуникација. Тренутно индустрија тежи ка томе да са класичног *AUTOSAR*-а пређе на адаптивни *AUTOSAR*, који је намењен да подржи захтевније и комплексније системе у возилима. Једна од главних карактеристика која разликује ове верзије *AUTOSAR*-а је прелазак на Етернет као главни протокол у комуникацији. Свакако, и даље постоји разноликост протокола која омогућава да сваки део возила комуницира користећи протокол која најбоље испуњава његове захтеве, уместо да цело возило користи један тип комуникације. Из овога настаје функционална расподела возила према задацима и комуникационим протоколима који се користе.

Како би функционалне зоне међусобно комуницирале, треба да постоје везивне компоненте између њих. Оне се појављују у виду конвертора протокола (*енг*. *gateway*) који за задатак имају да размењују информације између различитих комуникационих протокола. Идеја је да временом овакве компоненте добију већу улогу поред тога што само прослеђују информације. Иде се ка томе да постану компоненте које ће поред усмеравања такође и да управљају функционалним зонама, ако не и целим системом. Овакве компоненте носе назив **зонални контролери**.



Слика 1.1. Мрежа пре и након увођења зоналних контролера

Идеја дипломског рада је реализовати једно решење за усмеравање порука у зоналним контролерима. Идеја је да контролер води комуникацију између једне зоне која користи *CAN*, и друге која користи Етернет протокол. Инспирација за решење долази из иновација у индустрији које су напоменуте у тексту. На почетку ћемо се упознати са карактеристикама ова два протокола, њихове различитости и која је њихова примена. Решење ће бити имплементирано у једној компоненти, чија је програмска подршка дефинисана *AUTOSAR* архитектуром.

Помоћу ове архитектуре биће развијено идејно решење, са фокусом на модуле који су везани за комуникацију. Користићемо алате направљене да олакашају рад са *AUTOSAR* архитектуром, како би имплементирали нашу идеју. Решење треба да буде ефикасно и поуздано, као и да узима у обзир захтеве сигурности и перформансе система. Након развоја решења, тестираћемо понашање програмске подршке под различитим условима, и извршити потребну валидацију и верификацију. Као крајњи резултат, добићемо компоненту чија је главна особина да прихвата улазне *CAN* поруке, узима њихов садржај и смешта у излазну Етернет поруку.

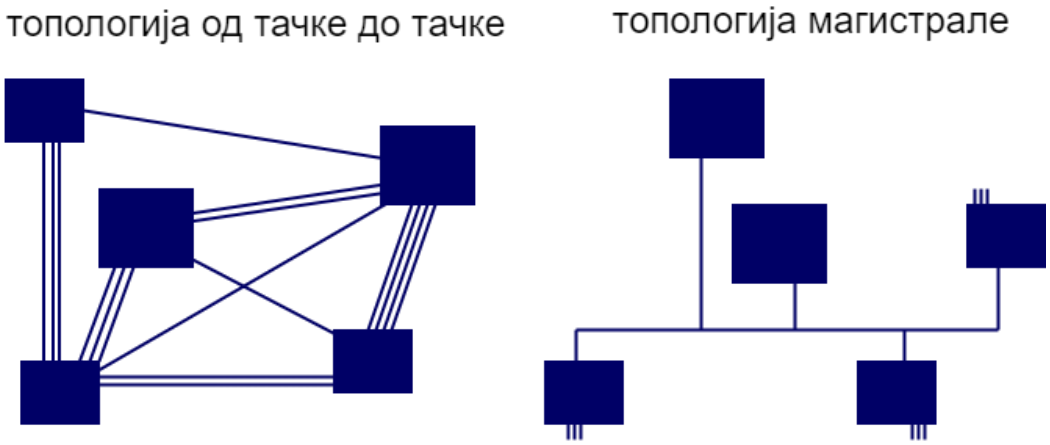
2. Теоријске основе

2.1 Комуникациони системи возила

Возило је машина, чија је основна намена преношење људи и ствари, са једног места на друго. Због своје корисне примене, настаје потреба да се возила развијају, постају комплекснија и количина делова у њима се знатно повећава. Сви ти делови морају да имају неку врсту међусобне сигнализације и управљања. Груписањем свих веза у возилу које преносе податке између ових делова, или компоненти, добијамо један **комуникациони систем возила**.

Потреба за комуникационим системима у возилима настаје када су се механички системи почели замењивати са електронским, почетком 70их, због разних могућности који овакви системи пружају. Уграђени систем који контролише један или више електронских система или подсистема у возилу се назива **електронска контролна јединица** (*енг. ECU*). Електронске контролне јединице су у почетку водиле међусобну комуникацију од тачке до тачке, што се брзо показало као скупо и неефикасано решење. Данашња возила броје између 50 до 150 електронских контролних јединица и готово је немогуће замислити величину и комплексност комуникационог система који користи топологију од тачке до тачке.

Средином 80их појављује се један од првих протокола за комуникацију у возилима са именом ***CAN* протокол**. Овај протокол ће први пут у возилима бити коришћен почетком 90их, а данас је убедљиво најкоришћенији протокол у индустрији. Једна од главних карактеристика која је допринела популарности овог протокола је топологија магистрале.



Слика 2.1. Разлика у топологијама пре и након увођења CAN протокола

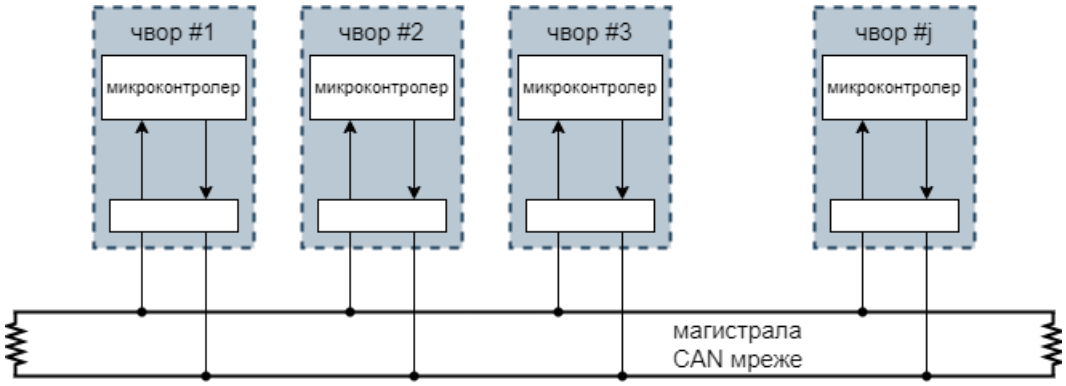
Након њега, појављују се други протоколи којима је у циљу да понуде другачија решења за комуникацију електронских контролних јединица, као и да унапреде неке од карактеристика *CAN* протокола. Неки од њих су: *LIN*, *FlexRay*, *MOST*, Етернет итд. Због својих различитих карактеристика, у возилима се често користи мешавина неколико протокола, а не само тачно један протокол. У овом раду ћемо се фокусирати на *CAN* протокол и Етернет (*енг. Ethernet*).

2.1.1 CAN протокол

CAN протокол описан је *ISO*-11898 стандардом. Дефинише физички слој и слој везе у *ISO OSI* референтном моделу. Према стандарду, протокол је заснован на порукама, дизајниран да дозволи корисницима мреже да комуницирају поуздано.

Као што смо споменули, протокол користи **топологију магистрале** (*енг. bus*); сви чворови су директно повезани на једну **полудуплексну** (*енг. half-duplex*) везу и међусобно су **равноправни** (*енг. multi-master*). Полудуплексна веза представља везу у којој чворови могу међусобно да комуницирају, али да у датом моменту само један може да користи магистралу за слање порука.

Када причамо о чворовима у *CAN* протоколу, мислимо на електронске контролне јединице које су повезане на магистралу. Треба напоменути да се архитектура електронских контролних јединица разликује у зависности од примене, углавном се састоје од микроконтролера који служи за извршавање и контролу операција, док постоје и меморија, сензори, актуатори, примопредајници итд.



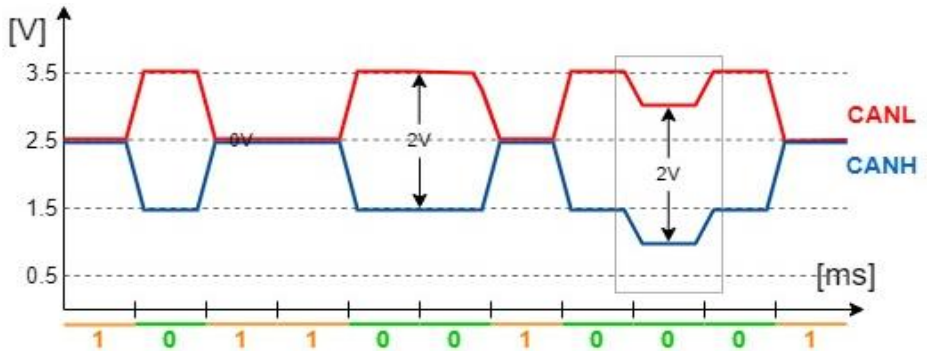
Слика 2.2. CAN магистрала са електронским контролним јединицама

Магистрала је направљена од две уврнуте жице (назваћемо их *CANL* и *CANH*) које представљају један кабл и два отпорника на крајевима, који носе карактеристичну импендансу од 120 Ω. Стандард препоручује кабл максималне дужине од 40 метара, са 30 чворова највише, како би се одржала брзина на магистрали од 1 Mb/s. Вредности две уврнуте жице заједно диктирају стање на магистрали.

стање магистрале	CANL	CANH	CANH - CANL	логичка вредност
рецесивно	≈ 2.5 V	≈ 2.5 V	≈ 0 V	1
доминантно	≈ 1.5 V	≈ 3.5 V	≈ 2 V	0

Табела 2.1. Стања на CAN магистрали

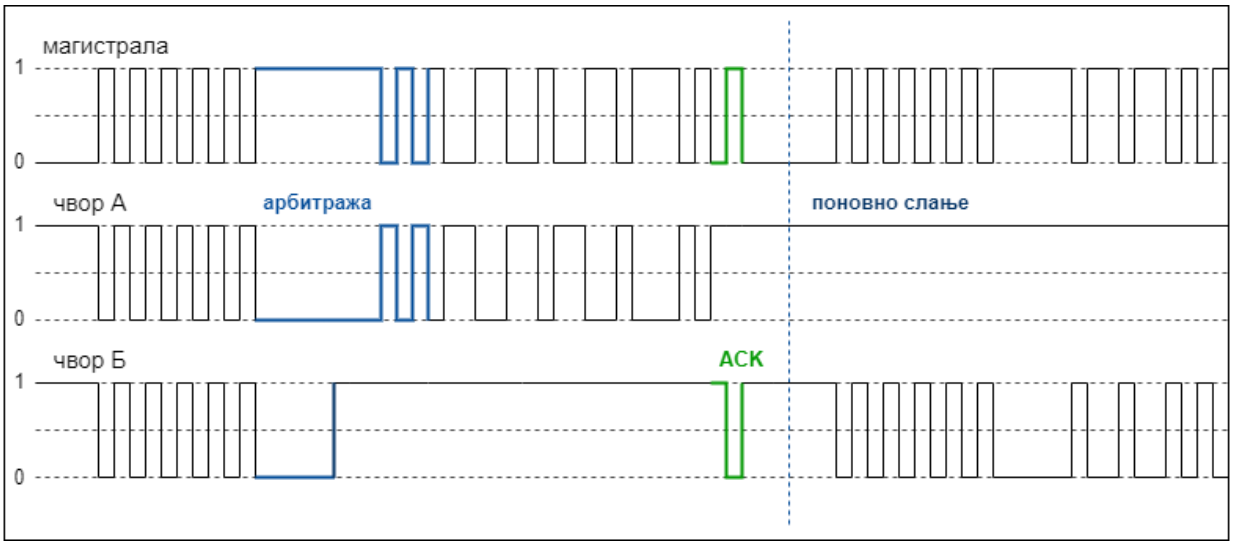
Како би се смањиле сметње и грешке које настају са нестабилном струјом у жицама, користи се диференцијално сигналирање, где разлика у волтажи између две жице заправо представља логичку вредност која се чита са магистрале. Дакле, вредност разлике у рецесивном стању је ≈ 0 V, а у доминантном стању ≈ 2 V. Логика на магистрали је обрнута, рецесивно стање се на магистрали чита као логичка вредност 1, а доминантно стање као логичка вредност 0.



Слика 2.3. Диференцијално сигналирање на CAN магистрали

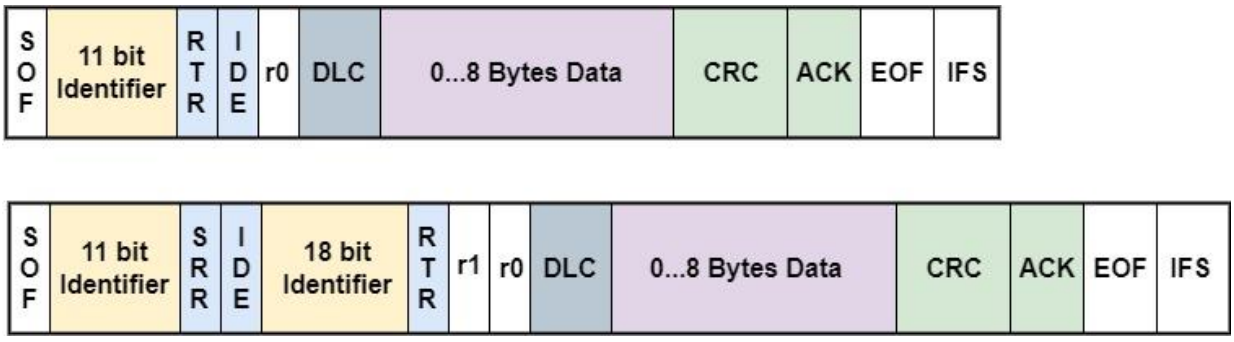
Корисник мреже користи **дифузну емисију** (*енг. broadcast*) за слање кратких порука, које су доступне свим чворовима у мрежи. На основу информације у поруци, корисник ће знати да ли поруку треба да прими или одбаци. Слање поруке засновано је на методи **вишеструког приступа са ослушкивањем носиоца уз откривање сукобљавања, са додатком арбитраже о приоритету поруке** (*енг. CSMA/CD + AMP*).

Арбитража је једна од најбитнијих карактеристика овог протокола. Како би поруке са већим приоритетом и значајем имале предност при коришћењу магистрале, свака порука у себи садржи идентификатор који говори о приоритету поруке. Сваки корисник ослушкује магистралу. Када на магистрали дође до судара две поруке, порука са вишим приоритетом наставља слање, а она друга га прекида и одлаже. Приоритет је већи што је бинарни број идентификатора нижи (идентификатор вредности 0 би представљао највећи могући приоритет). Корисник који је одложио слање поруке поново покушава након неког одређеног времена. Ако се судар понови, корисник одлаже слање са повећаним временом чекања. Време чекања се повећава до одређене границе, након које остаје исто.



Слика 2.4. Пример арбитраже на CAN магистрали

На слици 2.4. имамо пример магистрале са два чвора који шаљу поруке. У истом тренутку чвор А и чвор Б покушавају да пошаљу своје поруке. Порука коју шаље чвор А има идентификатор 00000000101 (*dec.* 5), а порука коју шаље чвор Б има идентификатор 00000100100 (*dec.* 36). Оба чвора прате вредност на магистрали и у моменту када чвор Б примети да се вредност на магистрали не поклапа са његовим идентификатором, он обуставља слање. Арбитражу побеђује порука са мањим идентификатором. Када се заврши слање поруке, чвор Б чека случајно одабрани интервал времена како би поново покушао да пошаље поруку. Такође на примеру можемо да видимо споменуту обрнуту логику на магистрали у односу на чворове.



Слика 2.5. Стандардна и проширена CAN порука

Сама порука коју шаљу корисници има дефинисан изглед оквира у који се податак смешта, с тиме да постоји стандардна верзија поруке у којој је за идентификатор приоритета резервисано 11 бита, и проширена верзија у којој је за идентификатор приоритета резервисано 29 бита. Податак има ограничену величину од 0 до 8 бајтова.

SOF – бит почетка оквира, служи за синхронизацију (1 бит)

Identifier – идентификатор приоритета арбитраже (11 или 29 бита)

RTR (*SRR*) – информација о захтеву одговора другог чвора ако постоји потреба за тиме (1 бит)

IDE – информација о томе да ли је идентификатор стандардан/проширен (1 бит)

r0 (r1) – додатни резервни бит (1 бит)

DLC – информација о величини податка који се преноси (4 бита)

0...8 *Bytes Data* – пренесени податак који је смештен у оквир, величине 0-8 бајтова

CRC – уметнути сувишни бити који служе за проверу исправности пренесене информације (16 бита)

ACK – бит који носи информацију о успешности слања (2 бит)

EOF – бити краја оквира (7 бита)

IFS – бити уметнути за смештање поруке у бафер (7 бита)

Дефинисана су четири различита типа порука:

1. порука информације – најчешћи тип, коришћен за пренос података
2. порука захтева – не носи у себи податке, већ захтева податке од другог чвора
3. порука грешке – порука која се шаље ако је детектована грешка
4. порука преоптерећења – уноси закашњење као решење за густ саобраћај

CAN има неколико начина провере грешки, три на нивоу поруке и два на нивоу бита. Када је грешка примећена, шаље се порука грешке, након које се порука на којој је грешка ухваћена поново шаље. У случају да се примети велики број грешака које долазе са истог чвора, том чвору се одузима могућност слања.

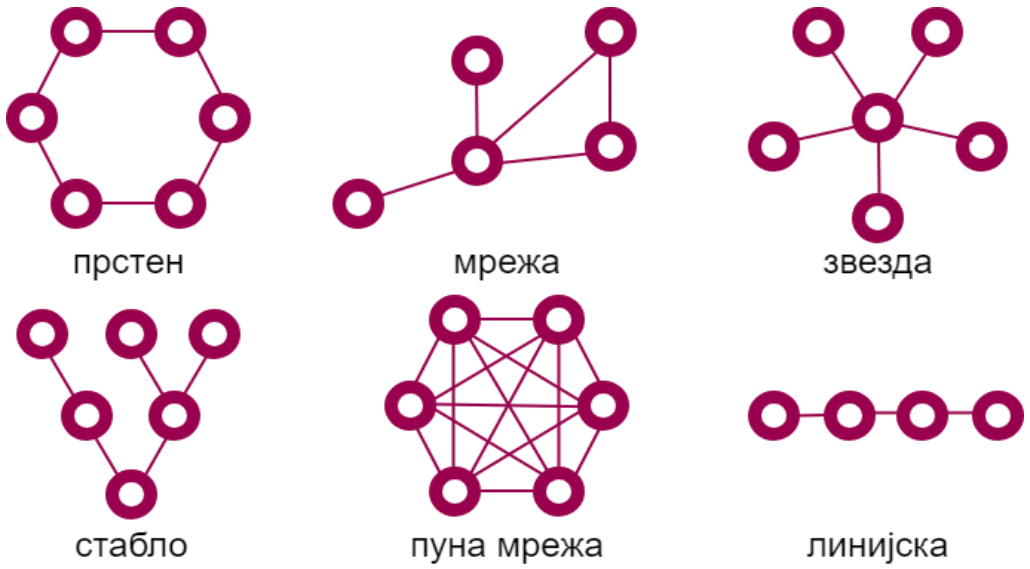
На нивоу поруке, провера се врши помоћу *CRC* и *ACK* бита. Поред тога, проверава се форма оквира. Постоје бити у оквиру чија је вредност по правилу увек рецесивна, а другачија вредност би била показатељ грешке приликом слања; то су *SOF*, *EOF*, *ACK* делимитер и *CRC* делимитер бит. На нивоу бита, примопредајник чвора проверава да ли су бити на магистрали исти са онима који су прочитани (изузеци су идентификатор и *ACK* који по правилу мења вредност). Други начин провере је уметање супротног бита након пет бита истих вредности, јер понављање истих бита указује на грешку.

Временом су се развиле унапређене верзије *CAN* протокола, од којих је *CAN FD* друга генерација оригиналног протокола. Његова главна карактеристика је динамичко мењање брзине слања бита на магистрали; може да се достигне брзина од 5 Mb/s до 8 Mb/s. Такође, величина података која се може пренети у једној поруци повећана је на до 64 бајтова. *CAN XL* је трећа генерација оригиналног протокола, која би требала да повећа брзину и величину његових претходника. Друга генерација *CAN* протокола је почела да се користи у индустрији, док је трећа генерација и даље у развоју.

2.1.2 Етернет у возилима (*енг. Automotive Ethernet*)

Сам етернет, описан великим бројем *IEEE* 802.3 стандарда, је најчешће коришћена технологија у локалним мрежама. По узору на њега, почетком 2000их се јавила идеја да се протокол прилагоди и примени у комуникационом систему возила. Етернет у возилима описан је *ISO*-21111 стандардом. Дефинише исте слојеве *ISO OSI* референтног модела (физички слој и слој везе) као и обичан етернет, с тиме да је физички слој прилагођен за коришћење у возилима.

Идеја за коришћење етернета произилази из жеље за бољим преносом података, који би задовољио потребе нових технологија у возилима. Као пример можемо да узмемо аутономна возила која користе камере; за пренос садржаја из потребних камера је погодан етернет због своје велике пропусности. Неке предности етернета су: већи пропусни опсег, скалабилност, коришћење постојећег поузданог стандарда, коришћење развијених технологија, прилагодљивост итд.



Слика 2.6. Пример неких топологија Етернета

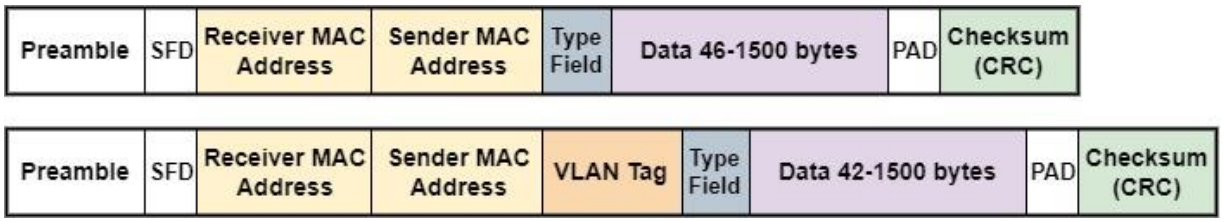
Етернет је прилагодљив протокол који може да поприми **различите облике топологије**, у зависности од потребе. Поруке имају унапред предодређене путање, тј. пренос може да биде **усмерен** (*енг. unicast*), **вишесмеран** (*енг. multicast*), или **дифузан** (*енг. broadcast*) у зависности од адресе која се користи у поруци.

Што се тиче карактеристичног физичког слоја етернета у возилима, користи се један кабл (може бити заштићен или незаштићен) који у себи има две бакарне жице заједно уплетене дужином кабла. Овакав кабл производи мање електромагнетно зрачење и отпорнији је смењама, што га чини поузданијим за пренос података. Постоје различите верзије етернета, са ознакама које говоре о максималној брзини преноса које могу бити постигнуте. У возилима се косите *10Base-TX* (10 Mb/s), *100Base-TX* (100 Mb/s), а у скоријим годинама стандард описује коришћење *1000Base-T* (1 Gb/s). Веза је **дуплексна** (*енг. full-duplex*), дакле поруке се могу слати и примати у истом тренутку, у оба правца.

Слика 2.7. Пренос података преко етернета

Слање порука засновано је на методи **вишеструког приступа са ослушкивањем носиоца уз откривање сукобљавања** (*енг. CSMA/CD*). Корисник који жели да пошаље пакет испитује да ли је канал слободан. Ако утврди да је канал слободан он шаље пакет и наставља да слуша канал, а у супротном чека. Ако је дошло до судара пакета, оба корисника прекидају слање и шаљу обавештење о судару, како би се појачао сам судар. Након тога, чекају случајно одабрани интервал времена и иду у процедуру поновног слања (ретрансмисија). Ако је саобраћај на мрежи густ, шансе за сударе се повећавају, као и време одлагања поновног слања поруке.

Стандард дефинише изглед етернет поруке и информације које оквир треба да има у себи. Постоји везрија оквира где је додат VLAN Tag, који ћемо објаснити у даљем тексту. Податак који се смешта у оквир има ограничену величину од 42 до 1500 бајтова, или од 46 до 1500 бајтова ако не користи VLAN Tag.



Слика 2.8. Етернет порука без и са VLAN Tag

Preamble – преамбула која служи за синхронизацију почетка поруке (7 бајтова)

SFD – почетак оквира (1 бајт)

Receiver MAC Address – одредишна адреса чворова који примају поруку (6 бајтова)

Sender MAC Address – адреса изворног чвора који шаље поруку (6 бајтова)

VLAN Tag – део оквира који се често користи у овој врсти етернета (4 бајтова), састоји се од четири поља (TPID, PRP, DEI, VID). Служи за боље описивање поруке, као што је приоритет и којој области примене припада.

Type Field – описује тип поруке и како да она буде обрађена (2 бајта)

PAD – пуњење које се додаје ако није смештена минимална количина података

Checksum (CRC) – поље генерисано алгоритмом које служи да провери интегритет поруке (4 бајта)

2.1.3 Етернет у поређењу са *CAN* протоколом

Битно је препознати квалитете и мане етернета, као и *CAN* протокола, како би одговарајући протокол искористили на правом месту. Ипак, то су два различита протокола намењена за другачије ствари; *CAN* је направљен за безбедно преношење података у реалном времену, а етернет за преношење велике количине података у разумном временском опсегу.

Једна од већих мана етернета је што не обећава безбедну комуникацију, за разлику од *CAN* протокола. Једина врста провере коју етернет има при пријему поруке је провера интегритета поруке, а у случају грешке та порука се одбацује. За *CAN* смо навели да има пет начина за проверу и исправљање грешака. Иако етернет нуди већи пропусни опсег, у пракси се може показати да ипак *CAN* протокол може брже доставити поруке, у зависности од густине саобраћаја. Велика предност *CAN* протокола је решавање судара на мрежи у реалном времену помоћу арбитраже, док се код етернета у случају судара сви преноси заустављају и тиме успоравају мрежу. Приоритет је основна карактеристика *CAN* протокола и самим тиме слање порука је организовано, док код етернета у свом основном облику нема приоритет, али може да буде додат. Етернет захтева опширнију програмску подршку и може бити скупље решење.

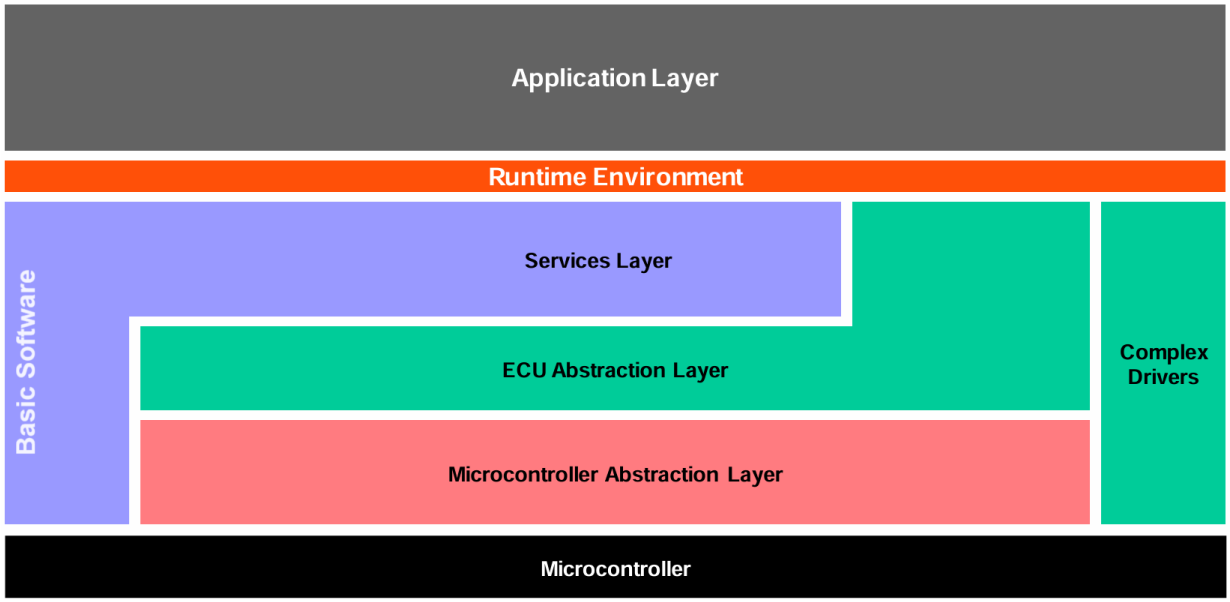
	CAN протокол	Етернет протокол
топологија	магистрала	линијска, звезда, прстен, стабло, мрежа
начин преноса	дифузан	дифузан, вишесмеран, усмерен
тип везе	полудуплексна	дуплексна
метода слања порука	CSMA/CD + AMP	CSMA/CD
пропусни опсег	1 Mb/s	10 Mb/s, 100 Mb/s, 1 Gb/s
величина податка	0-8 B	42-1500 B, 46-1500 B
веза чворова	више чворова	од тачке до тачке

Табела 2.2. Поређење неких карактеристика датих протокола

2.2 AUTOSAR

AUTOSAR је глобално партнерство водећих компанија у аутомобилској индустрији. Њихов циљ је да направе, развију и стандардизују **архитектуру програмске подршке електронских контролних јединица**. Идеја је да архитектура буде универзална за све платформе, како би програмска подршка била преносива, скалабилна и како би се развила сарадња између више партнера.

Како би се обезбедило да програмска подршка буде универзална, она мора бити независна од физичке архитектуре и платформе. Ово се постиже дефинисањем **слојевите архитектуре програмске подршке**. Концепт слојевите архитектуре је да се компоненте поделе у логичке групе на основу типа функционалности које пружају или на основу њихове интеракције са другим компонентама, тако да се међуслојна комуникација јавља само између суседних слојева.



Слика 2.9. Слојевита архитектура програмске подршке

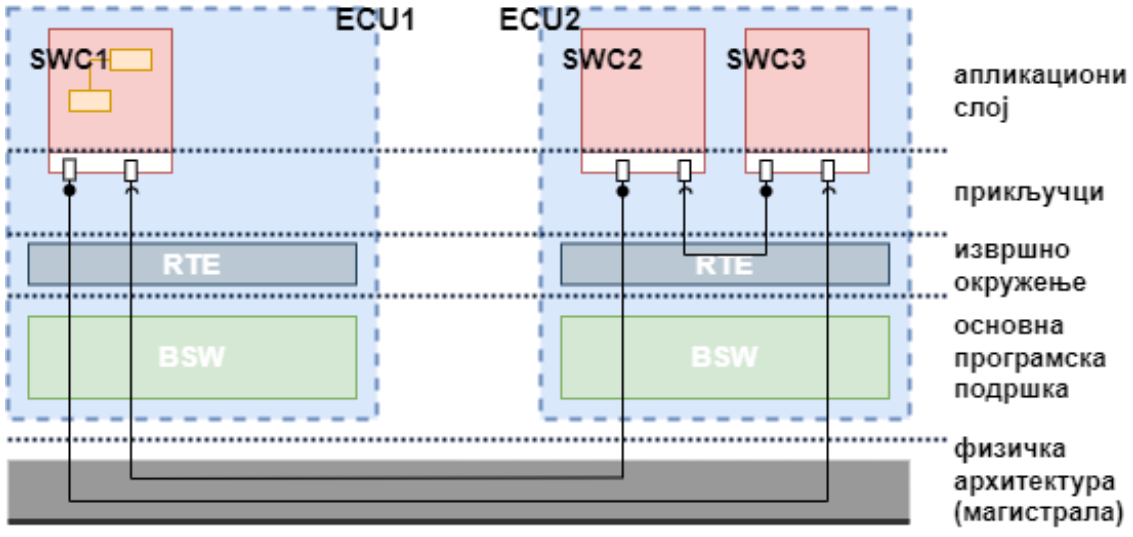
Програмска подршка смештена је на микроконтролеру електронске контролне јединице и дели се на три основна слоја: **апликациони слој** (*енг. Application Layer*), **извршно окружење** (*енг. Runtime Environment*), **основна програмска подршка** (*енг. Basic Software*). Основна програмска подршка се даље дели на четири слоја: **слој услуга** (*енг. Services Layer*), **слој апстракције електронске контролне јединице** (*енг. ECU Abstraction Layer*), **слој апстракције микроконтролера** (*енг. Microcontroller Abstraction Layer*), **сложени управљачи** (*енг. Complex Drivers*).

Слој апстракције микроконтролера (*енг. Microcontroller Abstraction Layer*) – одмах изнад физичке архитектуре, пружа приступ функционалностима микроконтролера, чини више слојеве независне од физичке архитектуре миктроконтролера.

Слој апстракције електронске контролне јединице (*енг. ECU Abstraction Layer*) – доставља горњим слојевима посебне услуге електронске контролне јединице, чини више слојеве независне од физичке архитектуре електронске контролне јединице.

Слој услуга (*енг. Services Layer*) – осигурава основне услуге за апликацијски слој и за друге слојеве основне програмске подршке, као што су меморијске услуге, дијагностричке услуге, комуникацијске услуге, управљање мрежом итд.

Сложени управљачи (*енг. Complex Drivers*) – слој за управљање сложених сензорских и актуаторских целина, као и за имплементацију нестандардизованих уређаја.

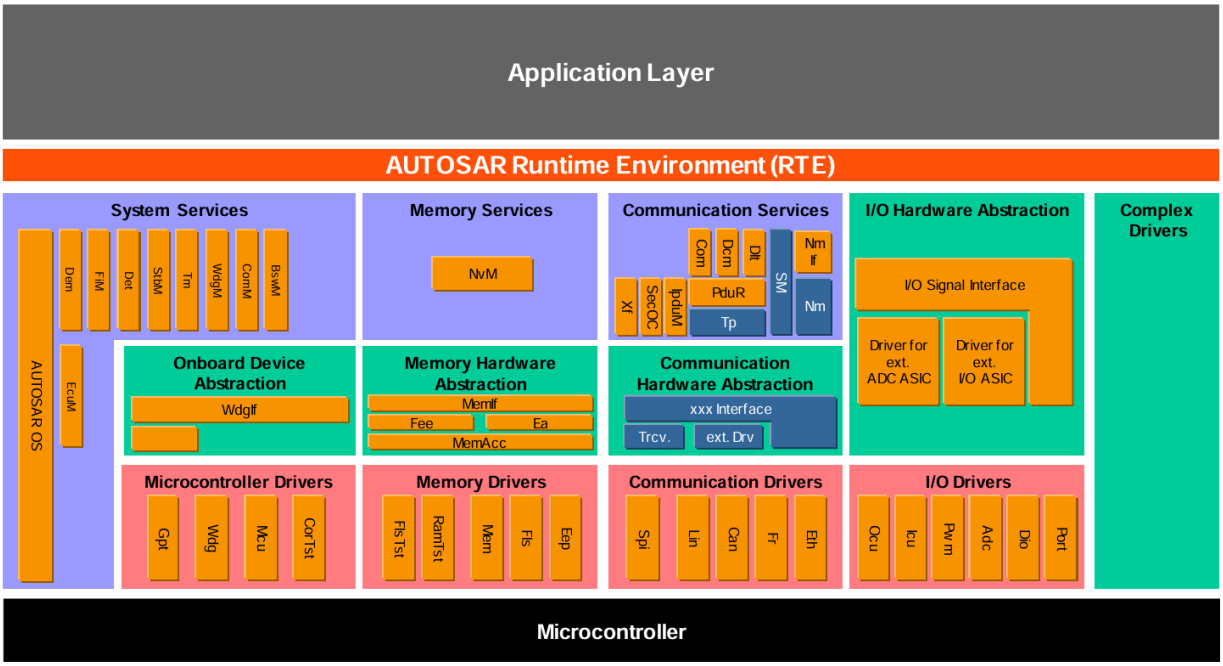


Слика 2.10. Пример комуникације компоненти програмске подршке

Апликациони слој (*енг. Application Layer*) – садржи *компоненте програмске подршке* (*енг. SWC*) које енкапсулирају неку функционалност електронске контролне јединице. Ове компоненте могу да буду атомске или да у себи садрже неколико мањих компоненти које су међусобно повезане (*енг. runnable*).

Извршно окружење (*енг. Runtime Environment*) – представља посредника који пружа комуникацијске услуге за апликације које су независне од физичке архитектуре. Испод њега се налази слојевита архитектура, а изнад њега она прелази у архитектуру подељену по компонентама програмске подршке. Овај слој такође пружа међусобну комуникацију између компоненти програмске подршке, која може бити унутрашња (комуницирају компоненте из исте електронске контролне јединице) или спољашња (комуницирају компоненте из различитих електронских контролних јединица). Комуникација компоненти програмске подршке врши се помоћу прикључака (*енг. port*).

Слојеви основне програмске подршке се даље деле у функционалне групе као што су системски, меморијски, криптографски, комуникациони, улазно/излазни сервиси. Функционалности ових сервиса су имплементиране у оквиру бројних **модула**. Модули су битни за организацију и структуирање програмске подршке електронске контролне јединице.

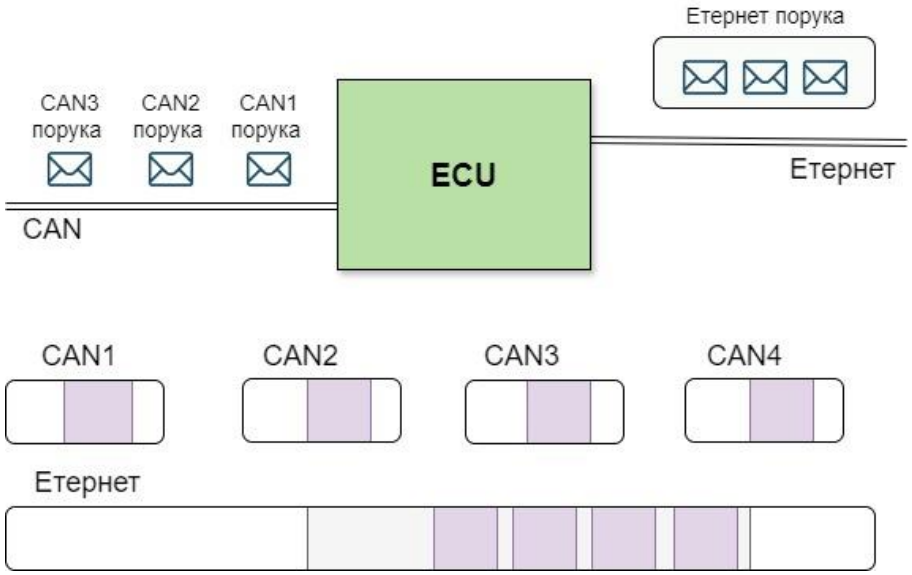


Слика 2.11. Модули мапирани према функционалностима

3. Концепт решења

3.1 Идеја решења представљена на *AUTOSAR* слојевитој архитектури програмске подршке за *ECU*

Задатак овог дипломског рада, као што смо већ споменули у уводу, је реализовати електронску контролну јединицу (*енг. ECU*) која ће се понашати као **конвертор протокола**. Електронска контролна јединица треба да прима *CAN* поруке, препакује у Етернет поруке и такве да их пошаље.



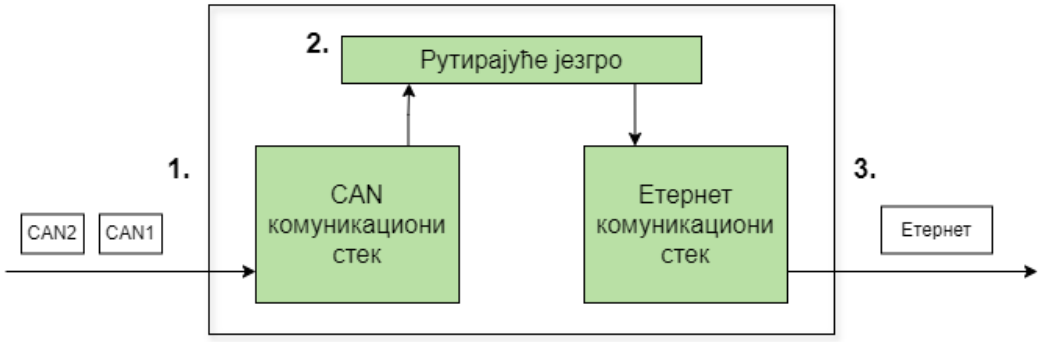
Слика 3.1. Сликовити приказ рада наше електронске контролне јединице

Сам пролазак и обрада порука у *AUTOSAR* архитектури може да се подели на неколико кључних етапа које ће бити описане на датом примеру:

1. На *CAN* магистрали се појављују два *CAN* оквира које наш *ECU* прима као улазне параметре. За пријем, обраду и прослеђивање порука ка вишим слојевима архитектуре (у овом случају до рутирајућег језгра) задужен је ***CAN* комуникациони стек модула**.

2. **Рутирајуће језгро** ове две распаковане поруке које прима са *CAN* стека истим редоследом усмерава ка Етернет комуникационом стеку.

3. У **Етернет комуникационом стеку** поруке се пакују у један Етернет оквир, спуштају се ка нижим слојевима и шаљу на Етернет магистралу као излазни параметри.

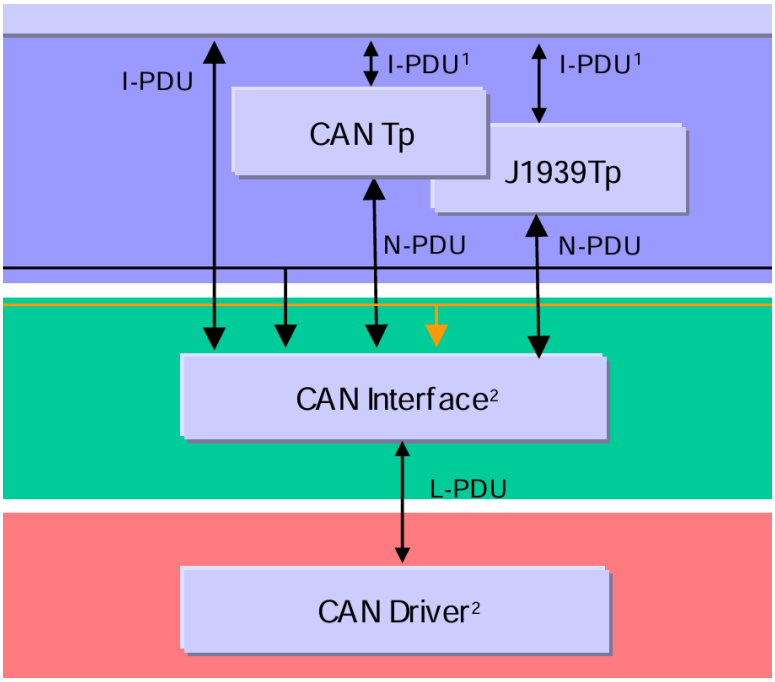


Слика 3.2. Главне етапе проласка порука кроз електронску контролну јединицу

3.1.1 *CAN* комуникациони стек

Пре објашњења *CAN* комуникационог стека, битно је објаснити како од сигнала са апликационог слоја добијамо поруку на магистрали неког комуникационог протокола. Приликом проласка сигнала кроз протокол стек, ка нижим нивоима, додају се заглавља са информацијама. Са друге стране, приликом проласка поруке навише у протокол стеку, заглавља се одбацују. Ова метода се зове **енкапсулација**. Сигнали заједно са заглављима носе назив *PDU* и представљају јединствену јединицу протокола која се преноси између модула.

Податак са апликационог слоја до слоја основне програмске подршке долази у виду сигнала. Када се сигналу дода загљављање (ово обавља *COM* модул) оно носи назив ***I-PDU***. Када се *I-PDU* спусти до *TP* модула одређеног протокола, носи назив ***N-PDU***. Након тога, у *Interface* и *Driver* модулима он носи назив ***L-PDU***. После тога, *L-PDU* се спушта на физички слој, тј. на магистралу протокола.



Слика 3.3. Модули у *CAN* комуникационом стеку

CAN поруку са *CAN* магистрале прихвата ***CAN Driver***, модул који је задужен за приступ физичкој архитектури. После њега налази се модул ***CAN Interface*** који представља интерфејс за услуге *CAN Driver* модула за горње комуникационе слојеве. ***CAN Transport Layer (TP)*** је модул чија је главна улога да раставља/саставља *CAN PDU*-ове који су дужи од предодређене дужине, а у случају обичног *CAN* протокола то би било 8 бајтова. ***CAN Network Management (NM)*** надгледа и организује комуникацију на мрежи и њена стања. ***CAN State Manager (SM)*** служи за мењање начина комуникације на мрежи, тј. управља контролом тока мреже.

3.1.2 Рутирајуће језгро (*енг. PDU Router*)

Овај модул је место сакупљања свих комуникационих протокола и њихових стекова. Испод овог модула су *PDU*-ови рутирани ка модулима везаним за специфичне протоколе, док су изнад њега модули независни од протокола. Његова главна улога је да рутира *PDU*-ове према модулима изабраног протокола.

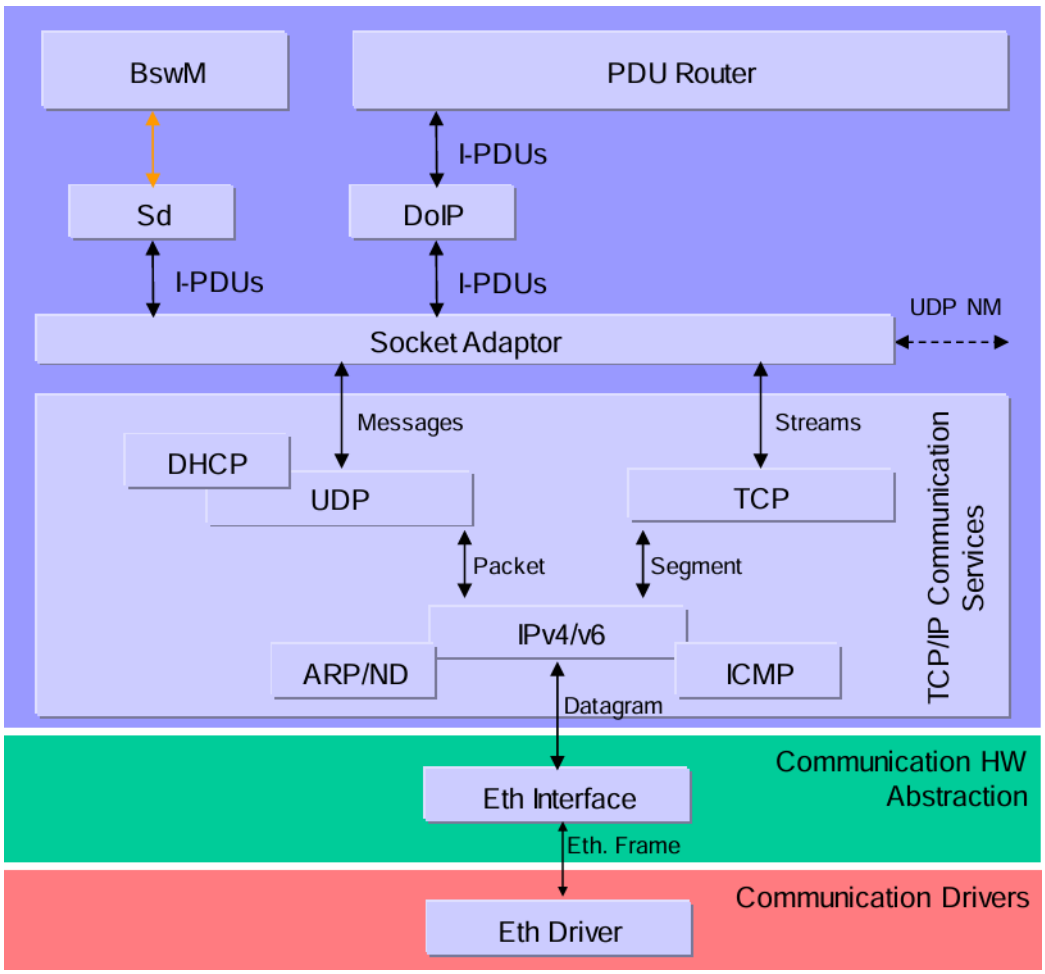
Рутирање се врши помоћу статички конфигурисаних табела рутирања и помоћу статички додељених идентификатора који носи сваки *PDU*. Сам модул не прави никакве измене над *PDU*-овима, само их прослеђује. Подржава два типа рутирања; први тип је рутирање између модула изнад и модула испод језгра; други тип је рутирање између два модула који су везани за различите протоколе.

Други тип рутирања је битан јер омогућава нашем *ECU*-у да се понаша као конвертор протокола, без да се *PDU* прослеђује скроз до апликативног слоја и да се тамо обрађује, а самим тиме се не користи *SWC*. Дакле, након што рутирајуће језгро прими *PDU* од *CAN* комуникационог стека, идеја је да се он одмах проследи ка Етернет комуникационом стеку.

3.1.3 Етернет комуникациони стек

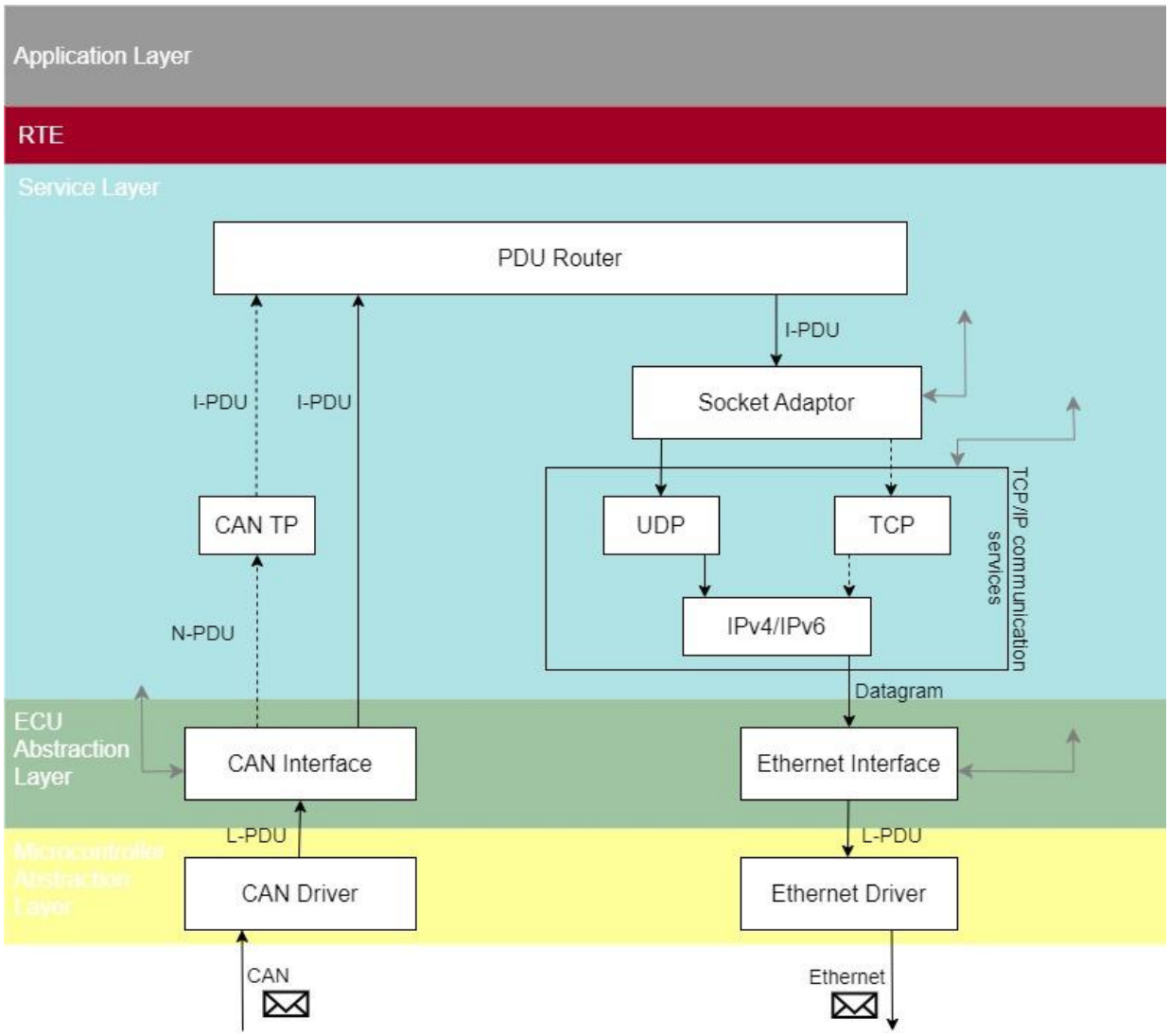
Етернет комуникациони стек користи *TCP/IP* фамилију протокола за успоставу и формирање везе између корисника. Као крајње тачке комуникационог тока између два корисника мреже користе се мрежни сокети. Ово је уједно и разлог зашто рутирајуће језгро не може директно да рутира *PDU*-ове према *Ethernet Interface* модулу, као што је случај код осталих протокола.

Због тога постоји ***Socket Adaptor (SoAd)***, модул који прима *I-PDU*-ове од рутирајућег језгра. Његов главни задатак је да мапира *ID*-ове надолазећих *I-PDU*-ова на мрежне сокете који имају отворене везе. Што се тиче интернет протокола, за потребе нашег задатка је довољно да користимо ***IPv4*** тип адресирања. Такође ћемо користити ***UDP***, протокол без успоставе везе и без гаранције испоруке пакета. *PDU* се спушта до ***Ethernet Interface*** модула у виду датаграма (добио је заглавље са свим потребним информацијама за ову врсту комуникације), који обезбеђује горњим слојевима интерфејс независан у односу на физичку архитектуру. *Ethernet Interface* не приступа директно физичкој архитектури, већ то ради ***Ethernet Driver*** модул.



Слика 3.4. Модули у Етернет комуникационом стеку

Пређени су најбитнији модули потребни у изради нашег задатка, конкретно они који су везани за комуникацију. Цела ова идеја за AUTOSAR слојевиту архитектуру програмске подршке за електронску контролну јединицу може бити представљена у виду следећег дијаграма.



Слика 3.5. Целокупан пролазак кроз комуникационе модуле

3.2 Одабир алата и процес рада

Vector Informatik је фирма која је развила своје решење за *AUTOSAR* архитектуру и њену примену. Идеја је да се поједностави развојни процес, побољша квалитет програмске подршке и свеукупно побољша поузданост и перформансе електронских контролних јединица у возилима.

За израду ћемо користити њихову свеобухватну платформу програмске подршке која обезбеђује модуле и компоненте усаглашене са *AUTOSAR*-ом; она се зове ***MICROSAR***. Користићемо и алате за развој и интеграцију програмске подршке наше електронске контролне јединице. Укратко ћемо представити све алате које планирамо да користимо, као и где се они налазе у процесу рада.

Слика 3.6. Развој решења програмске подршке

Vector DaVinci Developer Classic, алат дизајниран за развој програмске подршке у уграђеним системима, посебно у аутомобилској индустрији. Олакшава развој апликативног слоја са графичким дефинисањем компоненти програмске подршке (*SWC*) и њихових веза. Нуди функције за интеграцију, тестирање и отклањање грешака. Обезбеђује окружење за ефикасно пројектовање и компатибилност са другим Vecor алатима.

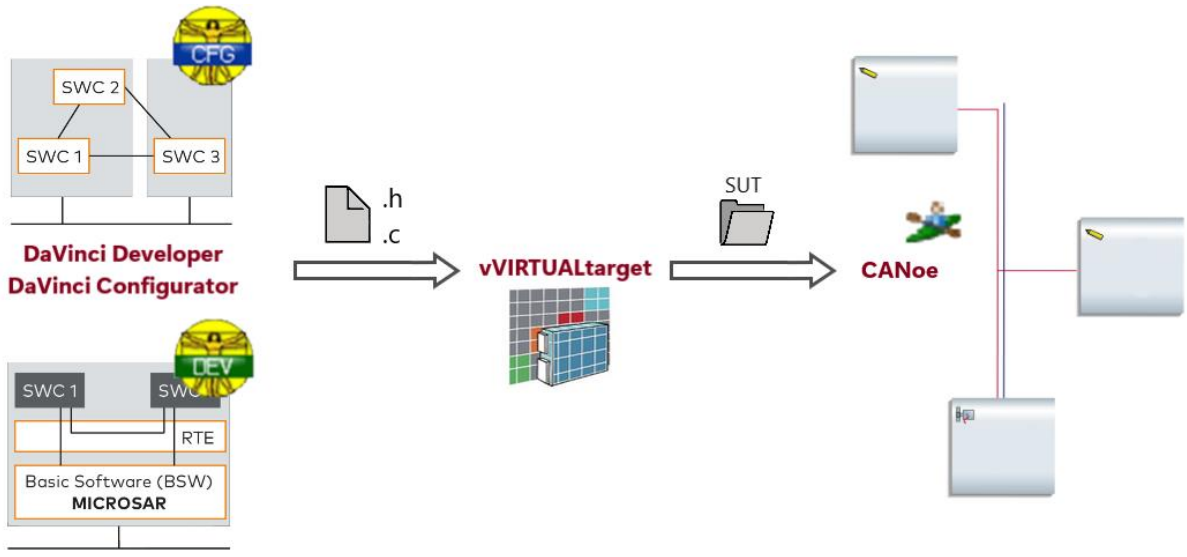
Vector DaVinci Configurator Classic, алат за конфигурисање основне програмске подршке (*BSW*) и извршног окружења (*RTE*). Садржи преглед свих модула, чије параметре и дефиниције можемо да мењамо. На основу дефинисане конфигурације генерише се решење програмске подршке. Обезбеђује валидацију конфигурације и друге функционалности као помоћ при развоју.

Vector vVIRTUALtarget, алат за виртуализацију појединих компоненти програмске подршке и потпуно конфигурисане електронске контролне јединице за све типичне AUTOSAR пројекте. Користи се у фази развоја и тестирања. Креира се виртуални *Systems Under Test* (*SUT*), тј. виртуални *ECU*.

Vector CANoe, алат тестирање и анализу појединачних *ECU*-ова и читавих мрежа у возилима. Омогућава интерактивну симулацију понашања *ECU*-ова на мрежи у реалном времену. Нуди графички интерфејс који олакшава тестирање, као и многе друге функционалности.

При раду са физичким *ECU*-ом, интегрише се програмско решење генерисано у *Configurator*-у. За потребе овог дипломског, одлучили смо да симулирамо окружење на Windows x86 плаформи. За интеграцију програмског решења у наш виртуални *ECU* користимо .dll датотеку генерисану из *vVIRTUALtarget*-а и налази се у *SUT*-у.

Сам процес рада почиње од *DaVinci Developer*-а и *DaVinci Configurator*-а. У *DaVinci Developer*-у дефинишемо структуру нашег апликативног слоја. У *DaVinci Configurator*-у конфигуришемо основну програмску подршку и извршно окружење. На основу целе слојевите архитектуре коју смо развили, генерише се код у *C* програмском језику. Прелазимо у *vVIRTUALtarget* како би генерисали виртулни *ECU* који нам је потребан за тестирање функционалности. У *CANoe*-у имплементирамо окружење које ћемо користити за тестирање функционалности наше електронске контролне јединице.

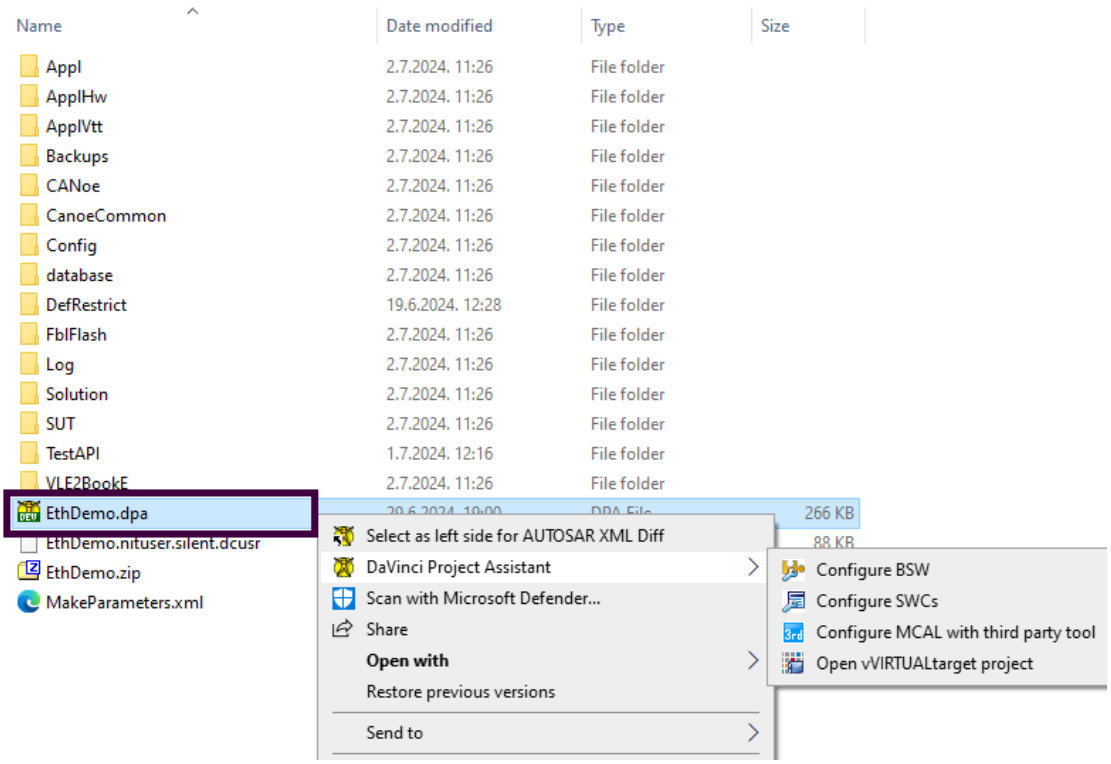


Слика 3.7. Процес рада са алатима

4. Програмско решење

4.1 Конфигурисање и генерисање програмског решења

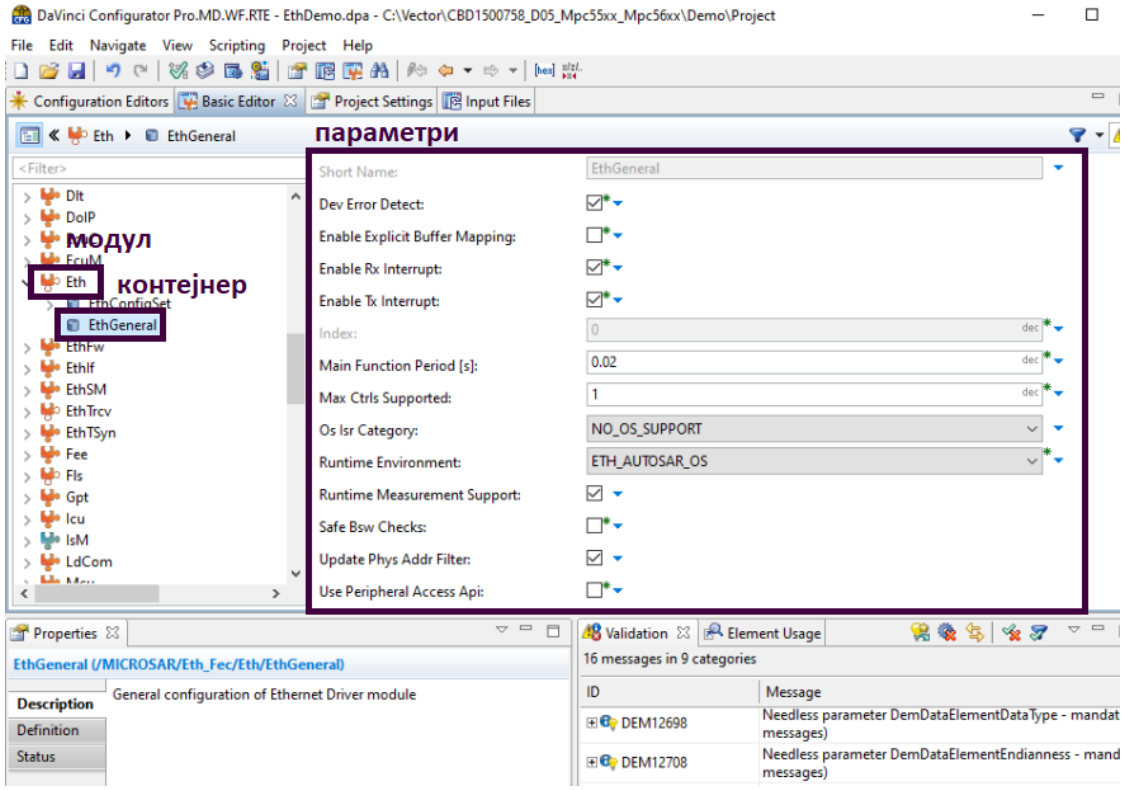
Као што је споменуто у концепту решења, први корак је израда конфигурације у *DaVinci Configurator*-у и *DaVinci Developer*-у. Иако је могуће да се направи нови пројекат, обично се не практикује, већ се користи демонстрациони пројекат који Vector доставља заједно са алатима и документацијом. За потребе нашег пројекта одабрали смо да користимо демонстрациони пројекат ***EthernetDemo***.



Слика 4.1. Демонстрациони пројекат EthernetDemo

EthDemo.dpa је векторова коренска датотека, помоћу које се покреће *Configurator* (*Configure BSW*) и *Developer* (*Configure SWCs*). Друге датотеке ће по потреби бити објашњене у даљем тексту. *Developer* окружење можемо да занемаримо, зато што наш *ECU* нема финкционалности везане за апликациони слој.

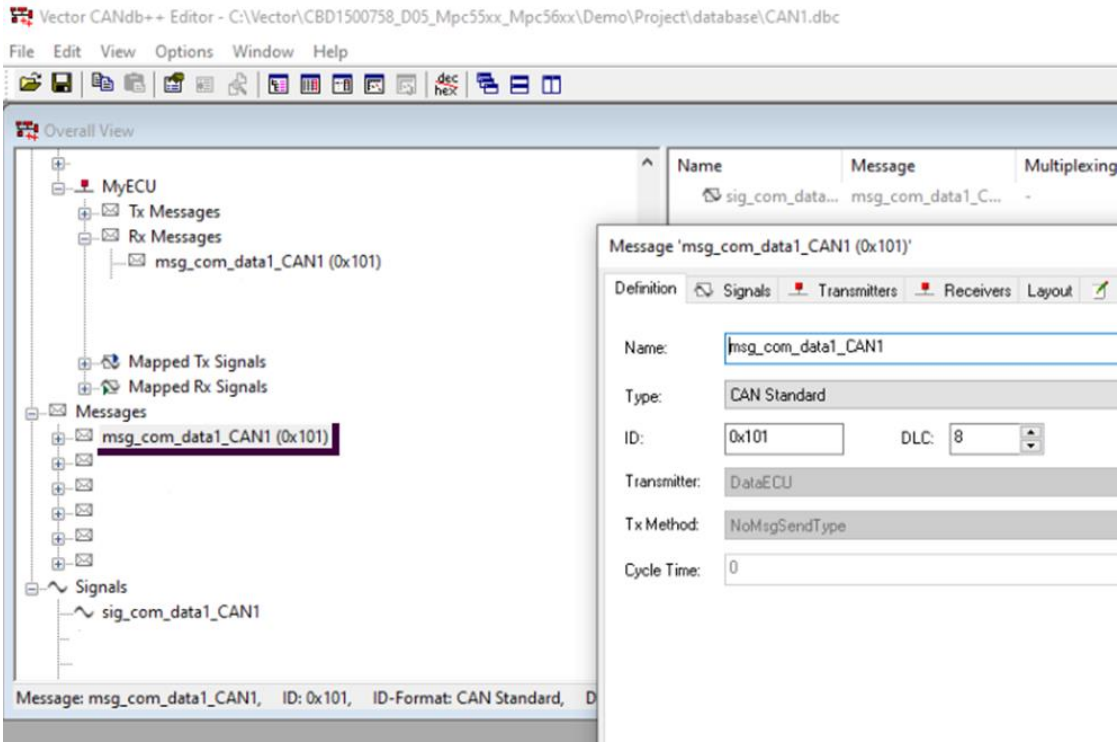
Радно окружење *Configurator*-а у себи садржи преглед свих модула који су укључени у конфигурацију. У себи садрже контејнере конфигурације модула, који у себи имају дефиниције параметара. Приликом генерисања, модули представљају датотеке програмског решења, а контејнери и дефинисани параметри представљају сам програмски код у тим датотекама.



Слика 4.2. Радно окружење DaVinci Configurator-а

Почињемо са ***CanIf*** модулом. Треба да се направи контејнер који ће представљати пријем *CAN* поруке са магистрале. Контејнер може да се направи ручно, или да буде генерисан, што се и практикује. Генерисање се врши тако што се конфигурација синхронизује са додатом датотеком у којој су претходно дефинисане поруке.

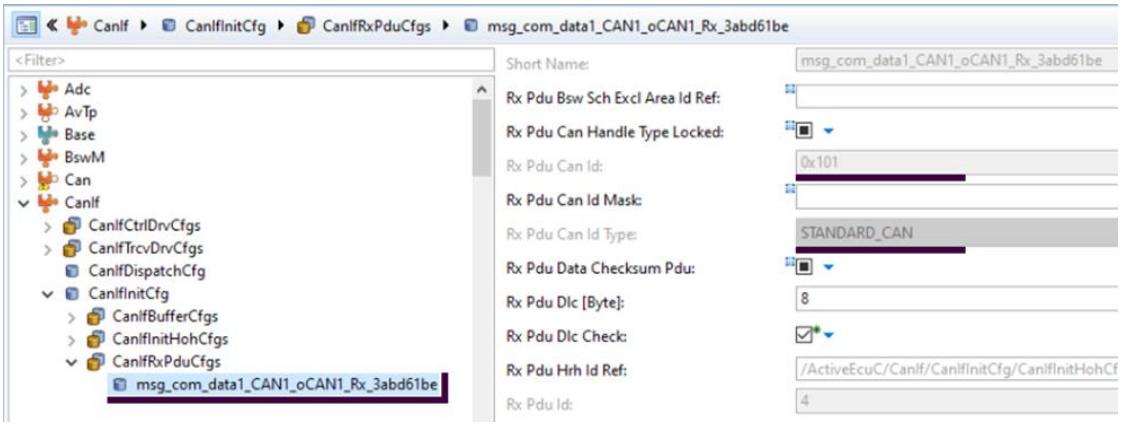
Једна врста датотеке у којој се дефинишу поруке је текстуална датотека .dbc (*CAN* база података), која се прави помоћу *Vector CANdb++* алата.



Слика 4.3. CAN1.dbc датотека

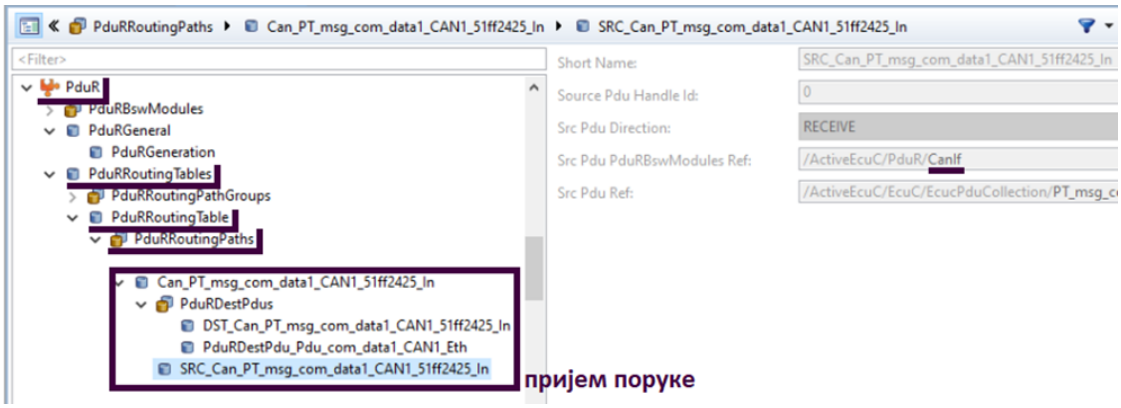
На слици можемо да видимо ***CAN1.dbc*** датотеку коју смо креирали. Направљена је `msg_com_data1_CAN1` порука, са дефинисаним идентификатором (0x101), типом поруке (*CAN* стандардни) и величином информације коју може да смести (8 бајтова). Порука је мапирана као улазна (*Rx*) порука нашег *ECU*-а, који је овде представљен као чвор `MyECU`. На поруку је мапиран сингал `sig_com_data1_CAN1`. Тај сигнал такође носи информације о себи као што су дужина (64 бита), тип вредности (неозначен) итд.

Направљену датотеку додајемо у пројекат (*Project -> Input Files -> Add, remove or modify legacy files*) и синхронизујемо га са изменама (*Update the configuration now to commit the project modifications*). Након овог корака, генеришу се контејнери у конфигурацији за додату поруку, конкретно контејнери у *CanIf* и *PduR* модулима. У генерисаном `msg_com_data1_CAN1_oCAN1_Rx` контејнеру можемо да видимо дефиницију поруке која се преузела из CAN1.dbc датотеке.



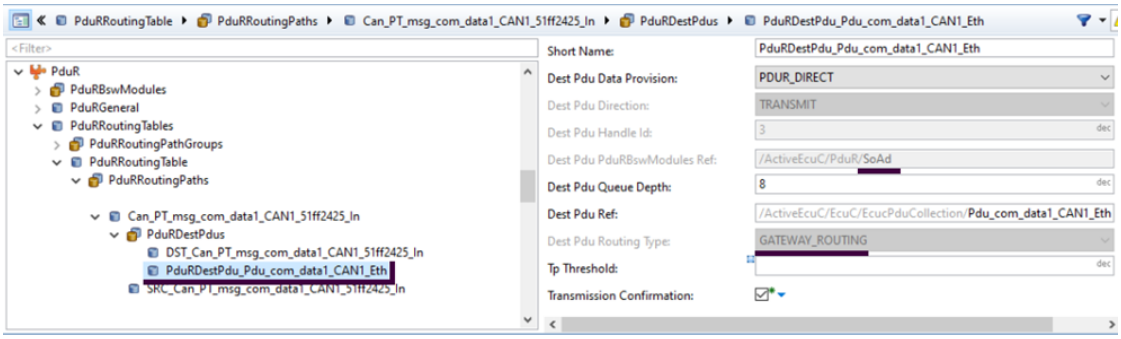
Слика 4.4. Контејнер генерисан у CanIf модулу

Модул који следећи преузима примљену *CAN* поруку је ***PduR***. У њему треба да се направе контејнери који описују пут усмеравања поруке, тј. из ког модула стиже порука и ком модулу се шаље. За нашу поруку је већ генерисан контејнер који описује њен пријем са *CanIf* модула.



Слика 4.5. Контејнер за пријем поруке генерисан у PduR модулу

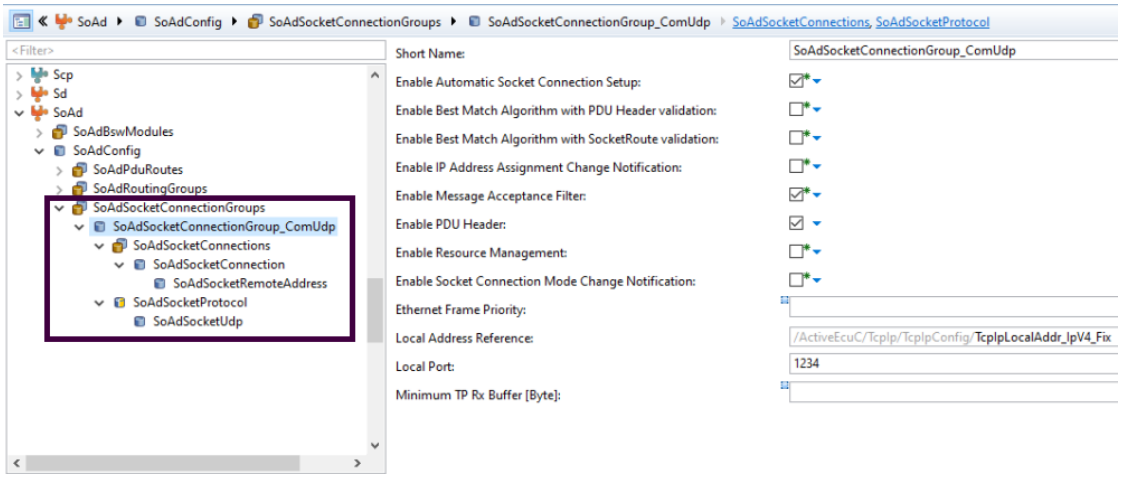
За слање на етернет комуникациони стек треба да направимо контејнер који ће описати слање примљене поруке на *SoAd* модул. У њему се бирају параметри одредишта поруке (*SoAd*), тип рутирања (*GATEWAY_ROUTING*).



Слика 4.6. Контејнер за усмеравање поруке у PduR модулу

Након пријема порука у ***SoAd*** модулу, пристигле поруке се мапирају на одабрани мрежни сокет. Пре тога се наравно треба направити контејнер који описује конфигурацију мрежног сокета и друге параметре везане за комуникацију. Као број порта стављамо 1234 и као *IP* адресу користимо 192.168.1.20. Дефинишемо да ћемо користити *UDP* тип протокола и параметре везане само за *UDP*. За тип адресе бирамо параметар *TCPIP_UNICAST* и *TCPIP_AF_INET*. Такође дефинишемо референцу на контролер ***EthIf*** на који ће порука бити усмерена и који ће обавити њено спуштање на физички слој етернета.

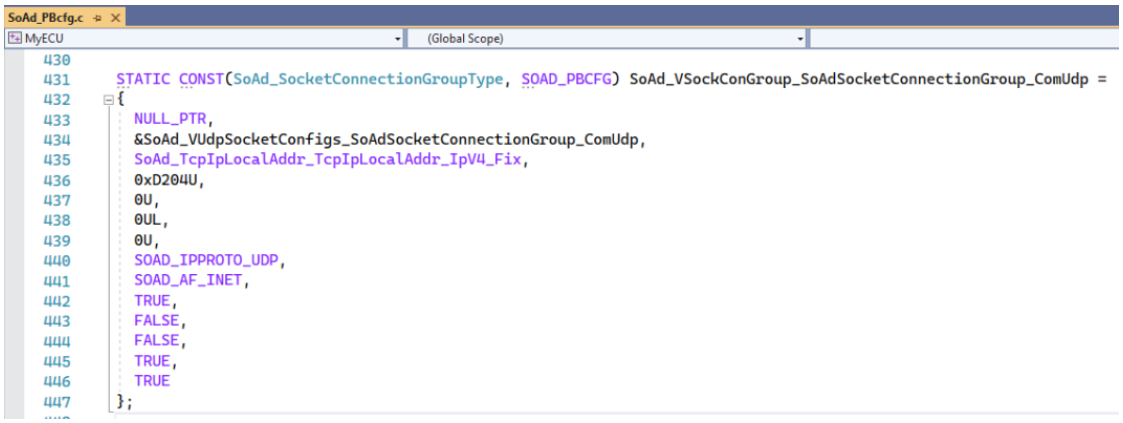
Постоји више начина да се намести окидач (*енг. trigger*) који ће одређивати на који начин ће да се сигнализира слање поруке на етернет. Овде смо одабрали да се порука шаље периодично (на сваких 1s). Дакле, све поруке које стижу са *PduR* се сакупљају и када истекне време од 1s оне се смештају у један етернет оквир и шаљу.



Слика 4.7. Контејнер у SoAd модулу

Када се заврши дефинисање конфигурације свих потребних модула, потребно је верификовати и генерисати програмско решење (*F9* или дугме *Generate*). Верификација проверава да ли су добро дефинисани сви потребни параметри. Потребно је одабрати да се генерише програмско решење за виртуално окружење, а постоји опција и за право окружење. Као производ генерисања на основу наше конфигурације добијају се .c и .h датотеке; у *Appl* датотеци налазе се датотеке везане за *SWC*-ове, а на путањи *Appl/Vtt/GetDataVtt/* налазе се датотеке везане за RTE и BSW, за виртуално окружење.

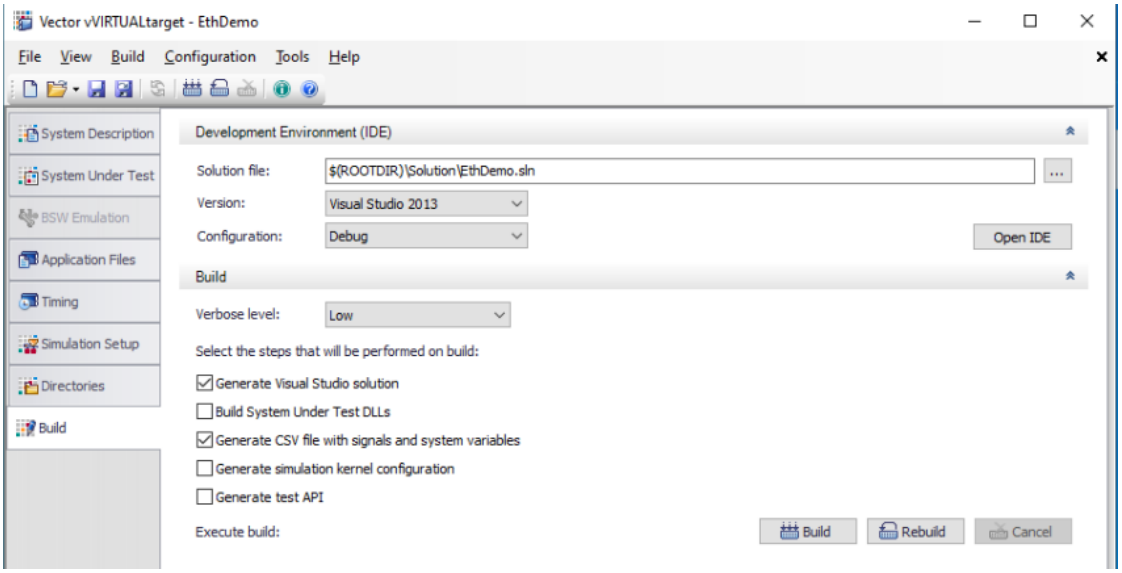
Код који генерише *DaVinci Configurator* је гломазан, али такође и организован ради лакшег сналажења. Можемо да видимо пример (слика 4.8.) генерисаног кода на основу контејнера `SoAd_VsockConGroup_SoAdSocketConnectionGroup_ComUdp` који смо направи у *SoAd*. За потребе овог рада не морамо да правимо никакве ручне измене у коду, идеја је да све буде покривено конфигурацијом.



Слика 4.8. Пример из генерисаног кода

4.2 Развој виртуалног окружења

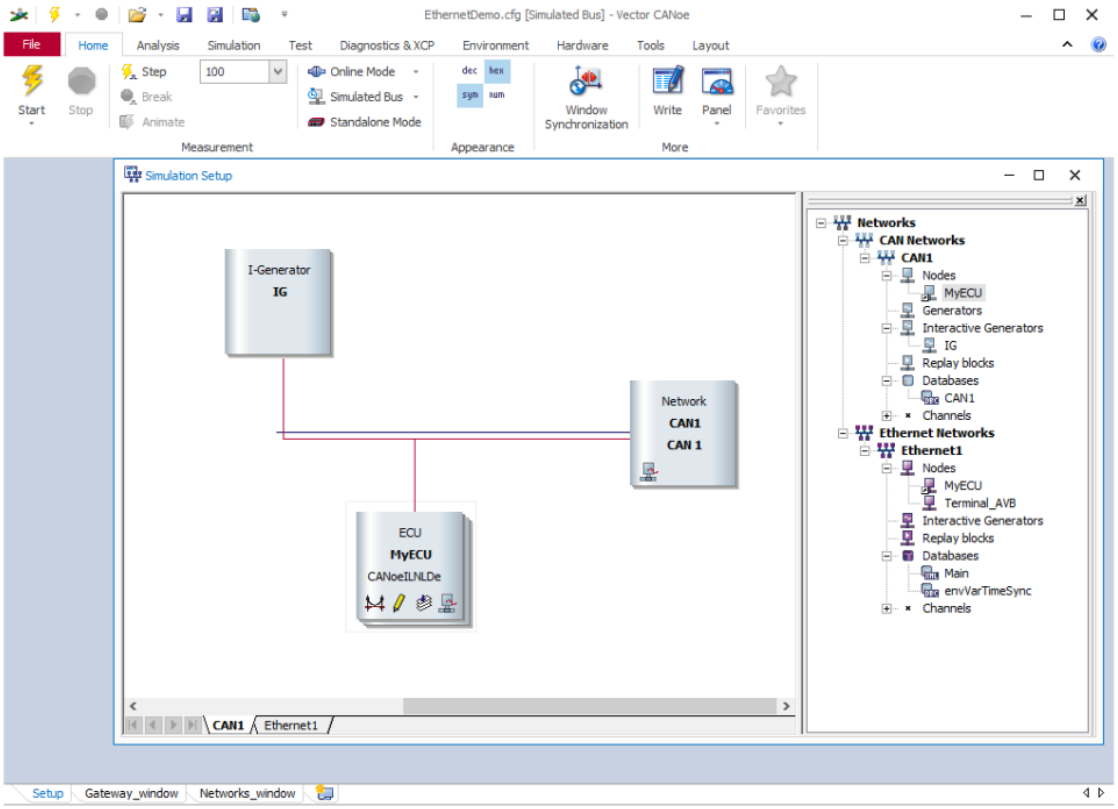
На путањи *Config/VTT/* налази се ***EthDemo.vttproj*** датотека генерисана из *DaVinci Configurator*-а. Она се отвара у *vVIRTUALtarget* окружењу. Већ смо рекли да нам је овај алат потребан за стварање виртуалног *ECU*-а, односно .dll датотеке која нам је потребна.



Слика 4.9. Окружење vVIRTUALtarget алата

Датотеку ***MyECU.dll*** можемо да добијемо на два начина; први начин је директно из *vVIRTUALtarget*-а у датотеци *SUT*; други начин је генерисање из ***EthDemo.sln*** *Visual Studio Solution* пројекта, који се такође ствара из *vVIRTUALtarget*-а и повезује све датотеке програмског решења (путања *Solution/Debug_vc120/*).

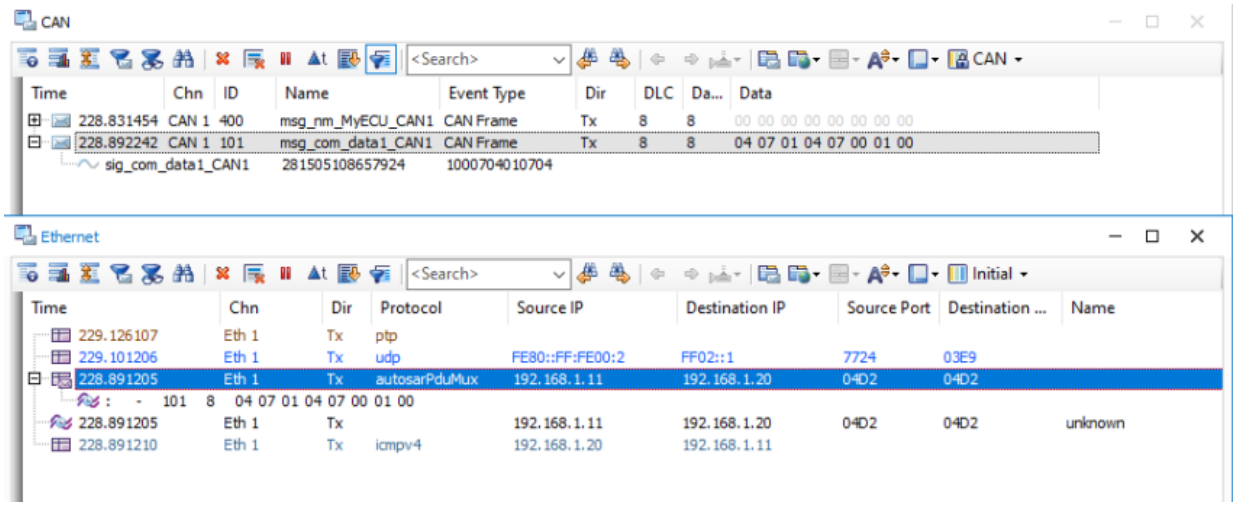
Следи коришћење *Vector CANoe* алата за имплементирање тестног окружења. Конфигурациона датотека ***EthernetDemo.cfg*** и све остало везано за овај алат се налази у *CANoe* датотеци. У окружењу додајемо две мреже, *CAN1* и *Ethernet1*. Додајемо виртуални *ECU* који је видљив на обе мреже и користи наше *MyECU.dll* решење. На *CAN* мрежи стављамо један интерактивни генератор порука који ће нам служити за симулирање порука на мрежи. Можемо да додамо и *Trace Window* који служи за слушање порука на магистралама. Симулација у реалном времену виртуалне комуникационе мреже се покреће са *Start* дугметом. Помоћу виртуалног окружења којег смо направили можемо да тестирамо и пратимо понашање нашег *ECU*-а.



Слика 4.10. Окружење *Vector CANoe* алата

5. Резултати

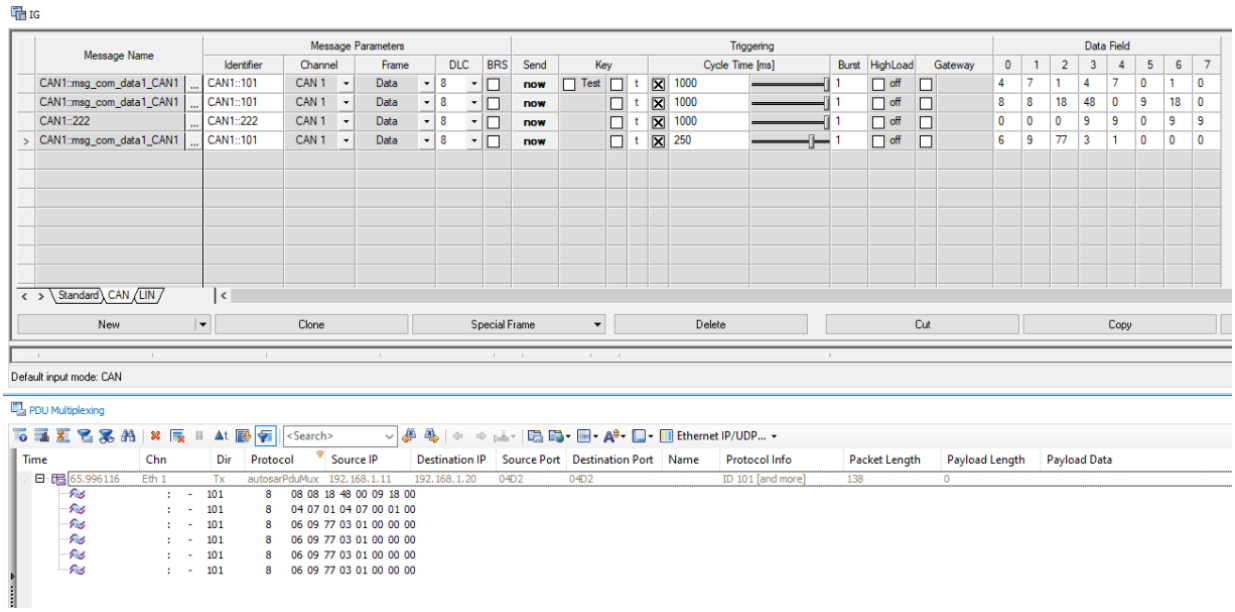
Као проверу исправности нашег програмског решења можемо да посматрамо симулацију коју смо направили у *CANoe* окружењу. Очекивани резултат су етернет поруке на мрежи које у себи садрше податке који су претходно послати у виду *CAN* порука. Рад електронске контролне јединице треба да буде поуздан.



Слика 5.1. Стање на мрежи током симулације

У првом прозору (слика 5.1.) за праћење порука на *CAN* мрежи можемо да видимо нашу *CAN* поруку која се периодично шаље, као и сигнал који се налази у поруци. У другом прозору за праћење порука на етернет мрежи видимо етернет поруку која у себи носи сигнал који је претходно био послат у *CAN* поруци. Можемо да видимо и параметре које смо задали за комуникацију као што су *IP* адреса 192.168.1.20 и број мрежног порта 1234 (*hex. 0x04D2*).

Интерактивни генератор нам омогућава да у реалном времену мењамо количину порука на *CAN* магистрали, брзину којом се оне шаљу, вредности сигнала у поруци итд. Овако можемо додатно да тестирамо понашање нашег решења.



Слика 5.2. Тестирање мењањем порука на CAN мрежи

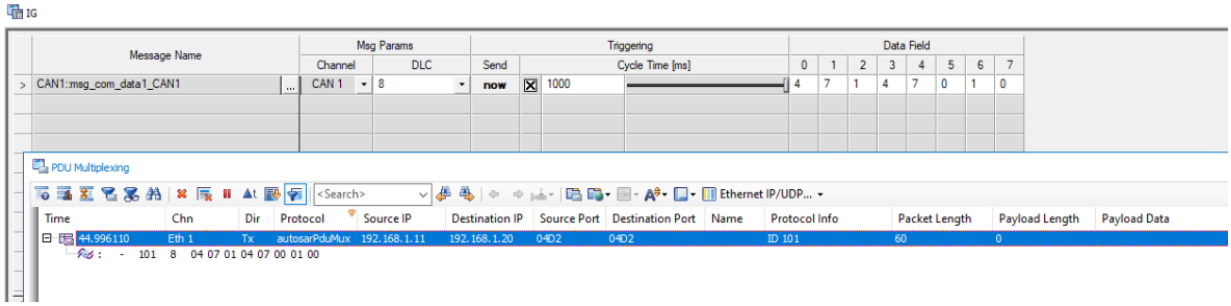
Прва *CAN* порука се периодично шаље на сваких 1s и у себи има сигнал са датим вредностима. Видемо да се њен сигнал појављује у етернет поруци, што се и очекује.

Друга *CAN* порука је иста као и прва, с тиме да су њене вредности сигнала промењене. Ове вредности се такође налазе у етернет поруци.

Трећа *CAN* порука има измењен идентификатор и вредности сигнала. Њене вредности се не налазе у етернет поруци. Разлог томе је што нашем *ECU*-у нисмо задали да прима поруке са овим идентификатором, тј. нисмо направили контејнер који описује пријем поруке са идентификатором 222 у *CanIf* модулу.

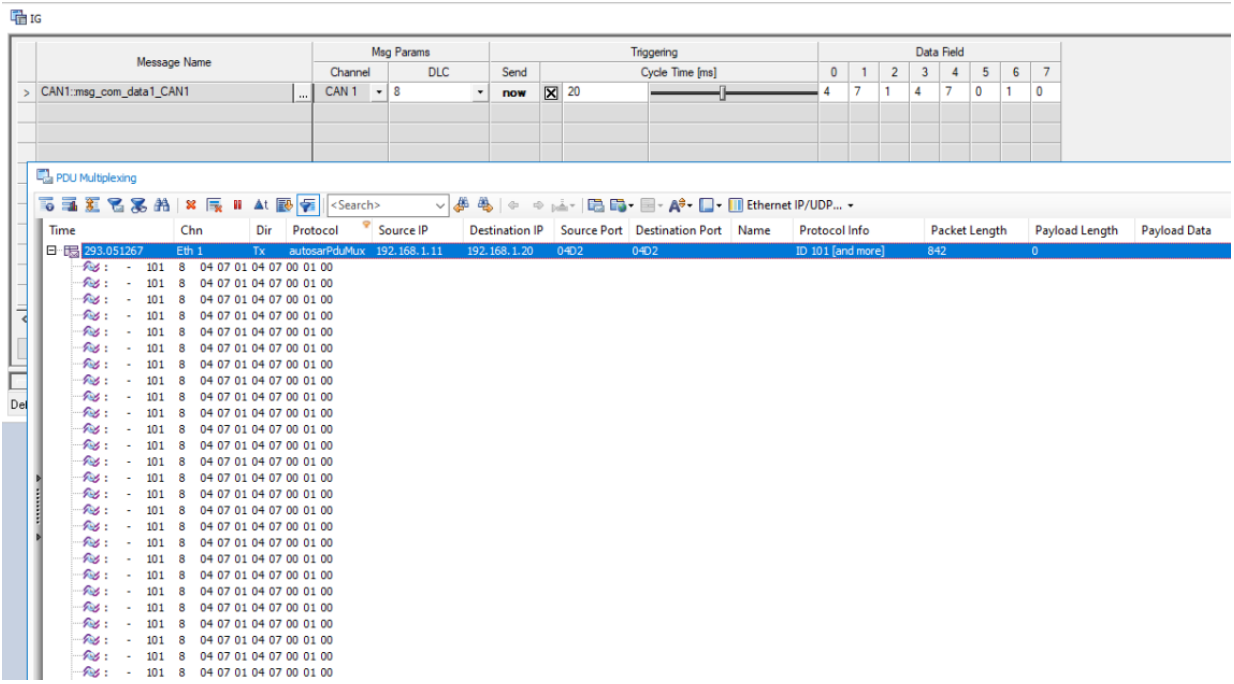
Четврта порука се периодично шаље сваких 250ms, дакле у једној секунди пристигну четири исте поруке. Вредности из све четири поруке се смештају у етернет поруку, што и видимо на слици 5.2.

Етернет порука се не шаље ако нема податак да се пошаље. Минимум мора да постоји једна *CAN* порука сваке секунде како би *ECU* имао шта да пошаље. Минимална дужина пакета је 60 бајтова.



Слика 5.3. Минимална дужина етернет поруке

Можемо да пронађемо максималну дужину пакета ако повећамо периодичност слања. Сигнали за које нема места да се пошаљу у тренутној етернет поруци се не губе, већ чекају следеће слање. Максимална дужина пакета је у нашем случају 842 бајта. *CAN* порука се шаље на сваких 20ms, што је ≈50 порука које стигну да се пошаљу.



Слика 5.4. Максимална дужина етернет поруке

6. Закључак

Идеја да се етернет користи је перспективна иновација која је већ пронашла своје место у аутомобилској индустрији, али то не утиче на *CAN* протокол који је имао деценије да се утемељи у индустрији као најкоришћенији протокол. Ова два протокола заједно могу да раде у истом комуникационом систему и да обављају задатке намењене да искористе њихове врлине. Ово се постиже добром организацијом система, што је у неку руку разлог за увођење зоналних контролера у новије пројекте.

Овај дипломски рад је за циљ имао да имплементира једно решење усмеравања у оквиру зоналног контролера, што је и урађено. За израду је коришћено знање о *CAN* и етернет протоколима, као и *AUTOSAR* архитектура која се показала као корисно решење за прављење програмске подршке електронских контролних јединица. Упознали смо се и са *Vector* породицом алатима, која олакшава цео процес прављења програмске подршке електронских контролних јединица, од дизајнирања до тестирања. Решење је успешно имплементирано и као резултат можемо да користимо конвертор протокола за преношење порука са *CAN* магистрале на етернет.

7. Литература

[1] И. Башичебић, М. Поповић, В. Ковачевић, *Основи рачунарских мрежа 1*, FTN.

[2] Nicolas Navet, Françoise Simonot-Lion. *In-vehicle communication networks - a historical perspective and review*, Richard Zurawski, Industrial Communication Technology Handbook, Second Edition, CRC Press Taylor&Francis, 2013. https://inria.hal.science/hal-00876524/document

[3] Wikipedia Contributors, *CAN bus*, Wikipedia, 2019. https://en.wikipedia.org/wiki/CAN_bus

[4] *The Evolution of In-vehicle Network Architectures*. https://blog.teledynelecroy.com/2023/02/the-evolution-of-in-vehicle-network.html

[5] S. Corrigan, *Introduction to the Controller Area Network (CAN) Application Report Introduction to the Controller Area Network (CAN),* 2002. https://www.ti.com/lit/an/sloa101b/sloa101b.pdf?ts=1720044341018

[6] *Automotive Ethernet: An Overview*, 2014. https://support.ixiacom.com/sites/default/files/resources/whitepaper/ixia-automotive-ethernet-primer-whitepaper_1.pdf

[7] John Sprovieri, *Processing Automotive Ethernet Cables*, ASSEMBLY, 2022. https://www.assemblymag.com/articles/96832-processing-automotive-ethernet-cables

[8] *Automotive Ethernet,* Vector E-Learning, 2018. https://elearning.vector.com/mod/page/view.php?id=149

[9] Konrad Etschberger, *Comparing CAN- and Ethernet-based Communication.* https://copperhilltech.com/content/comparison_can_and_ethernet.pdf

[10] *Layered Software Architecture AUTOSAR Classic Platform*.
https://www.autosar.org/fileadmin/standards/R22-11/CP/AUTOSAR_EXP_LayeredSoftwareArchitecture.pdf
[11] *Classic Platform AUTOSAR*, Autosar.org, 2023.
https://www.autosar.org/standards/classic-platform
[12] *AUTOSAR-COM STACK*, YouTube.
https://www.youtube.com/live/pPAw0DCaTiA?si=elzp-_DA74jlsG_O
[13] *Overview on Ethernet Communication Stack*, YouTube.
https://www.youtube.com/watch?v=8P1xruWOoBo
[14] *DaVinci Developer Classic*, Vector Informatik GmbH.
https://www.vector.com/int/en/products/products-a-z/software/davinci-developer-classic/#c10366
[15] *DaVinci Configurator Classic*, Vector Informatik GmbH.
https://www.vector.com/int/en/products/products-a-z/software/davinci-configurator-classic