

Universitatea “Politehnica” din Timișoara  
Facultatea de Automatică și Calculatoare  
Specializarea Ingineria Sistemelor

# **LUCRARE DE LICENȚĂ**

**-iunie 2013-**

**Analiza și implementarea standardului AUTOSAR în controlul  
diferențialului dintr-un automobil**

**Coordonatori proiect:**



**Ș. L. Dr. Ing. Sorin Nanu**



**Ing. Bogdan Atănăsoaiei**

**Candidat:**

**Paula Vasile**

## Cuprins

1. Introducere .....	5
1.1. Context.....	5
1.2. Situația actuală.....	5
1.3. Problema de rezolvat.....	5
2. Documentare .....	6
2.1. Prezentare sisteme de transmisie a puterii .....	6
2.2. Prezentarea standardului AUTOSAR .....	10
2.3. Prezentarea de ansamblu a sistemului.....	16
2.3.1. Sistemul mecanic .....	16
2.3.2. Sistemul hardware.....	17
2.3.3. Sistemul software.....	20
3. Realizarea proiectului în ansamblu .....	24
3.1. Prezentarea temei pe larg .....	24
3.1.1. Prezentarea proiectului existent - QHCU.....	24
3.1.2. Prezentarea proiectului nou - QCU .....	25
3.1.3. Problema de rezolvat.....	27
3.2. Prezentare ciclu de dezvoltare V.....	29
3.3. Planificarea activității .....	30
4. Elaborarea cerințelor proiectului.....	33
4.1. Cererile clientului.....	34
4.2. Cerințe AUTOSAR.....	37
4.3. Analiza cerințelor în vederea alegerii variantei optime de implementare.....	39
4.3.1. Varianta 1 - Proiect complet standardizat AUTOSAR .....	40
4.3.2. Varianta 2 - renunțarea completă la standardul AUTOSAR.....	41

4.3.3.	Varianta 3 - proiect parțial standardizat AUTOSAR .....	42
4.4.	Cerințele proiectului.....	43
5.	Proiectarea software-ului pe baza cerințelor .....	46
5.1.	Design și arhitectură .....	46
5.1.1.	Tool-uri utilizate .....	46
5.1.2.	Structură.....	46
5.2.	Implementare .....	56
6.	Testarea software-ului.....	58
6.1.	Utilitare folosite .....	58
6.2.	Testarea propriu-zisă.....	59
7.	Concluzii .....	60
7.1.	Rezultatele proiectului .....	60
7.2.	Direcții de dezvoltare .....	60
7.3.	Utilitatea proiectului .....	61
8.	Bibliografie și studiu bibliografic .....	62
ANEXA 1	- Lista de figuri.....	63



## 1. Introducere

### 1.1. Context

Compania Audi dorește dezvoltarea unui nou software pentru comanda diferențialului de pe puntea din spate a autoturismului Audi Quattro Sport. Acest software va fi implementat în colaborare de către companiile Continental Automotive și Magna Powertrain. Produsul final va rula pe o unitate de control a transmisiei, denumită mai departe în lucrare **TCU (Transmission Control Unit)**.

### 1.2. Situația actuală

În cadrul companiei Continental s-a dezvoltat software pentru modelul anterior al autoturismului. Acest proiect vechi nu respectă standardul AUTOSAR, deci nu urmărește o anumită arhitectură sau implementare, ci este particularizat.

Proiectul anterior este foarte util în dezvoltarea celui nou, deoarece ambele proiecte îndeplinesc aceleași cerințe funcționale. Diferențele intervin în aspectele legate de structurarea proiectului, în interfetele utilizate și în modul de comunicare.

### 1.3. Problema de rezolvat

În cadrul proiectului nou se dorește standardizarea software-ului pentru cât mai multe module. AUTOSAR este un standard pentru arhitecturile sistemelor din domeniul automotive care furnizează o infrastructură de bază în dezvoltarea de software pentru autovehicule.

Tema lucrării de licență o reprezintă implementarea unui software standardizat AUTOSAR pentru unitatea de control a diferențialului dintr-un autoturism. Aceasta implementare se va baza pe cererile clientului, pe cerințele standardului care se dorește a fi respectat, dar și pe implementările funcționalităților realizate în proiectul existent.

Codul unei aplicații standardizate AUTOSAR poate fi implementat în mai multe moduri, el poate fi generat automat și configurat ulterior manual sau poate fi implementat în integralitate manual. Modalitatea de implementare este determinată de mai mulți factori de natură tehnică sau nu. Analiza asupra acestor modalități de elaborare a software-ului reprezintă, de asemenea, o parte importantă a acestei lucrări, iar ea este prezentată pe larg într-un capitol ulterior, încheindu-se cu alegerea variantei optime de implementare.

## 2. Documentare

### 2.1. Prezentare sisteme de transmisie a puterii

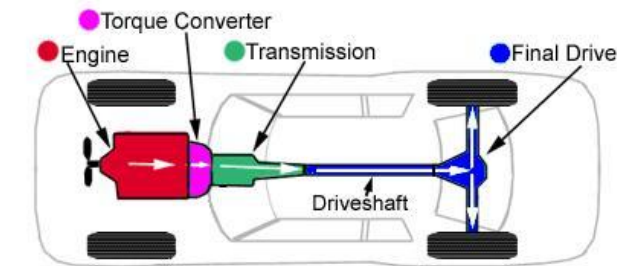
[4] “Necesitatea unui sistem de transmisie într-un automobil apare ca o consecință a caracteristicilor motorului cu combustie internă. Aceste motoare operează între 600 și 7000 de rotații pe minut, în timp ce roțile unei mașini au o viteză între 0 și 1800 de rotații pe minut.

Orice mașină constă dintr-o sursă de putere și un sistem de transmisie a puterii, care asigură aplicarea controlată a acesteia. De cele mai multe ori, termenul “transmisie” se referă la cutia de viteze care se folosește de niște angrenaje pentru a furniza roților valori convertite ale vitezei și cuplului provenite de la motor. În continuare este descris mai detaliat modul în care puterea produsă de motor este transmisă roților sub formă de mișcare de rotație.”

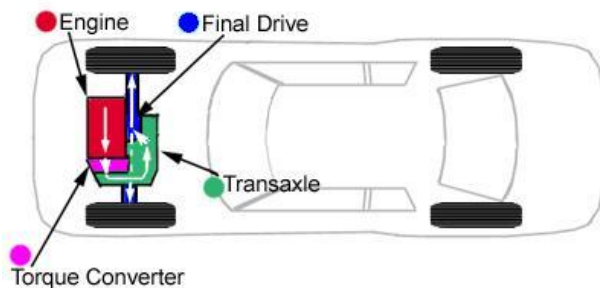
#### Sistemul de tracțiune

[4] “Sistemul de tracțiune al unei mașini este compus din toate elementele ce contribuie la transformarea energiei produse de motor în energie cinetică, pentru a fi posibilă deplasarea unui automobil.”

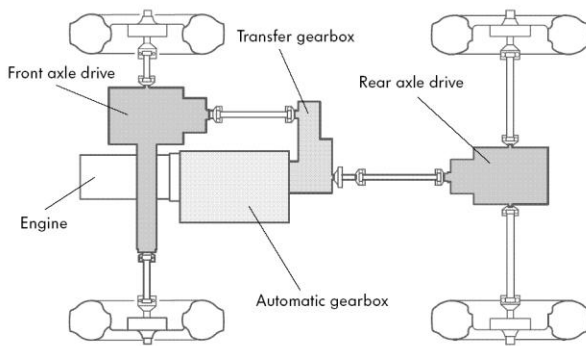
Tracțiunea este de mai multe categorii, în funcție de tipul mașinii:



Tracțiune spate: întâlnită, în general, la mașinile sport (BMW, Ferrari, Lamborghini etc)



Tracțiune față: se întâlnește la majoritatea mașinilor mici, cele care pun în mișcare mașina fiind roțile din față (ex. Renault, Ford, VW, Opel etc.)



Tracțiune integrală: întâlnită la mașinile de teren, 4x4 (BMW x5, VW Touareg, Audi Q7)

**Fig. 1** Tipuri de tracțiune

## Elementele unui sistem de transmisie a puterii

### a. Ambreiajul

Ambreiajul are rolul de a împiedica oprirea motorului în cazul în care se opreste automobilul prin decuplarea manuală a motorului de cutia de viteze. De asemenea, contribuie la pornirea lină a autovehiculului de pe loc.

### b. Cutia de viteze

Cutia de viteze este un ansamblu de roți dințate care servește la transformarea forței și transmiterea mișcării de rotație la roțile autovehiculului.

**Raportul de transmisie** este raportul dintre viteza de rotație a roților dințate ce compun cutia de viteze și viteza de rotație a roților automobilului.

Cutiile de viteze se împart în mai multe categorii, în funcție de:

1. Raportul de transmisie:
  - cu raport de transmisie fix
  - cu raport de transmisie variabil (CVT)
2. Gradul de automatizare:
  - manuală
  - CVT (Continuously Variable Transmission)
  - automată

În cazul **cutiei de viteze manuale**, pentru fiecare viteză a cutiei este cuplată o roată dințată a acesteia pentru care există un raport de transmisie fix, ca în imaginea de mai jos:

Viteză	Raport
1	3.455
2	1.944
3	1.286
4	0.939
5	0.745
R	3.167

**Cutiile de viteze CVT (Continuously Variable Transmission)**, spre deosebire de cutiile manuale, au capacitatea de a parcurge un număr infinit de rapoarte de transmisie. CVT oferă eficiență sporită în ceea ce privește consumul de carburant, deoarece îi permite motorului să funcționeze la cea mai eficientă turație.

**Cutiile de viteze automate** sunt cutiile care realizează schimbarea treptelor de viteză fără intervenția conducătorului automobilului. Mai mult, decizia de schimbare a treptelor de viteză este luată de asemenea automat, pe baza informațiilor provenite de la diferiți senzori. Realizarea unei trepte de viteză se face prin intermediul mai multor mecanisme planetare.

**Cutiile de viteze automate cu dublu ambreiaj**, din punct de vedere cinematic, sunt de fapt compuse din două cutii de viteze manuale, dispuse în paralel. Practic, în aceeași carcasă avem două cutii de viteze, fiecare cu propriul ambreiaj, o cutie conținând treptele impare (1, 3, etc.) iar a doua treptele pare (2, 4, etc.). Acest principiu permite ca vitezele să fie schimbate fără să se piardă din putere. În timp ce unul din ambreiaje transmite puterea, celălalt este pregătit pentru a cupla următoarea viteză, care este preselectată. Astfel, schimbul vitezei se realizează într-o fracțiune de secundă.

### c. Diferențialul

[4] “Diferențialul este un dispozitiv mecanic care permite roților automobilului să se deplaseze cu viteze diferite în timp ce puterea le este furnizată de aceeași sursă. De asemenea, diferențialul are rolul de a distribui cuplul provenit de la motor între cele două roți de pe punte.”



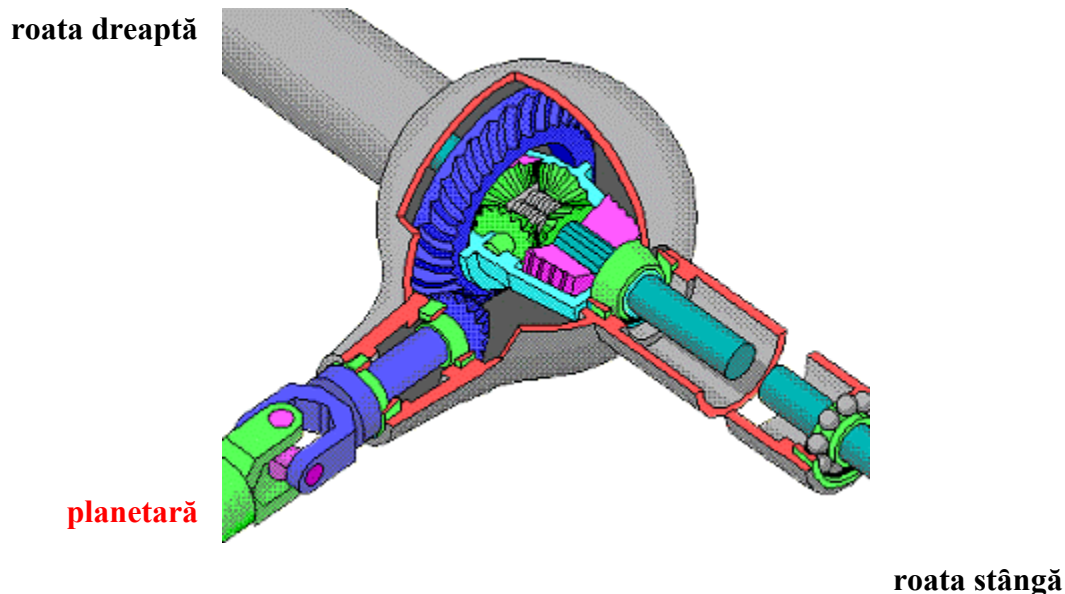
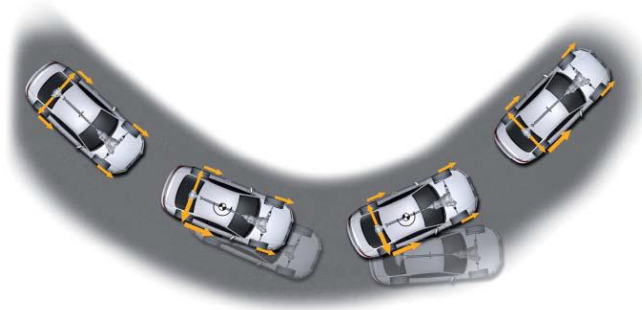


Fig. 2 Diferențialul

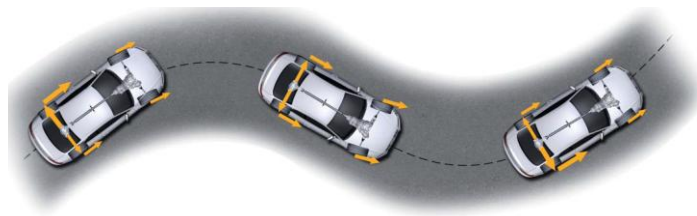
Tipuri de diferențiale:

- Diferențialul deschis - cel mai des întâlnit la automobilele de serie
- Diferențialul sensibil la viteză
- Diferențialul sensibil la cuplu sau diferențialul de tip Torsen

În timpul rulării, roata ce se află pe interiorul unei curbe pierde din tracțiune și, o dată cu aceasta, și abilitatea de a furniza cuplu, cauzând și roata opusă, aflată în exterior să piardă, de asemenea, din tracțiune, automobilul nemaifiind în stare să vireze. Diferențialul sport, spre deosebire de alte diferențiale, pe lângă faptul că transferă cuplu roții din exteriorul curbei, suprapune acestuia și un cuplu suplimentar prin intermediul unui ambreiaj, astfel încât să fie eliminat cazul subvirajului în curbe și pierderea stabilității în cazul parcurgerii mai multor viraje succesive.



**Fig. 3** Eliminarea subvirajului cu ajutorul diferențialului sport



**Fig. 4** Menținerea stabilității în curbe succesive cu ajutorul diferențialului sport

## 2.2. Prezentarea standardului AUTOSAR

[4]”AUTOSAR (AUTomotive Open System ARchitecture) este un parteneriat între mai mulți producători de automobile și componente care conlucrează pentru a dezvolta un standard pentru industria automotive. Obiectivul acestora este să creeze un standard pentru arhitecturile electrice/electronice din domeniul automotive care să furnizeze o infrastructură de bază în dezvoltarea de software pentru autovehicule. Aceasta presupune, printre altele, standardizarea funcțiilor de bază ale sistemului, scalabilitatea la diferite autovehicule și platforme, posibilitatea integrării de componente provenite de la diferiți furnizori, asigurarea mentenanței de-a lungul întregului ciclu de viață al produsului.

Primele discuții cu privire la un astfel de standard s-au purtat în luna august a anului 2002 între companiile BMW, Bosch, Continental, DaimlerChrysler și Volkswagen cărora li s-a alăturat în curând și Siemens VDO. Parteneriatul a fost semnat în mod formal în noiembrie 2003. Până în prezent, acestora li s-au alăturat Peugeot Citroën Automobiles S.A. și Toyota Motor Corporation în decembrie 2003 și General Motors în noiembrie 2004.”

[3] „Este nevoie de un astfel de standard din mai multe motive, cum ar fi:

- creșterea complexității software-ului în general precum și creșterea numărului funcționalităților cerute pe piață
- necesitatea unei mai mari flexibilități în cazul modificării, îmbunătățirii, înnoirii software-ului
- necesitatea de a crește gradul de calitate a produselor și de a spori încrederea în sistem

Standardul AUTOSAR optează pentru o arhitectură ce suportă un design bazat pe componente și încearcă să reorienteze modul de abordare al proiectelor din domeniu de la implementarea bazată pe ECU la cea bazată pe funcționalități. Scopul final este acela de a crea componente disponibile “la raft”, care pot fi achiziționate și pot rula fără modificări sau adaptări suplimentare pe platforme diferite, datorită interfețelor de asemenea standardizate.

Standardul AUTOSAR prezintă următoarele caracteristici cheie:

- Modularitate și configurabilitate
  - o definirea unei arhitecturi modulare pentru unitățile de control electronice, denumite în continuare **ECU (Electronic Control Unit)**
  - o dezvoltarea deopotrivă de module dependente și independente de structura hardware a ECU-ului
  - o capacitatea de integrare a modulelor software provenite de la diferiți furnizori pentru a mări capacitatea de reutilizare a funcționalităților
  - o optimizarea infrastructurii software a fiecărui ECU din punctul de vedere al utilizării resurselor
- Interfețe standardizate
  - o standardizarea diferitelor API-uri pentru a separa straturile software AUTOSAR
  - o facilitarea încapsulării componentelor software funcționale
  - o definirea standardizată a tipurilor de date pentru componentele software
  - o standardizarea interfețelor modulelor BSW
- Runtime Environment (RTE)
  - o furnizarea de comunicație între și în cadrul ECU-urilor prin toate nodurile rețelei unui autovehicul
  - o localizat între componentele FSW-ului și BSW-ului
  - o toate entitățile conectate la RTE trebuie să se conformeze specificațiilor AUTOSAR
  - o face posibilă integrarea cu ușurință a componentelor FSW specifice clientului

Pentru a îndeplini scopurile propuse: **modularitatea**, **scalabilitatea**, **transferabilitatea** și **reutilizabilitatea** funcționalităților, AUTOSAR furnizează o infrastructură software comună pentru sistemele software automotive care se bazează pe interfețe standardizate între diferitele straturi. AUTOSAR face posibilă optimizarea procesului de configurare și, dacă este necesar, permite optimizarea locală în vederea îndeplinirii cerințelor privind timpul de execuție și a constrângerilor hardware.

**Modularitatea** se referă la posibilitatea de a adapta software-ul la cerințele individuale ale ECU-urilor și a task-urilor acestora.

**Scalabilitatea** funcționalităților se referă la posibilitatea de adaptare a modulelor software comune la diferite platforme, pentru a evita dezvoltarea mai multor variante de software care să îndeplinească, în final, aceleași funcționalități.

**Transferabilitatea** funcționalităților optimizează utilizarea resurselor disponibile în cadrul arhitecturii electronice a unui vehicul.

**Re-utilizabilitatea** funcționalităților ajută la sporirea calității și încrederii, precum și la întărirea imaginii companiei în cadrul liniei de produse pe care o dezvoltă.

Cheia în îndeplinirea scopurilor menționate mai sus este prezentă **intefețelor standardizate**, atât între furnizori și producători, cât și între diferitele straturi software ale arhitecturii.”

Mai jos este prezentată o privire de ansamblu asupra arhitecturii AUTOSAR:

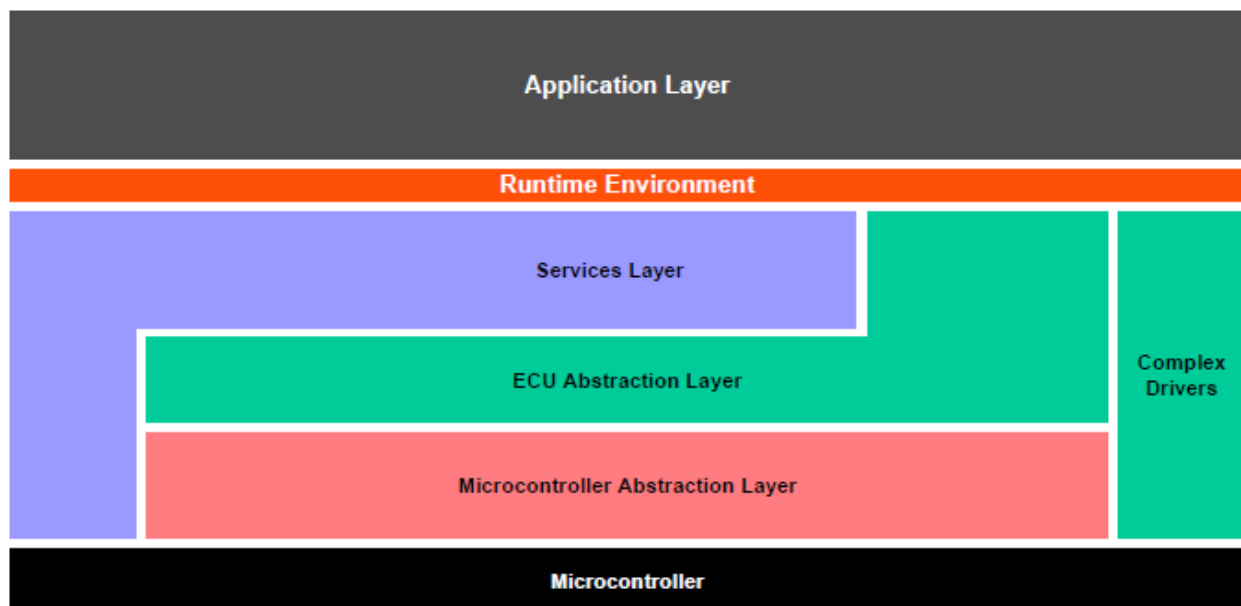
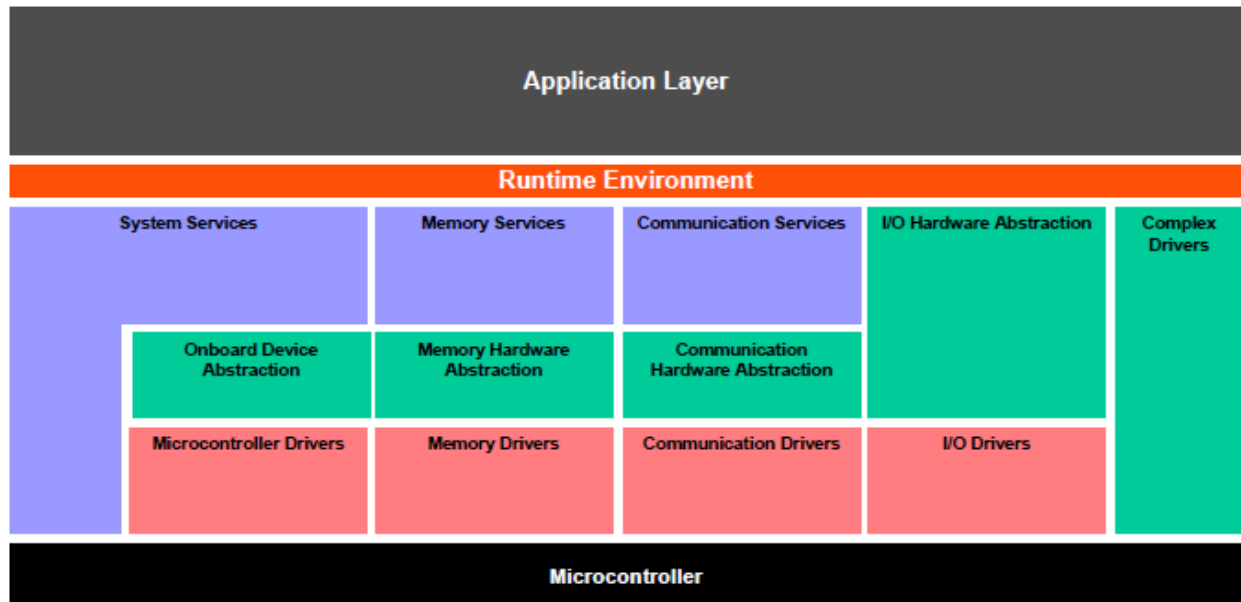


Fig. 5 Privire de ansamblu asupra arhitecturii AUTOSAR

[3] “Standardul AUTOSAR presupune asigurarea independenței straturilor software superioare față de caracteristicile microcontroller-ului folosit. În acest sens, arhitectura este dezvoltată pe mai multe straturi, după cum urmează:

- **Stratul de abstractizare al microcontroller-ului (Microcontroller Abstraction Layer)**
  - este cel mai de jos nivel, conține driver-e interne care sunt, de fapt, module software cu acces direct la perifericele și dispozitivele externe ale microcontroller-ului
  - are rolul de a face straturile superioare lui independente de microcontroller
  - implementarea lui este dependentă de microcontroller-ul folosit, iar interfața superioară este standardizabilă și independentă de microcontroller
- **Stratul de abstractizare al ECU-ului (ECU Abstraction Layer)**
  - interfațează driver-ele stratului de abstractizare al microcontroller-ului și conține, de asemenea, driver-e pentru dispozitivele externe
  - oferă o interfață pentru accesul la periferice și dispozitive indiferent de localizarea lor (în interiorul/exteriorul microcontroller-ului) sau conexiunea lor la microcontroller (pini pe porturi sau alt fel de interfețe)
  - are rolul de a face straturile superioare independente de ECU
  - implementarea lui este independentă de microcontroller și dependentă de configurația hardware a ECU-ului
  - interfața superioară este independentă de microcontroller și ECU și dependentă de tipul semnalelor
- **Stratul de servicii (Service Layer)**
  - este cel mai înalt nivel al BSW-ului și, în timp ce stratul de abstractizare al ECU-ului oferă acces la semnalele de intrare/ieșire, stratul de servicii oferă: funcționalități de sistem de operare, servicii de comunicare și management a rețelei vehiculului, servicii de memorie (NVRAM), servicii de diagnoză, management-ul stărilor în care se află ECU-ul
  - are rolul de a furniza servicii de bază pentru modulele FSW-ului și altor module ale BSW-ului
  - implementarea este specifică parțial microcontroller-ului, configurației hardware a ECU-ului și FSW-ului
  - interfața superioară este independentă față de microcontroller și configurația hardware a ECU-ului”

Fiecare din aceste straturi este structurat pe mai multe module specializate în îndeplinirea unei anumite funcționalități:



**Fig. 6** Arhitectura AUTOSAR în detaliu

Exemplu de funcționare pentru achiziția unui semnal de către modulul RTE:

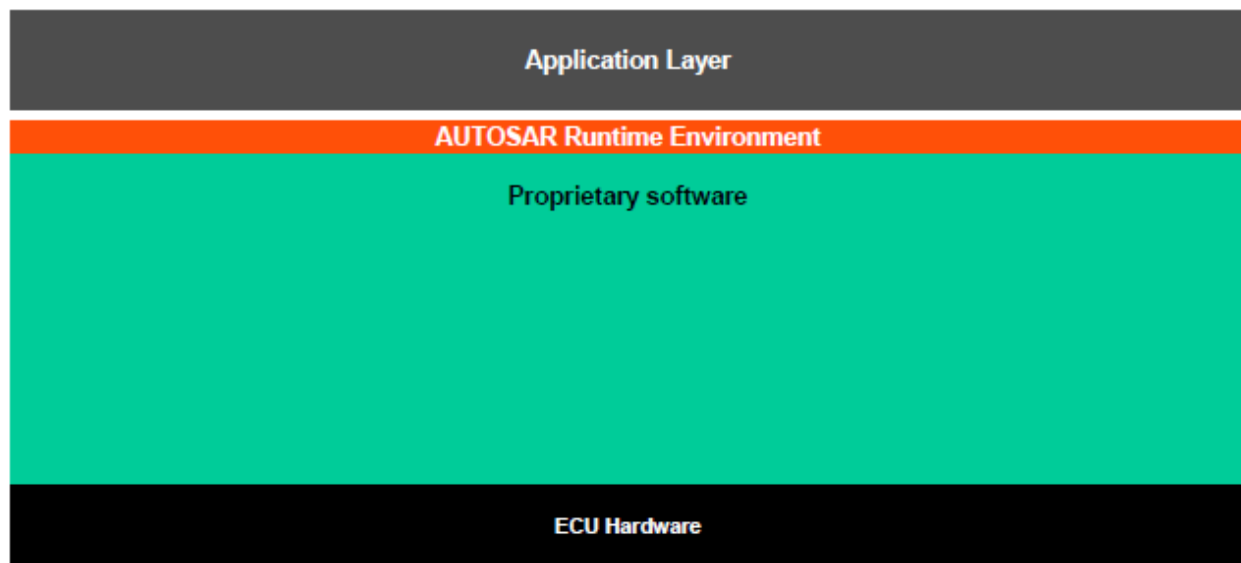
- RTE apelează funcția de achiziție din cadrul modulului I/O Hardware Abstraction cu parametrii corespunzători semnalului dorit a fi achiziționat;
- modulul I/O Hardware Abstraction apelează, la rândul lui, funcția corespunzătoare din modulul I/O Drivers; modulul I/O Hardware Abstraction nu deține cunoștințe referitoare la modalitatea de achiziție a semnalului;
- modulul I/O Drivers achiziționează, prin metode dependente de microcontroller-ul pe care rulează, semnalul dorit.

Avantajul major al acestei arhitecturi se observă în momentul în care apar modificări în structura microcontroller-ului. În acest caz, singurul modul care va suferi modificări va fi modulul I/O drivers, deoarece este singurul modul care depinde de configurația microcontroller-ului.

Deoarece se dorește actualizarea unor proiecte deja existente astfel încât acestea să îndeplinească cerințele AUTOSAR, s-au definit mai multe clase de conformitate, denumite în continuare **ICC (Implementation Conformance Class)**:

- **ICC3**: toate modulele BSW sunt conforme cu cerințele AUTOSAR, iar RTE-ul este văzut ca un modul separat și are rolul de interfață între BSW și FSW;
- **ICC2**: BSW-ul este împărțit în “clustere” - module care interfațează cu alte module conform cerințelor AUTOSAR, dar a căror implementare internă nu respectă normele AUTOSAR - iar RTE-ul este văzut ca făcând parte din BSW și își păstrează rolul de interfață între BSW și FSW;
- **ICC1**: întregul BSW este dedicat unui anumit ECU, iar RTE-ul este văzut ca făcând parte din BSW și își păstrează rolul de interfață între BSW și FSW;

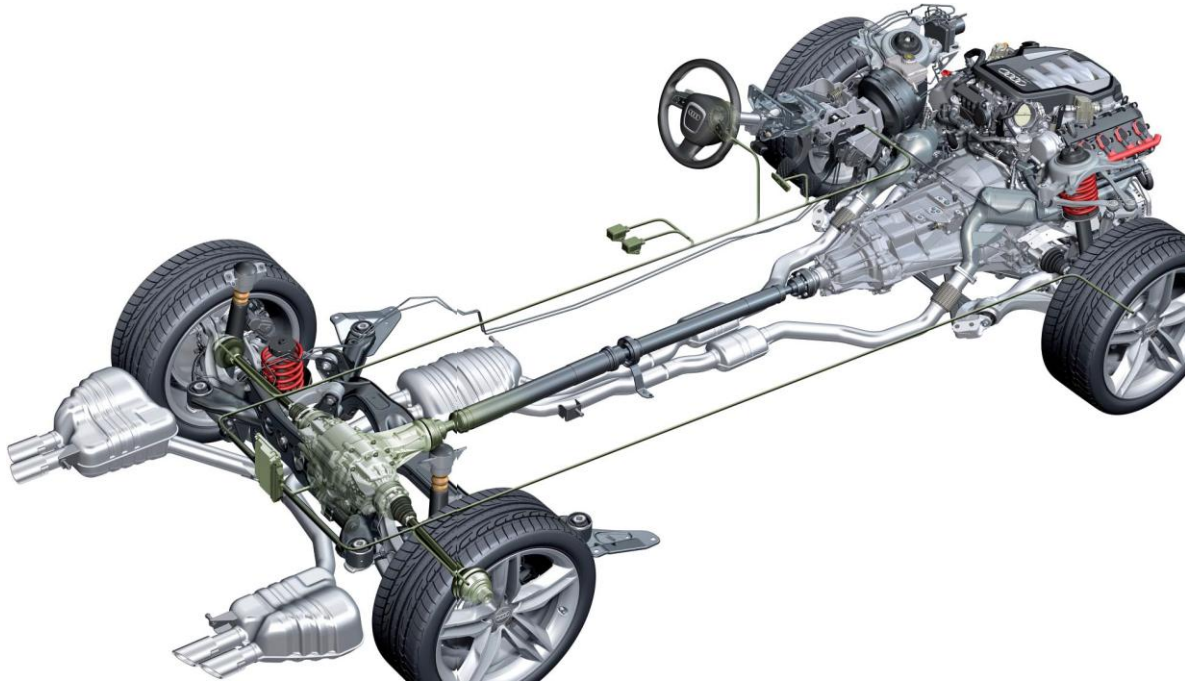
Proiectul de față se încadrează în categoria ICC1, deoarece BSW-ul este format atât din module care se conformează normelor AUTOSAR, cât și module care nu respectă cerințele standardului. Arhitectura unui proiect din clasa ICC1 este, în mare, cea din figură:



**Fig. 7** Arhitectura unui proiect din clasa de conformitate ICC1

## 2.3. Prezentarea de ansamblu a sistemului

### 2.3.1. Sistemul mecanic



**Fig. 8** Sistemul de transmisie al autoturismului Audi Quattro Sport

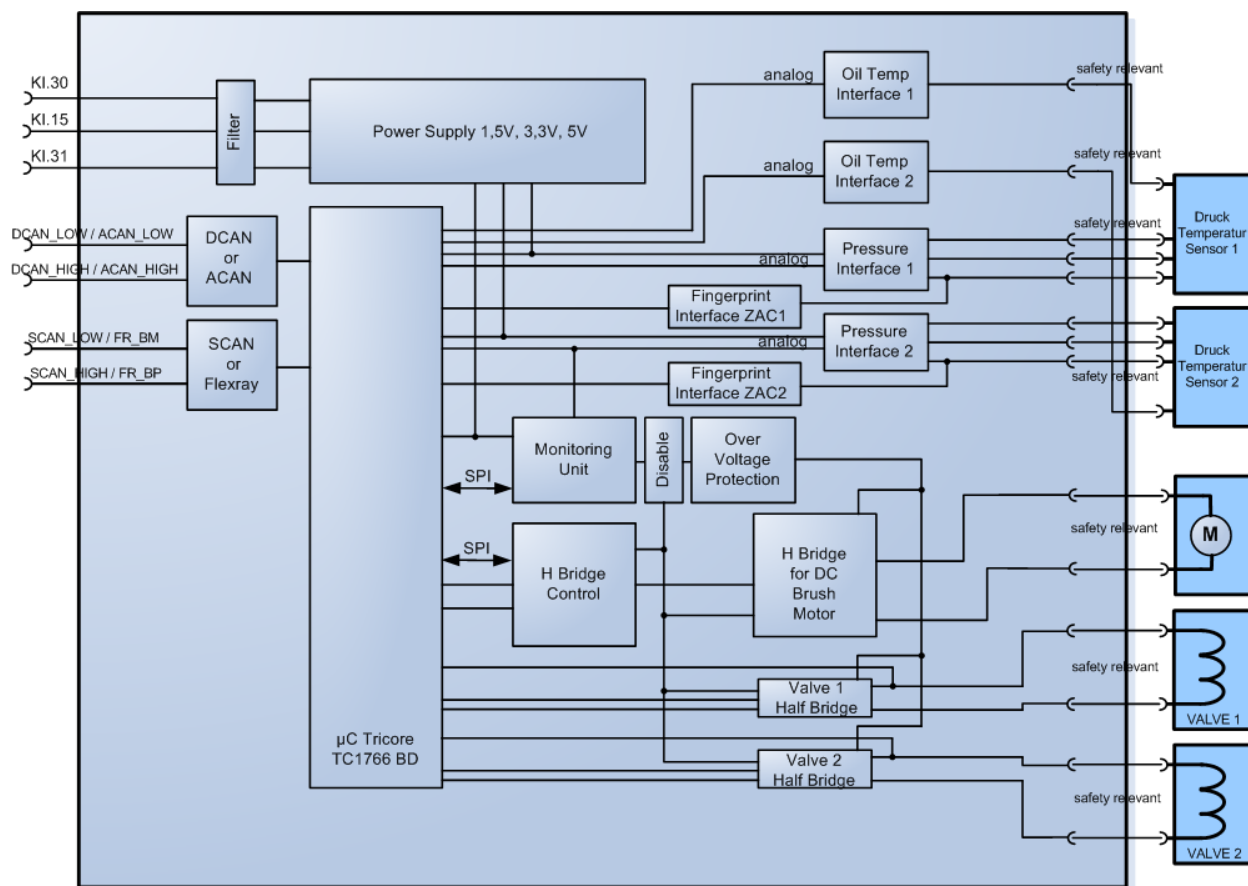
Diferențialul sport de tip Torque Vectoring aflat pe puntea din spate a modelului Audi Quattro Sport, este o nouă tehnologie implementată în domeniul diferențialelor pentru automobile. Tehnologia Torque Vectoring conferă diferențialului abilitatea de a varia cantitatea de putere/cuplu transmis fiecărei roți. Unitatea de control a diferențialului are de luat în considerare vitezele de rotație a fiecăreia din cele două roți, precum și viteza de deplasare a mașinii. Cum acești doi factori variază în timpul mersului, forțe diferite sunt exercitate asupra roților. Diferențialul monitorizează aceste forțe și ajustează cuplul în consecință. Această abilitate sporește capacitatea de a menține tracțiunea în condiții meteo nefavorabile. Când una din roți derapează, diferențialul reduce cuplul transmis acelei roți, efectiv aplicând o frânare a acesteia. Diferențialul va mări cuplul transmis roții opuse, ajutând la păstrarea stabilității mașinii. Diferențialul de tip Torque Vectoring aflat pe puntea din față a unui autoturism, pe lângă vitezele celor două roți, mai monitorizează și unghiul de bracaj al acestora.



### 2.3.2. Sistemul hardware

Din punct de vedere hardware, TCU-ul cuprinde, în principal, un microcontroller (TC1782F - Infineon) care se ocupă de toate prelucrările în timp real, software-ul fiind stocat în memoria flash.

Mai jos este prezentată schema bloc a proiectului QHCU:



**Fig. 9** Schema bloc a proiectului

În cazul proiectului QCU, singura diferență în sistem apare la comunicarea cu alte ECU-uri din rețeaua autovehiculului, aceasta realizându-se utilizând protocolul Flexray, spre deosebire de protocolul CAN, utilizat în proiectul anterior - QHCU.

Așadar, sistemul este alcătuit din:

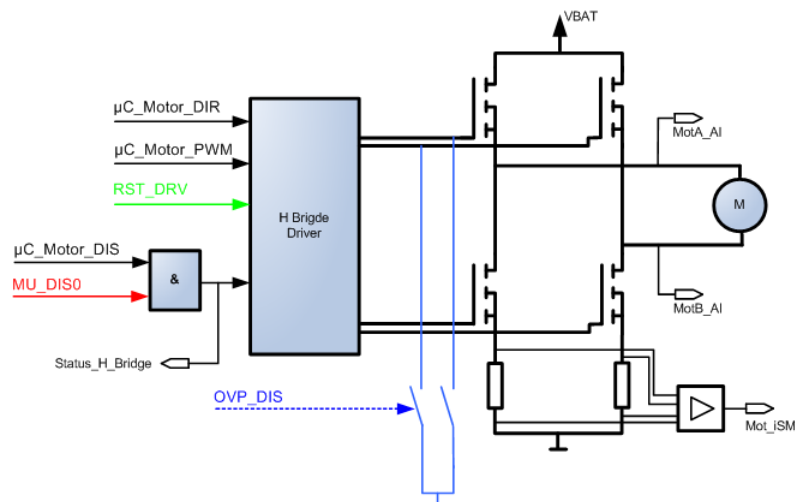
- microcontroller-ul Infineon TC1782F: controller-ul principal
- o unitate de monitorizare (Monitoring Unit) cu rol de supraveghere a activității microcontroller-ului
- interfețe pentru achiziția de date din exterior
- o punte H pentru comanda motorului

#### Intrări ale sistemului:

- 3 intrări de alimentare:
  - o KL30: alimentare cu curent continuu de la bateria automobilului
  - o KL31: ground (GND)
  - o KL15: intrare digitală ce semnalizează starea de “contact” (starea imediat anterioară/premergătoare pornirii motorului, cand toate sistemele electronice și electrice sunt alimentate de la bateria mașinii)
- interfețe pentru comunicarea în rețeaua Flexray
- intrări analogice pentru cei 2 senzori de presiune, ce furnizează, de asemenea, informații referitoare la temperatură

#### Ieșiri ale sistemului:

- un motor comandat în punte H
- 2 valve, fiecare din ele comandată într-o semipunte



**Fig. 10** Motorul comandat în punte H

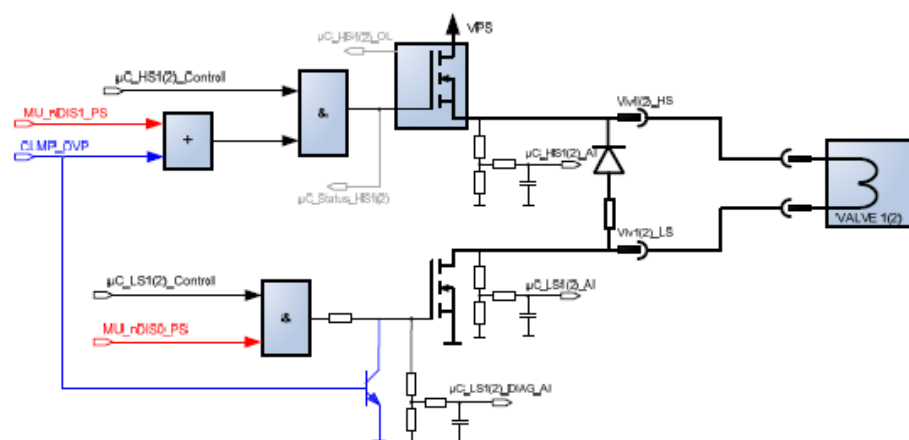


Fig. 11 Valva comandată în semipunte

## Protocolul Flexray

[4] “FlexRay este un protocol de comunicare în rețea din domeniul automotive dezvoltat de către consorțiul cu același nume. Este mai rapid și mai sigur decât CAN sau TTP, dar este, de asemenea, și mai scump. La momentul actual se lucrează la convertirea standardului FlexRay în standard ISO.

FlexRay este caracterizat, în principal, de:

- transmisii rapide de date (până la 10 Mbit/s)
- comportament declanșat pe bază de timp și evenimente
- transmisia datelor sub formă de cadre/frame-uri
- redundanță
- toleranță la erori
- deterministic

Primul automobil produs în serie care folosea tehnologia FlexRay a fost BMW X5, apărut la sfârșitul anului 2006. BMW Seria 7, introdus în 2008, a fost primul automobil din lume care să se folosească integral de tehnologia FlexRay.”

Transmisia datelor se realizează asemănător cu transmisia pe CAN:

- există mai multe mesaje transmise între ECU-urile dintr-un automobil, fiecare mesaj fiind compus din mai multe semnale;
- mesajele nu au o destinație prestabilită, oricare din ECU-uri are acces la ele.

Principalele semnale FlexRay care prezintă interes în cadrul proiectului de diplomă sunt cele referitoare la viteza automobilului și vitezele cu care rulează cele patru roți ale acestuia. Pe baza

acestor informații transmise prin RTE, software-ul funcțional poate lua decizii pertinente cu privire la comanda motorului și a valvelor.

### 2.3.3. Sistemul software

Software-ul rezultat va fi compus din trei module principale, denumite **straturi software**:

#### a. FSW (Functional Software)

Este stratul superior al software-ului, care ia decizii funcționale pe baza informațiilor primite de la senzori și emite comenzi în consecință. Deoarece FSW-ul este un strat superior, achiziția datelor de la senzori, precum și comanda elementelor mecanice din cadrul sistemului nu se fac prin acces direct la resursele unității de control (ECU - Electronic Control Unit), ci prin intermediul unor alte straturi.

Această parte de software este dezvoltată de compania Magna Powertrain (MPT) care furnizează o librărie pentru integrare, compania Continental neavând acces la codul sursă. Software-ul furnizat se conformează integral standardului AUTOSAR din domeniul automotive.

#### b. BSW (Basic Software)

Este stratul inferior al software-ului, care se ocupă cu achiziția de date de la senzori, de la convertoarele analog-numerice sau de la porturi și care se ocupă de comanda elementelor mecanice cum ar fi motorul sau valvele. BSW-ul nu ia decizii funcționale, ci doar achiziționează datele necesare FSW-ului și îndeplinește comenzile corespunzătoare deciziilor luate de acesta. Această parte a software-ului este cea care are acces direct la resursele ECU-ului.

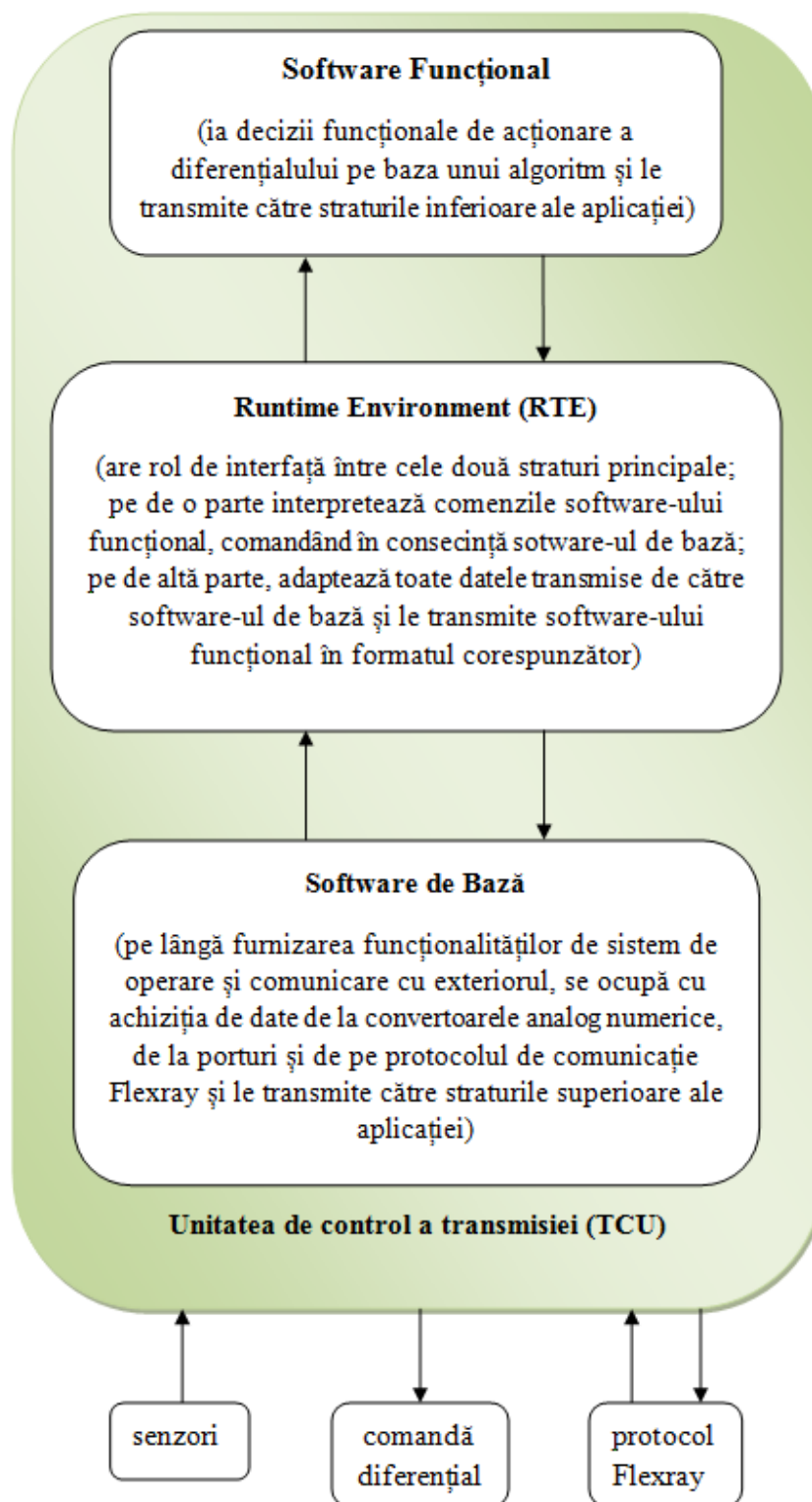
Această parte de software este dezvoltată de compania Continental, doar unele module care compun stratul conformându-se standardului AUTOSAR din domeniul automotive.

#### c. RTE (Runtime Environment)

Este, de fapt, o interfață între cele două straturi prezentate mai sus și are rolul de a facilita comunicația dintre ele. În cadrul acestui strat se realizează integrarea FSW-ului în task-urile BSW-ului. RTE-ul se ocupă, de asemenea, de toate transformările necesare a fi aplicate semnalelor interschimbate între BSW și FSW înainte de transmisia lor.

RTE va fi implementat de compania Continental, care este responsabilă și de integrarea celor două straturi sus-menționate.

Arhitectura sistemului software este prezentată, în mare, mai jos. În cadrul Continental, se dezvoltă software pentru BSW și RTE.



**Fig. 12** Arhitectura simplificată a sistemului software

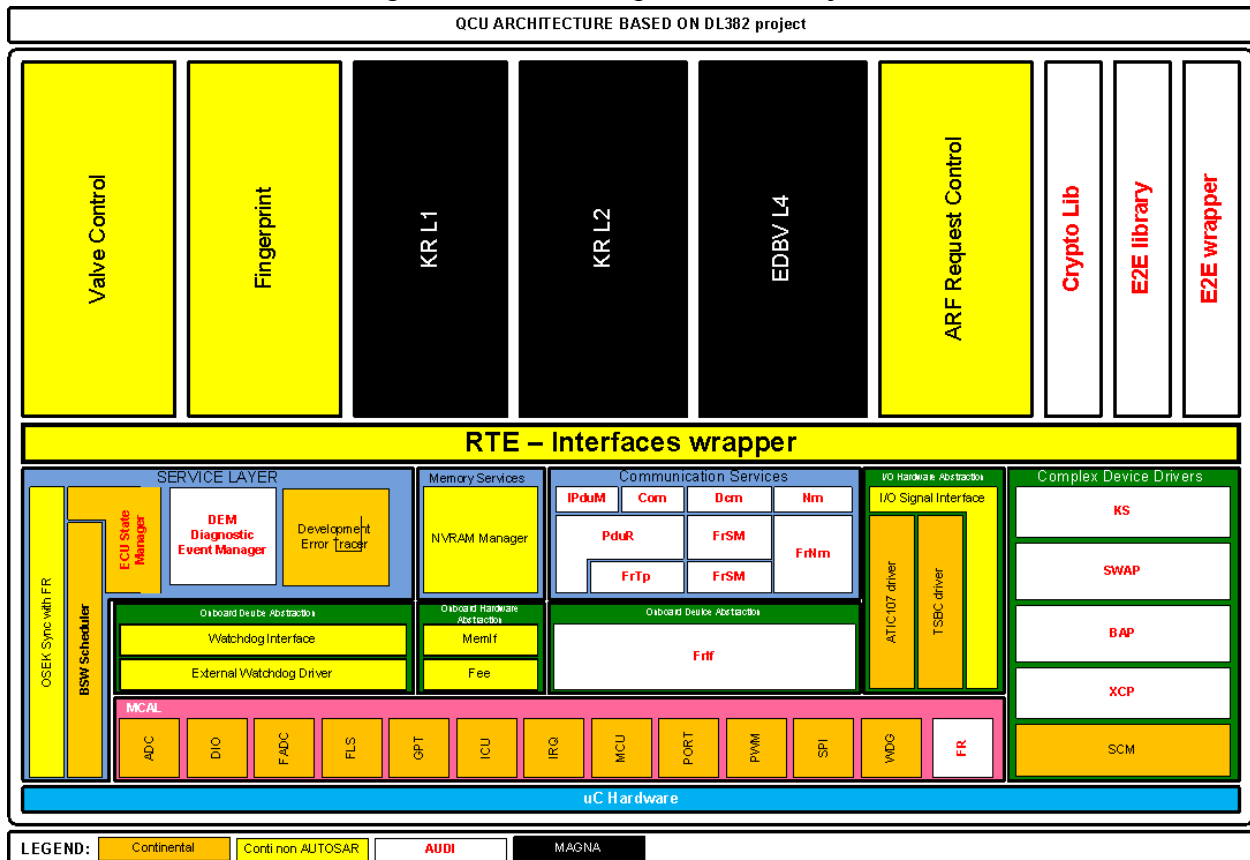
Rolul RTE-ului este de a face componentele software ce compun FSW-ul independente de maparea la un anumit TCU. Implementarea este specifică pentru fiecare TCU în parte pentru a asigura optimizarea și este, de obicei, generat. Interfața superioară este complet independentă de TCU.

BSW este dezvoltat de Continental, dar nu este complet compatibil AUTOSAR, în timp ce componentele FSW-ului sunt. Standardul permite și ca pentru acest caz să fie posibilă utilizarea unui RTE (ICC1 - Implementation Conformance Class 1):

Obiectul lucrării îl constituie analiza variantelor de implementare posibile și anume:

- dezvoltarea unui BSW complet AUTOSAR, cu generarea și configurarea automată a RTE-ului
- utilizarea unui BSW parțial compatibil AUTOSAR, cu implementarea manuală a RTE-ului
- renunțarea completă la arhitectura AUTOSAR și, în consecință, la RTE

O detaliere a arhitecturii întregului software este prezentată mai jos:



**Fig. 13** Arhitectura detaliată a sistemului software

Semnificația culorilor:

- portocaliu: modul AUTOSAR implementat de Continental
- galben: modul non-AUTOSAR implementat de Continental
- alb: modul implementat de Audi
- negru: modul AUTOSAR implementat de Magna Powertrain

### 3. Realizarea proiectului în ansamblu

#### 3.1. Prezentarea temei pe larg

##### 3.1.1. Prezentarea proiectului existent - QHCU

Proiectul QHCU (**Quattro Hidraulic Control Unit**) este varianta software deja existentă care rulează pe unitățile de control a transmisiei pe automobilele Audi Quattro Sport în prezent. Ea a fost dezvoltată în colaborare de companiile Continental și Magna Powertrain.

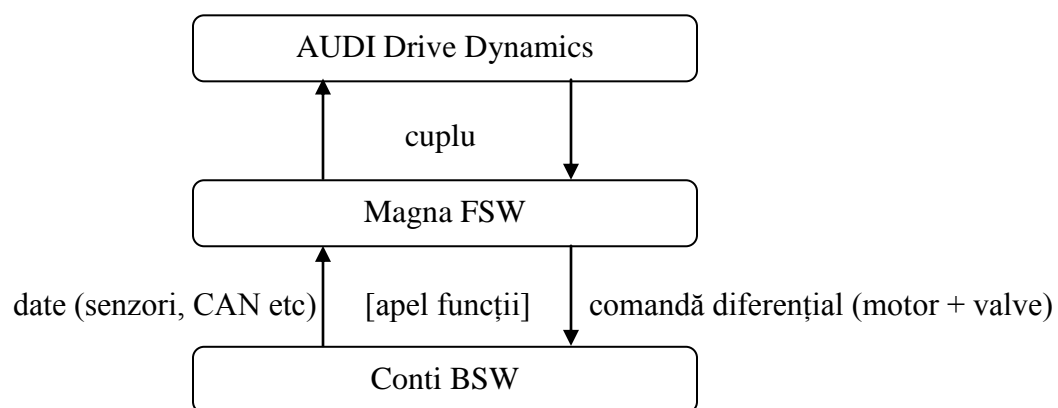
Arhitectura software-ului este cea specifică proiectelor Powertrain, compusă din:

- BSW (Basic Software): este partea de software care îndeplinește acțiunile de bază ale proiectului, cum ar fi achiziția de semnale, prelucrarea lor, distribuția/emiterea de semnale (ex: PWM) sau setarea/resetarea anumitor pini/porturi; în general, BSW se ocupă cu interacțiunea cu microcontroller-ul; în cadrul proiectului, el este compus dintr-o serie de drivere (de ex: pentru achiziția de semnale analogice, pentru manipularea protocolului CAN, pentru generarea de semnal PWM etc.) și o serie de interfețe care îndeplinesc diferite cerințe specifice proiectului;
- FSW (Functional Software): este partea de software care ia decizii funcționale pe baza datelor recepționate de la senzori (presiune, temperatură), date preluate din mesajele de CAN referitoare la viteza de deplasare a autoturismului, viteza de deplasare a fiecărei roți în parte etc. și a cuplului comandat de modulul Audi Drive Dynamics; pe baza acestor informații și a unui algoritm, modulul FSW ia decizii funcționale cum ar fi acționarea valvelor sau a motorului din cadrul diferențialului într-o anumită manieră pentru a îndeplini cerințele de funcționare.

Pentru a achiziționa date și pentru a emite comenzi, FSW apelează diferitele interfețe specifice ale BSW-ului, astfel încât între cele două module software există o comunicare directă.

Exemplu de funcționare: Pentru achiziția anumitor date cum ar fi date de la senzori, valori ale unor tensiuni, valorile unor anumiți pini, FSW apelează funcții ale BSW-ului. După interpretarea datelor, FSW comandă motorul și valvele diferențialului tot prin apel de funcții ale BSW-ului.





**Fig. 14** Funcționarea proiectului deja existent (QHCU)

### 3.1.2. Prezentarea proiectului nou - QCU

**QCU (Quattro Control Unit)** este noua variantă de software pentru unitățile de control a transmisiei din cadrul autoturismelor Audi Quattro Sport. Acesta acționează diferențialul sport aflat pe puntea din spate a autovehiculului.

Arhitectura software-ului este asemănătoare cu cea a proiectului anterior - QHCU, dar respectă și câteva din cerințele AUTOSAR privind împărțirea pe module și funcționalitățile îndeplinite de acestea. Așadar, în mare, software-ul este compus din:

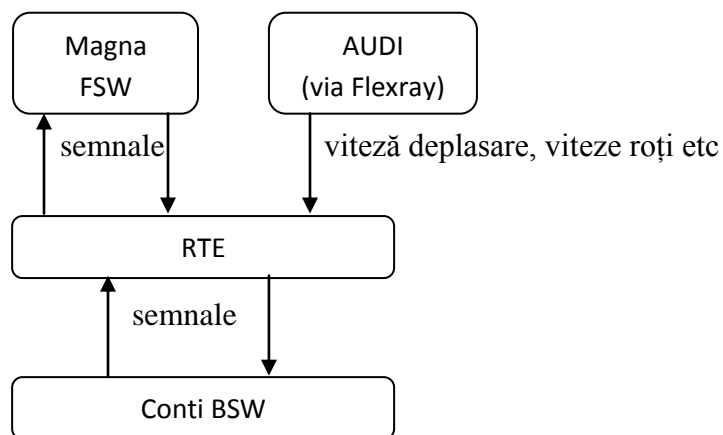
- BSW: are, în principiu același rol ca și în proiectul QHCU, dar implementează funcționalități cât mai generale, nespecifice proiectului, acestea urmând a fi preluate de alt modul - modulul RTE; el este reproiectat din punctul de vedere al arhitecturii pentru a corespunde normelor AUTOSAR de grupare a funcționalităților pe straturi și împărțirea acestora pe module (stratul de abstractizare al microcontroller-ului, stratul de abstractizare al ECU-ului (Electronic Control Unit), stratul de servicii și modulul de drivere complexe); BSW-ul achiziționează date de la microcontroller (citește valori de la convertoarele analog numerice, starea porturilor)
- FSW: ca și în proiectul QHCU, este partea de software care se ocupă de luarea deciziilor funcționale pe baza informațiilor primite de la BSW și Audi Drive Dynamics;
- RTE: este un modul de interfațare între BSW și FSW, iar prezența acestuia se datorează cerințelor AUTOSAR; are rolul de a prelua semnale și comenzi de la unul din cele 2 module prezentate anterior, de a le interpreta și de a le transmite celuilalt modul; de asemenea, se ocupă și cu sincronizarea task-urilor FSW-ului cu task-urile sistemului de operare al BSW-ului.

În proiectul cel nou - QCU - există câteva diferențe majore față de cel anterior - QHCU, diferențe prezentate pe scurt în tabelul următor:

QCU	QHCU
- comunicația cu alte ECU-uri se realizează utilizând protocolul Flexray	- comunicația cu alte ECU-uri se realizează utilizând protocolul CAN
- comunicația între BSW și FSW se realizează prin intermediul RTE-ului	- nu există o interfață specializată de comunicare între BSW și FSW
- comunicația între BSW și RTE se realizează prin intermediul semnalelor	- comunicația între BSW și FSW se realizează prin apel direct de funcții
- RTE-ul realizează citirea datelor apelând la funcționalitățile BSW-ului	- FSW-ul realizează citirea datelor apelând la funcționalitățile BSW-ului
- FSW-ul nu are acces direct la funcționalitățile BSW-ului, RTE-ul fiind cel care citește date, le interpretează, le adaptează și le distribuie	- FSW-ul are acces la unele funcționalități ale BSW-ului cum ar fi citirea datelor sau comanda diferențialului

**Fig. 15** Diferențe între proiectul vechi și cel nou

După cum am precizat în tabelul de mai sus, în proiectul cel nou, comunicația între stratul software superior și cel inferior nu se mai realizează prin apel de funcții, ci prin semnale schimbate prin intermediul RTE-ului. Astfel se asigură faptul că software-ul funcțional nu mai are acces direct la funcționalitățile software-ului de bază. Mai jos este prezentat schimbul de date între diferitele module ale proiectului:



**Fig. 16** Funcționarea proiectului deja existent (QHCU)

Noul proiect - QCU - se bazează în mare măsură pe proiectul anterior - QHCU - întrucât funcționalitatea de bază este aceeași. Noul proiect trebuie să respecte arhitectura AUTOSAR, dar el nu va fi implementat integral conform normelor impuse de standard.

Software-ul funcțional (**FSW**) este dezvoltat de compania Magna Powertrain și este livrat sub formă de fișier binar, Continental neavând acces la codul sursă, doar la librăria și interfețele necesare pentru integrare. Acesta este implementat conform AUTOSAR.

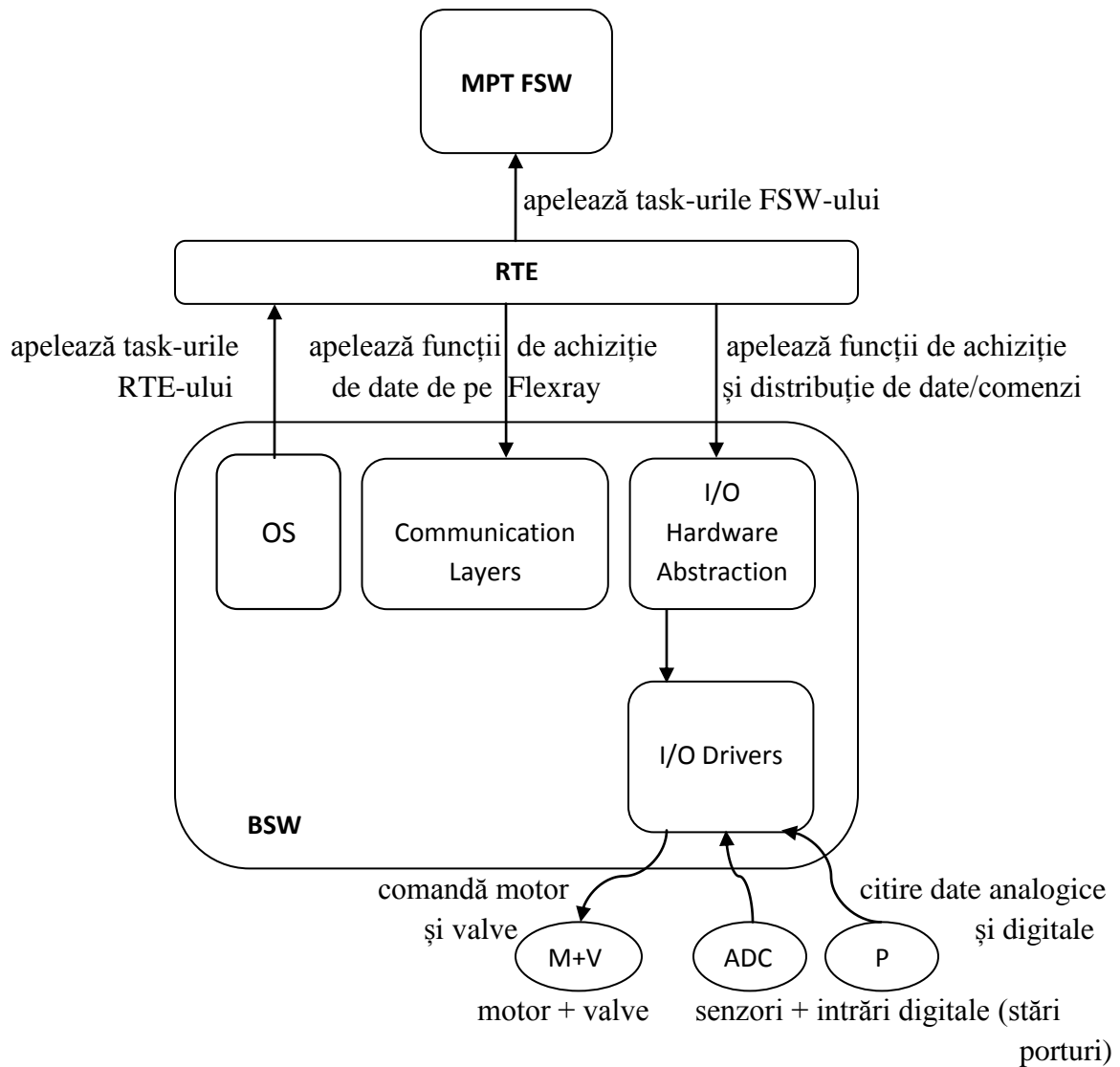
Software-ul de bază (**BSW**) este dezvoltat de compania Continental. El este împărțit pe straturi și module, doar unele din ele respectând standardul AUTOSAR.

### 3.1.3. Problema de rezolvat

Cele două părți software ale proiectului (BSW și FSW) trebuie integrate prin intermediul RTE-ului, astfel încât să fie posibilă funcționarea întregului software. Tema proiectului de licență presupune identificarea unei modalități optime de integrare a unui FSW conform cu normele AUTOSAR și a unui BSW care nu este implementat integral în spiritul standardului. În urma studiului și identificării soluției, se va implementa interfața cu rol de integrare, RTE, astfel încât BSW și FSW să interacționeze doar prin intermediul acesteia.

Implementarea RTE-ului trebuie să integreze task-urile FSW-ului în task-urile BSW-ului și să realizeze achiziția și distribuția semnalelor prin care cele două soft-uri comunică și schimbă informații.

În continuare este prezentată o privire de ansamblu asupra funcționării sistemului din punctul de vedere al ordinii și direcției de apelare a funcțiilor și asupra rolului pe care trebuie să îl îndeplinească RTE-ul în cadrul aplicației.

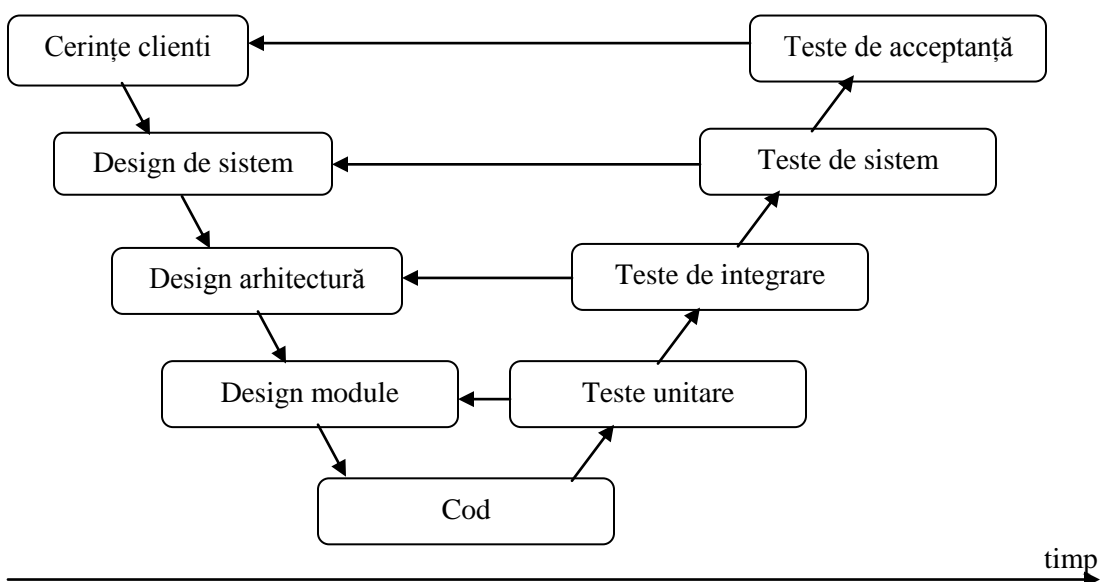


**Fig. 17** Schimburile de informații ce intervin în funcționarea proiectului

Schimbul de date între RTE și FSW se realizează cu ajutorul unei structuri de date comune între cele două module. Faptul că datele din cadrul structurii sunt pregătite pentru a fi copiate este semnalizat prin apel de funcții ale FSW-ului de către RTE.

### 3.2. Prezentare ciclu de dezvoltare V

Dezvoltarea proiectului s-a realizat urmărind fazele de dezvoltare ale ciclului V. Acest model reprezintă un proces de dezvoltare software care poate fi considerat o extensie a modelului cascadă. În locul utilizării unui model liniar pentru a ilustra fazele de dezvoltare, pașii procesului sunt reprezentați într-o formă ascendentă după faza de codare, pentru a forma imaginea literei “V” și pentru a ilustra relațiile dintre fiecare fază a ciclului de dezvoltare și faza de testare corespunzătoare.



În faza formulării cerințelor, primul pas din procesul de verificare, cerințele sistemului sunt elaborate pe baza nevoilor clienților și cerințelor acestora. În această fază se stabilește ce trebuie să îndeplinească un sistem ideal, dar nu determină cum va fi proiectat acesta. În urma acestei faze rezultă un document cuprinzând cerințele clienților, pe baza căruia se elaborează design-ul sistemului. În această fază se definesc testele de acceptanță.

Design-ul sistemului este faza în care inginerii de sistem analizează documentul de cerințe ale clienților, rezultând într-o serie de posibilități și tehnici prin care pot fi îndeplinite aceste cerințe. Documentul rezultat în urma acestui pas este cel de “Specificații software”. Tot în această fază se pregătesc și documentele pentru testele de sistem.

Faza de design a arhitecturii poate fi considerată și design de nivel înalt. Arhitectura specifică modulele din care e compus sistemul, funcționalitatea, în mare, a fiecărui modul, interfețele utilizate, dependențele etc. În această fază se definesc testele de integrare.

Faza de design a modulelor poate fi referită și ca design de nivel inferior. Sistemul este “spart” în unități mai mici sau module și fiecare din aceste module este explicat cât mai detaliat astfel încât programatorul să poată să codeze direct.

### **3.3. Planificarea activității**

Orice activitate întreprindem trebuie planificată cu atenție de la început. În felul acesta se asigură un management al timpului cât mai eficient, se pot estima mai bine momentele cheie din dezvoltarea proiectului și se pot trasa obiectivele. La fel s-a întâmplat și în cazul proiectului de diplomă.

În mare, proiectul poate fi împărțit în două activități principale:

- implementare
- documentație și prezentare

Implementarea implică atât partea de scriere efectivă a codului, cât și activitatea de documentare, studiu, precum și testare. În ceea ce privește documentația, aceasta cuprinde atât lucrarea de licență detaliată, cât și prezentările finale: teoretică și practică.

Ipotezele sau constrângerile pe care s-a bazat planificarea activității sunt:

- lucrul la proiectul de diplomă începe în data de 19.11.2012, norma de lucru fiind de 4 ore/zi
- proiectul trebuie finalizat înainte de data de 17.06.2013
- deoarece proiectul de diplomă coincide cu un proiect real al companiei, prima versiune a software-ului trebuie finalizată înainte de data de 01.04.2013, conform cerinței clientului
- implementarea nu poate începe înainte de 01.01.2013, deoarece nu s-au stabilit toate cerințele

În concluzie, am identificat următoarele activități care necesită planificare:

- documentare și studiu
  - o documentare cu privire la standardul AUTOSAR
  - o documentare cu privire la cerințele proiectului vechi
  - o documentare cu privire la cerințele proiectului nou și modificările apărute față de proiectul vechi
  - o studiul funcționării proiectului vechi
  - o studiul funcționării unui proiect deja implementat conform cu cerințele AUTOSAR
- elaborare specificații
  - o elaborarea cerințelor proiectului pe baza cerințelor clientului
  - o elaborarea cerințelor proiectului pe baza cerințelor AUTOSAR
  - o introducerea cerințelor în programul dedicat evidenței acestora, DOORS
- elaborare arhitectură și design
- implementare pe baza design-ului
  - o scrierea propriu-zisă a codului
  - o compilarea proiectului
  - o introducerea codului în utilitarul de versionare MKS
- testare implementare
- pregătire documentație proiect de diplomă
- pregătire prezentare teoretică (prezentare powerpoint) și practică

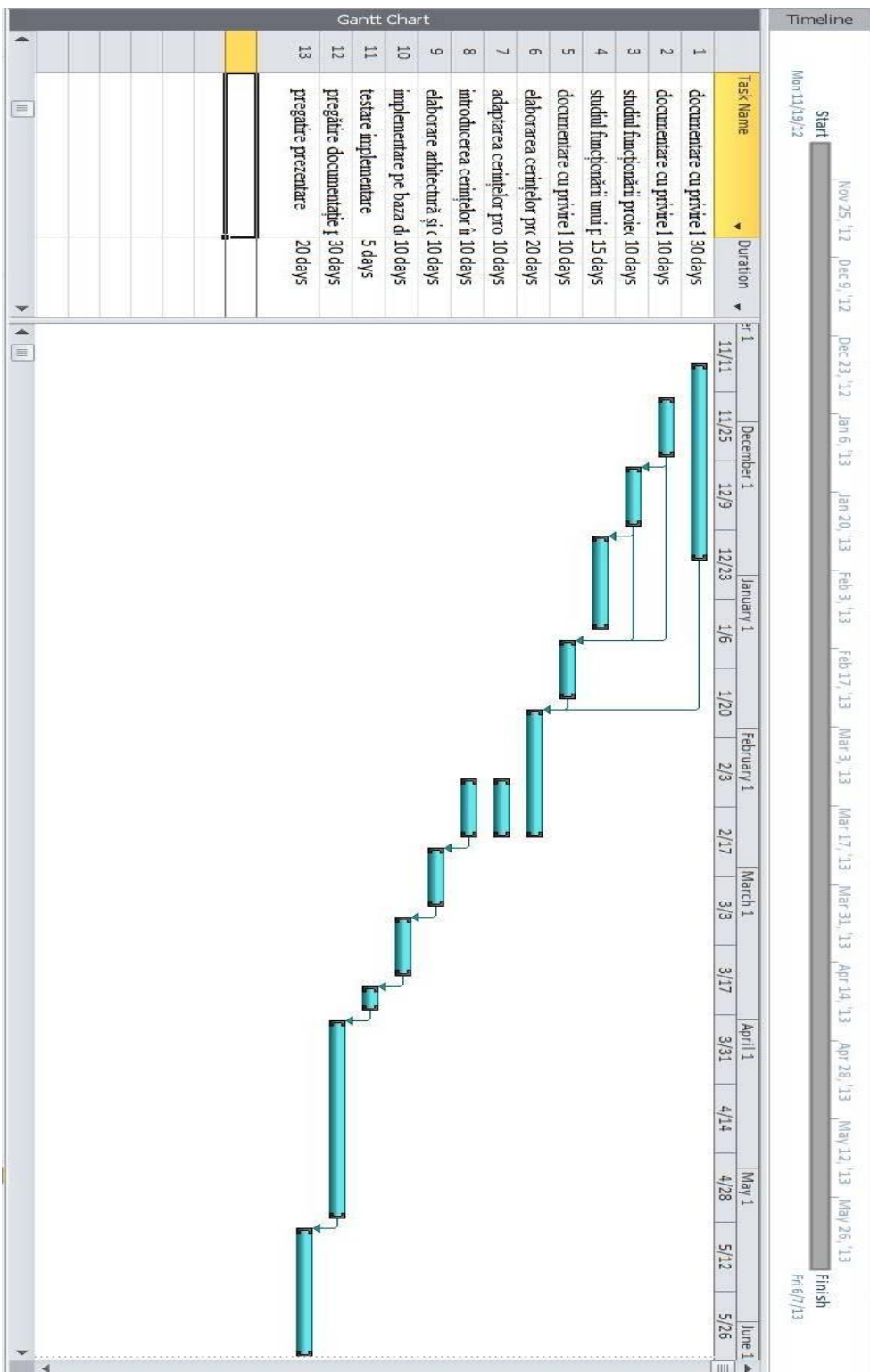
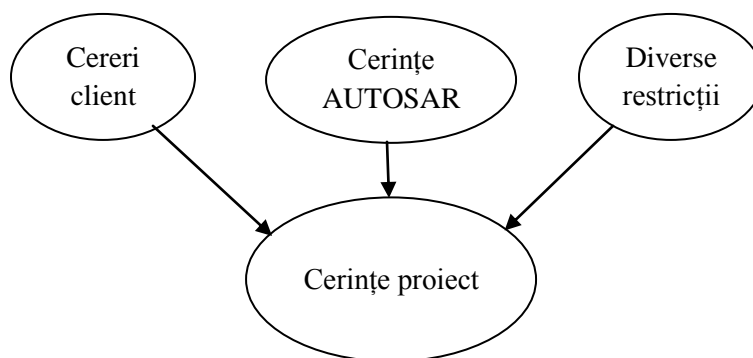


Fig. 18 Diagrama Gantt a planificării activității



#### 4. Elaborarea cerințelor proiectului

Faza de analiză și elaborare a cerințelor este prima etapă din dezvoltarea oricărui proiect software și una din cele mai importante, în care se analizează **cererile** venite din partea clientului (**requests**) și, pe baza acestora, se formulează **cerințele** proiectului (**requirements**). Această etapă poartă numele de **Requirement Engineering**. În lucrarea de față, în formularea cerințelor proiectului s-au luat în considerare cererile din partea clientului, cerințele standardului AUTOSAR și alte cerințe sau restricții de natură tehnică sau nu, cum ar fi resursele umane, restricții legate de timp sau tehnologii utilizate:



O importanță deosebită în înțelegerea acestei faze din cadrul procesului de producție o deține diferențierea clară a **cererilor** față de **cerințe**:

- **cererile** sunt formulate de client, care, de cele mai multe ori, nu aparține domeniului software, iar acestea nu sunt redactate într-un mod științific și pot fi interpretabile sau neclare;
- **cerințele** sunt formulate de furnizorul produsului software, în urma analizei cererilor clientului, respectând norme de redactare care determină cerințele să fie cât mai clare și elimină posibilitatea interpretării lor greșite.

În această fază se investește mult timp în înțelegerea cererilor clientului și identificarea cerințelor, care pot reieși în mod explicit din formularea cererii, sau pot fi implicate (cerințe care nu reies în mod direct din cererile clientului, dar fără îndeplinirea cărora produsul nu va funcționa corespunzător).

Importanța fazei de analiză și formulare a cerințelor este datorată mai multor motive:

- arhitectura, design-ul, codul și testele vor fi implementate pe baza lor;
- cu cât cerințele sunt formulate mai corect, cu atât vor apărea mai puține erori funcționale;
- în cazul apariției unei erori, este preferabil ca ea să fie depistată în fazele incipiente ale proiectului, întrucât costurile de remediere ale acesteia sunt considerabil mai reduse pe măsură ce ea este depistată mai devreme; acordând o atenție sporită formulării cerințelor, erorile, ambiguitățile, neclaritățile și interpretările sunt clarificate și remediate fără costuri suplimentare prea ridicate.

În elaborarea cerințelor trebuie să se țină cont de o serie de aspecte, cum ar fi:

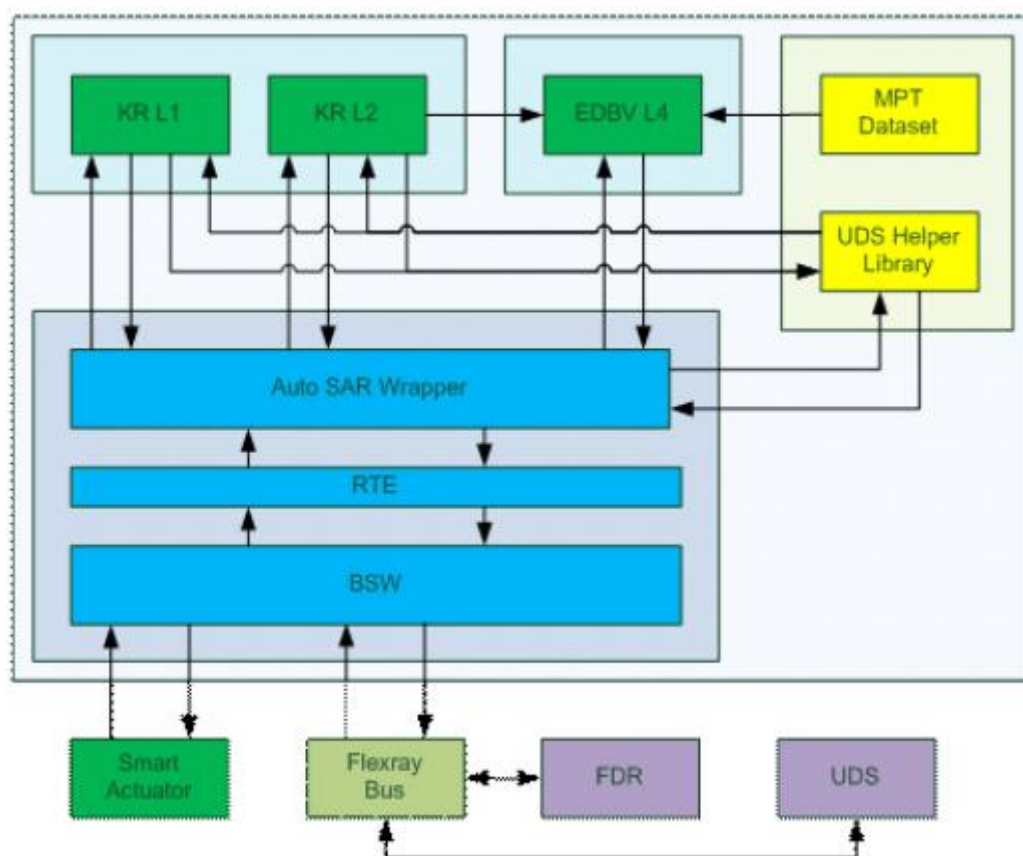
- cerințele trebuie să fie testabile: testele care verifică îndeplinirea lor vor fi formulate pe baza cerințelor; dacă cerința este formulată într-o manieră greșită, ea nu va putea fi testată corespunzător, produsul nemaicorespunzând normelor de calitate impuse de client;
- o cerință trebuie să conțină o singură idee, pentru a fi cât mai clară și neinterpretabilă; prin aceasta se simplifică, de asemenea, și elaborarea testelor;
- în elaborarea cerințelor trebuie să se acorde o atenție deosebită formulării acestora: pentru a elimina formulările ambigue se evită utilizarea unor anumite cuvinte și se construiesc propoziții cât mai simple și scurte, preferându-se termenii tehnici.

#### **4.1. Cererile clientului**

Clientul final al proiectului este compania Audi. Acesta dorește implementarea unui nou TCU pentru modelul Audi Quattro Sport. În consecință, compania Magna Powertrain va livra partea de software funcțional (FSW) iar compania Continental are rolul de a furniza BSW-ul și de a realiza integrarea celor două prin RTE, modul prezentat în capitolul anterior.

Pentru a realiza această integrare, este nevoie de un mod de comunicare, de schimb de date și comenzi între diferitele module software furnizate de cele două companii. RTE-ul deține acest rol de interfață între BSW și modulele FSW-ului, precum și între modulele FSW.

FSW-ul livrat de Magna este compus din 3 module separate. Mai jos este ilustrată schematic arhitectura software-ului complet:



**Fig. 19** Arhitectura proiectului cuprinsă în cererile clientului

Unde:

- modulul L1 realizează controlul ambreiajului
- modulul L2 este un strat de siguranță și supraveghează activitatea modulului L1
- modulul L4 (EDBV) este un strat de siguranță care împiedică transferul unui cuplu prea mare la roțile automobilului

Pe langa aceste 3 module, RTE-ul interacționează și cu o librărie de UDS, care este un protocol folosit pentru diagnoză, reprogramare, citire/scriere a memoriei.

După cum am menționat în capitolul anterior, acest proiect se bazează pe un altul mai vechi, în care schimbul de date și comenzi se realiza direct, prin apel de funcții din BSW sau FSW, nefiind utilizată o interfață specializată. În cazul de față, interfațarea se face printr-o serie de semnale, iar în urma analizării lor, RTE-ul va efectua acțiunile și apelurile corespunzătoare. Semnalele sunt definite de Magna și au fost furnizate Continental-ului sub forma unui tabel conținând detalii referitoare la format, conținut, precum și referințe către proiectul vechi, unde este cazul.

Cererile din partea Magna sunt, în principal, referitoare la aceste semnale și sunt cuprinse în fișierul furnizat. De asemenea, cererile funcționale și cele mai speciale referitoare la anumite semnale sunt cuprinse într-un fișier separat. Astfel de cereri ar fi:

- cereri referitoare la arhitectura proiectului
- cereri referitoare la recurență, task-uri
- modul de apel al funcțiilor furnizate de Magna
- serviciile de UDS ce trebuie implementate
- cereri referitoare la semnalele critice din punctul de vedere al siguranței
- cereri privind fișierele ce trebuie furnizate de Continental și detalii despre conținutul acestora

Din punctul de vedere al integrării BSW-ului și FSW-ului prin intermediul RTE-ului, cerințele din partea Magna sunt următoarele:

- RTE-ul trebuie să fie capabil să interacționeze cu modulele FSW-ului, care sunt dezvoltate după regulile impuse de standardul AUTOSAR; schimbul de date și apelul funcționalităților trebuie să fie posibil atât în cazul dezvoltării unui RTE AUTOSAR, cât și în cazul contrar;
- se vor integra următoarele „task”-uri ale FSW-ului:
  - o Task\_init() - task-ul de inițializare;
  - o EDB\_Init() - task-ul de inițializare al modului EDBV;
  - o Task\_ST() - se va apela cu o recurență de 2 ms;
  - o Task\_MT() - se va apela cu o recurență de 10 ms;
  - o Task\_Shutdown() - task-ul de finalizare a activității FSW-ului;
  - o EDB\_PressureMonitoring() - se va apela cu o recurență de 10 ms, după apelarea Task\_MT();
- înainte de apelarea „task”-urilor, se vor apela funcții de copiere a valorilor semnalelor din BSW în FSW; funcțiile sunt livrate de Magna, iar modul lor de utilizare și secvențele de apelare sunt prezentate într-un tabel cuprins în documentul de cereri;
- înainte de apelarea funcțiilor de copiere menționate mai sus, semnalele necesare pentru un anumit „task” vor fi pregătite; lista de semnale primită din partea Magna cuprinde și precizări cu privire la task-ul în care va fi folosit fiecare semnal, cel de 2 sau cel de 10 secunde;
- după apelarea „task”-urilor, se vor apela funcții de copiere ale valorilor semnalelor din FSW în BSW; și aceste funcții sunt furnizate de Magna, utilizarea lor fiind descrisă în documentul de cerințe;
- după apelarea funcțiilor de copiere menționate mai sus, semnalele necesare pentru un anumit „task” vor fi distribuite; și în acest caz se vor respecta precizările din lista de semnale;

- din BSW se vor exporta anumite funcții și macro-uri în anumite fișiere header; acestea sunt prezentate pe larg în documentul de cerințe primit de la Magna.

#### 4.2. Cerințe AUTOSAR

Deoarece se dorește dezvoltarea unui RTE conform cu specificațiile AUTOSAR, în analiza și, în final, extragerea cerințelor proiectului, s-au luat în considerare și cerințele standardului. Acestea sunt publice și se găsesc pe site-ul [www.autosar.org](http://www.autosar.org).

Cerințele cuprinse în documentele mai sus menționate au fost analizate separat, urmărindu-se, în primul rând, aplicabilitatea lor în proiectul de față și impactul pe care îl va avea implementarea sau neimplementarea lor. De asemenea, ele au fost confruntate cu cererile clientului în vederea identificării posibilelor contradicții. Au fost luate în considerare și respectate doar acele cerințe aplicabile proiectului și care ajută implementarea integrării BSW-ului și FSW-ului prin RTE.

Au fost studiate atât cerințele generale, referitoare la întreaga arhitectură a sistemului, cât și cerințele particulare referitoare la arhitectura RTE-ului, rolul lui în cadrul sistemului și interacțiunea acestuia cu restul componentelor.

În urma pașilor enumerați mai sus, cerințele relevante pentru proiectul de față au fost cuprinse în tabelul de mai jos. De asemenea, în tabel este precizată și referința către cerința AUTOSAR originală:

Nr. crt.	ID	Cerinta
1	[RTE00019] RTE is the communication infrastructure	toate schimburile de date și semnale între BSW și FSW și între componentele FSW-ului se realizează doar prin intermediul RTE-ului
2	[RTE00020] Access to OS	toate task-urile și funcționalitățile sistemului de operare vor fi invizibile componentelor FSW-ului; aceasta înseamnă că FSW-ul nu va avea posibilitatea de accesare a funcționalităților sistemului de operare, iar funcționarea acestuia, succesiunea TASK-urilor și accesarea resurselor va fi invizibilă FSW-ului
3	[RTE00068] Signal initial values	toate semnalele pentru care este precizată o valoare inițială vor fi inițializate de către RTE; documentația referitoare la cerințele clientului cuprinde și aceste valori inițiale pentru diferitele semnale ce se doresc a fi transmise; în cazul în care nu este precizată o valoare inițială pentru un semnal, acesta va fi inițializat cu o valoare implicită

4	<b>[RTE00073]</b> Data items are atomic	fiind vorba de o interfață, în cazul în care este necesară prelucrarea unor parametrii sau semnale în cadrul RTE-ului, se vor crea copii locale ale acestora, pentru a evita situația în care sunt modificate accidental date originale
5	<b>[RTE00021]</b> Per-ECU RTE customization	este recomandat a se folosi funcții dedicate unei anumite operații, în detrimentul funcțiilor generice, care sunt mai complexe prin faptul că trebuie să decidă singure asupra acțiunilor de realizat; de exemplu, pentru achiziția și prelucrarea fiecărui semnal se vor implementa funcții dedicate în cadrul modului corespunzător din RTE
6	<b>[RTE00055]</b> Use of global namespace	toate entitățile vizibile unei componente a FSW-ului (variabile, constante, funcții etc) se vor conforma unei convenții de denumire, cunoscută și de autorul componentei
7	<b>[RTE00098]</b> Explicit Transmission <b>[RTE00129]</b> Implicit Transmission <b>[RTE00128]</b> Implicit Reception <b>[RTE00141]</b> Explicit Reception	se vor asigura mecanisme de transmisie/recepție implicită, respectiv explicită: transmisia/recepția implicită presupune posibilitatea de a accesa o anumită zonă de memorie cu scopul de a citi sau scrie informații fără a face o cerere în acest sens, iar cea explicită presupune transmiterea unei cereri explicite de citire sau scriere a datelor
8	<b>[RTE00179]</b> Support of Update Flag for Data Reception	în cazul comunicației emițător-receptor, unde ultima recepție este cea luată în considerare, dacă configurația o cere, RTE-ul va asigura un fanion (flag) indicând faptul că datele au fost actualizate de la ultima operație de citire
9	<b>[RTE00052]</b> Initialization and finalization of components	RTE-ul va inițializa și va finaliza toate componentele software ale FSW-ului; această cerință va fi îndeplinită prin apelarea unor anumite funcții de inițializare ale componentelor conform cu specificațiile primite din partea clientului
10	<b>[RTE00100]</b> Compiler independent API	API-ul RTE-ului va fi independent față de compilator și platformă, pentru a nu fi nevoie de modificarea codului atunci când componenta software este mutată pe alt ECU sau când este schimbat compilatorul
11	<b>[RTE00059]</b> RTE API passes 'in' primitive data types by value	toți parametrii de intrare care sunt tip de dată primitiv, cu excepția tipului "string" vor fi transmiși prin valoare, iar cei care sunt tip de dată complex sau "string" vor fi transmiși prin referință

	<b>[RTE00060]</b> RTE API shall pass 'in' complex data types by reference	
<b>12</b>	<b>[RTE00115]</b> API for data consistency mechanism	RTE-ul va furniza mecanisme de asigurare a corectitudinii datelor, cum ar fi algoritmi de calculare a checksum-urilor sau de calculare a valorilor complementare ale datelor
<b>13</b>	<b>[RTE00116]</b> RTE Initialization and finalization	se vor asigura mecanisme de inițializare și finalizare a RTE-ului; această cerință va fi îndeplinită prin construcția unor funcții care să îndeplinească rol de inițializare și finalizare

**Fig. 20** Cerințele standardului AUTOSAR luate în considerare

#### **4.3. Analiza cerințelor în vederea alegerii variantei optime de implementare**

O parte importantă în elaborarea lucrării de licență o reprezintă analiza variantei optime în ceea ce privește dezvoltarea RTE-ului.

Mai sus am prezentat, pe scurt, **3 variante de implementare** a integrării BSW-ului și a FSW-ului:

- dezvoltarea unui BSW complet AUTOSAR, cu generarea și configurarea automată a RTE-ului
- renunțarea totală la arhitectura AUTOSAR și, în consecință, la RTE
- utilizarea unui BSW parțial compatibil AUTOSAR, cu implementarea manuală a RTE-ului

În continuare este analizată, separat, fiecare variantă, din punctul de vedere al avantajelor și dezavantajelor pe care le presupune, luând în considerare, pe lângă cererile formulate de către client și standardul AUTOSAR, și posibilitățile de timp sau resurse umane și materiale.

#### 4.3.1. Varianta 1 - Proiect complet standardizat AUTOSAR

Dezvoltarea unui BSW complet AUTOSAR presupune respectarea atât a arhitecturii standardizate, cât și a cerințelor și restricțiilor privind funcționalitățile fiecărui modul, interfațarea și interacțiunea între diferitele module etc. Toate acestea sunt cuprinse în documentația disponibilă pe site-ul [www.autosar.org](http://www.autosar.org).

Disponând de un BSW complet AUTOSAR și un FSW care, de asemenea, respectă cerințele AUTOSAR, este posibilă generarea și configurarea automată a RTE-ului. În acest sens, compania Vector a dezvoltat o suită de tool-uri cu care se poate realiza această generare și configurare automată a RTE-ului.

##### AVANTAJE:

- codul pentru RTE este generat automat, deci implementarea este mult mai rapidă și se lovește de mult mai puține erori;
- prin implementarea completă a unui BSW AUTOSAR și generarea automată a RTE-ului, se dobândește o cantitate semnificativă de cunoștințe în acest domeniu, fiind primul proiect dezvoltat în această idee din departament;
- dacă se implementează un BSW complet AUTOSAR, el poate fi folosit integral sau doar module din cadrul lui și în alte proiecte, efectuând eventual modificări doar în straturile inferioare, care interacționează direct cu ECU-ul.

##### DEZAVANTAJE:

- pentru dezvoltarea unui BSW complet AUTOSAR este nevoie de un timp mai îndelungat care cuprinde și o perioadă de studiu și analiză a cerințelor AUTOSAR; fiind primul proiect de acest gen din departament, programatorii nu dețin cunoștințele necesare;
- compania Continental nu deține tool-urile de generare și configurare a RTE-ului, acestea necesitând a fi cumpărate;
- după investirea unui timp îndelungat în analiza fezabilității aplicării standardului AUTOSAR, există riscul de a descoperi că aplicarea lui nu este eficientă, intrând în criză de timp în ceea ce privește finalizarea proiectului propriu-zis.

##### CONCLUZII:

Dezvoltarea unui BSW complet AUTOSAR este ceva nou în cadrul departamentului, nu se știe exact de cât timp este nevoie pentru a duce la bun sfârșit implementarea acestuia. Termenele de livrare ale produsului software au fost deja stabilite, iar încercarea a ceva nou în acest moment



este considerat ca fiind riscant în ceea ce privește managementul timpului. Se poate pierde prea mult timp cu studiul și înțelegerea cerințelor AUTOSAR, precum și cu familiarizarea cu tool-urile necesare generării automate a RTE-ului, ceea ce poate duce la imposibilitatea livrării la timp a produsului. Ținând cont de aceste aspecte, s-a ajuns la concluzia că avantajele prezentate de această variantă sunt nesemnificative în comparație cu dezavantajele, în final renunțându-se la varianta de implementare full-AUTOSAR.

#### **4.3.2. Varianta 2 - renunțarea completă la standardul AUTOSAR**

După cum este prezentat mai sus, proiectul se bazează pe unul vechi, în care interfațarea între BSW și FSW se realizează prin apel direct de funcții între cele două. Una din variantele de implementare luată în considerare este cea de renunțare la arhitectura AUTOSAR și ceea ce aceasta presupune. Așadar, în acest caz, interfațarea nu va mai fi realizată de RTE, iar comunicarea, schimbul de date și accesul la funcționalități se realizează prin apel direct de funcții ale BSW-ului sau FSW-ului.

##### **AVANTAJE:**

- proiectul actual va fi foarte ușor de adaptat, deoarece se va baza în totalitate pe mecanismele celui vechi; eventualele modificări ce pot apărea se referă la valorile ce intervin în calcule, rezoluții etc, algoritmi nefiind necesar a se modifica;
- programatorii dețin cunoștințele necesare implementării tuturor funcționalităților și nu mai este nevoie a investi timp în documentare;
- proiectul îndeplinind aceleași funcționalități și având, în mare, aceeași arhitectură, se pot reutiliza părți din proiectul vechi, nefiind nevoie a fi reimplementate;
- implementarea și codul nu vor fi restricționate de standardul AUTOSAR, se vor lua în considerare doar regulile de codare a microcontroller-elor în limbajul C.

##### **DEZAVANTAJE:**

- nu se descoperă/învață nimic nou;
- proiectul rămâne pe arhitectura clasică, ceea ce nu este în concordanță cu tendințele actuale de implementare;
- proiectul va deveni curând depășit, orice cerere de innoire/îmbunătățire apărută ulterior presupunând implementarea standardizată AUTOSAR;
- interfețele FSW-ului sunt deja implementate pentru a fi utilizate într-o arhitectură AUTOSAR; mai concret, conform standardului, interfețele sunt reprezentate, de fapt, de o serie de semnale/variabile la care au acces atât BSW, cât și FSW; în implementarea clasică, fără RTE, aceste semnale nu își au rostul.

## CONCLUZII:

Daca proiectul este dezvoltat pe arhitectura clasică, renunțându-se la standardul AUTOSAR, avantajul principal este acela că proiectul va fi finalizat într-un timp semnificativ mai scurt. În orice caz, tendința este ca, pe viitor, standardul AUTOSAR să fie aplicat în toate proiectele din domeniul automotive, așa că investirea timpului în studiul acestuia cât mai devreme reprezintă un avantaj, mai ales când încă nu există obligativitatea aplicării lui. Ținând cont de cele de mai sus și de faptul că pe viitor vor apărea tot mai multe proiecte bazate pe AUTOSAR, caz în care ar fi de ajutor cunoștințe prealabile referitoare la acest standard, s-a decis să se renunțe și la aceasta variantă.

### 4.3.3. Varianta 3 - proiect parțial standardizat AUTOSAR

În analiza preliminară s-a luat în considerare și posibilitatea implementării parțiale a standardului AUTOSAR în cadrul proiectului. Astfel, unele module sunt compatibile AUTOSAR, iar altele, cum ar fi sistemul de operare, sunt dezvoltate în varianta clasică sau preluate din alte proiecte. În acest caz, RTE-ul este implementat manual, nefiind posibilă generarea lui în mod automat.

#### AVANTAJE:

- sunt studiate cerințele AUTOSAR referitoare doar la câteva module, spre deosebire de cazul în care trebuia implementat întregul BSW conform standardului;
- chiar dacă nu se studiază întreaga arhitectură, se dobândește o cantitate semnificativă de cunoștințe în ceea ce privește unele module; în cazul în care pe viitor va exista cerința implementării conform AUTOSAR și a celorlalte module, acestea vor fi mai ușor de dezvoltat, beneficiind de experiența deja dobândită;
- cunoștințele vor putea fi utilizate pe viitor în proiecte și în ceea ce privește managementul timpului și a resurselor;
- modulele compatibile AUTOSAR vor putea fi refolosite în alte proiecte fără a face modificări majore sau chiar deloc.
- 

#### DEZAVANTAJE:

- se investește un timp semnificativ în studiul și analiza cerințelor AUTOSAR pentru modulele ce se doresc a fi dezvoltate bazat pe acest standard;
- deoarece proiectul nu este dezvoltat integral AUTOSAR, având doar câteva module ce respectă cerințele standardului, se pierde câteva avantaje pe care acesta le-ar aduce, cum ar fi generarea automată a codului sau reutilizarea unor module;

**CONCLUZII:**

Daca se aplică standardul AUTOSAR doar pentru o parte din proiect, timpul investit în studiul și analiza cerințelor este mult mai mic decât în cazul implementării integrale în ideea AUTOSAR. Nefiind posibilă generarea automată a codului, nu sunt necesare alte programe, așadar nu mai este necesară familiarizarea programatorilor cu instrumente noi. De asemenea, acestea nu mai este nevoie a fi achiziționate. Aceasta fiind o variantă de mijloc, care satisface atât cerințe care privesc management-ul timpului, cât și cerințe privind tendințele de viitor din domeniul automotive, cum ar fi aplicarea standardului AUTOSAR, este considerată ca fiind varianta optimă de implementare.

**4.4. Cerințele proiectului**

Formularea cerințelor este una din cele mai importante activități din cadrul oricărui proiect. Cerințele pot fi privite ca fiind fundația pe care se bazează întregul proiect, deoarece pe baza lor se elaborează arhitectura, designul, testele și se implementează codul. O cerință scrisă greșit și nedetectată ca fiind greșită în faza de analiză a cerințelor se va perpetua în fazele următoare ale proiectului. Cu cât aceasta este detectată mai târziu, cu atât costurile de remediere sunt mai mari. Așadar, cea mai mare perioadă de timp a fost dedicată analizei și elaborării cerințelor acordând o atenție deosebită modului de formulare pentru a înlătura posibilitatea de interpretare eronată a acestora.

Mai jos sunt precizate câteva reguli de elaborare a cerințelor:

- orice cerință trebuie să fie derivată dintr-o cerere a unui client;
- cerințele trebuie să fie clare pentru toți cititorii interesați;
- toate cerințele trebuie să aibă posibilitatea de a fi testate;
- cerințele trebuie să nu fie ambigue, adică să nu lase loc de multiple interpretări; fiecare din ele trebuie să conțină o cerere de îndeplinire a unui singur obiectiv;
- cerințele complexe trebuie transformate într-o serie de mai multe cerințe;
- cerințele trebuie să fie abstracte, adică, pe cât posibil, să nu ofere soluții; ele trebuie elaborate sub forma unor cutii negre și să descrie doar caracteristicile și comportamentul exterior al sistemului;
- toate cerințele trebuie să fie fezabile, adică să poată fi implementate;
- trasabilitatea unei cerințe trebuie să fie ușor de urmărit, adică să se poată determina ce anume a generat cerința și eventual cerințele generate ulterior de ea;
- toate cerințele trebuie să respecte anumite reguli de redactare, în funcție de obligativitatea fiecăreia; cerințele pot fi atât obligatorii cât și opționale, caracteristică ce reiese din formulare, în cazul în care aceasta este redactată corespunzător;

Cerințele finale ale proiectului, care vor sta la baza elaborării documentului de design, derivă din analiza cerințelor clientului confruntate cu cerințele funcționale, cerințele AUTOSAR și cu posibilitățile de implementare.

În cazul proiectului de față, am menționat anterior că din partea Magna s-au primit o serie de documente cuprinzând cerințele acestora: o listă de semnale cu rol de interfață între BSW și FSW, în care sunt precizate detaliile fiecărui semnal în parte și un document PDF cuprinzând restul de cerințe referitoare la arhitectură, cerințe funcționale, cerințe suplimentare în cazul unor semnale, precum și câteva informații referitoare la modul de funcționare al FSW-ului.

Programul utilizat pentru redactarea și gestiunea cerințelor este Rational DOORS, dezvoltat de compania IBM. Aceasta este o aplicație de management al cerințelor cu rolul de a optimiza comunicarea acestora, precum și colaborarea și verificarea între o organizație și lanțul de furnizori. Rational DOORS permite urmărirea, analiza și gestiunea modificărilor informației, păstrând, în același timp, conformitatea cu normele și standardele impuse.

Aplicația Rational DOORS permite:

- managementul cerințelor într-o locație centralizată pentru o colaborare sporită în cadrul echipei
- trasabilitate prin corelarea cerințelor cu elemente de design, planuri și cazuri de testare, precum și cu alte cerințe
- tool de urmărire/supraveghere a testării
- managementul schimbărilor cerințelor folosind un sistem simplu de propunere de schimbare sau unul mai complex, configurabil, folosind soluțiile oferite de “Rational change management”

Pentru redactarea cerințelor în DOORS, am analizat documentele primite din partea Magna. Fiecare din cerințele cuprinse în acestea le-am transpus în formatul potrivit și le-am confruntat atât cu cerințele impuse de compania Continental, cât și cu cele impuse de standardul AUTOSAR.

De asemenea, s-a decis ca proiectul să fie livrat în mai multe etape, pentru prima etapa livrându-se doar aspectele privind integrarea BSW-ului și a FSW-ului. În consecință, o dată cu redactarea, pentru fiecare cerință în parte, am decis dacă ea va fi satisfăcută pentru prima livrare sau nu.

În urma introducerii cerințelor în aplicația DOORS și numai după validarea acestora se poate continua cu dezvoltarea arhitecturii, a design-ului și se poate trece la implementarea codului. O dată cu îndeplinirea uneia dintre cerințe, în aplicație se realizează și legătura cu implementarea din cod a acesteia.

Requirement	Requirement Class	Description of the Requirement (M)	Source Reference (M)	SW Release
_321				
000aa647_236	Heading	<b>14 KR component input/output</b>		
000aa647_286	Information	Interface signals exchanged between FSW and BSW shall be defined according to Schnittstellenliste_Konsolidiert.xlsx from QCU DOORS database.	Stakeholder Request Object Identifier = 000aa5e0_3970	QCU_ISW_O100
000aa647_287	Requirement	Exchange of information between internal and external signals shall be done by copy-operations (MemCopy_2ms_From_ExternOS(), MemCopy_2ms_To_ExternOS(), MemCopy_10ms_From_ExternOS(), MemCopy_10ms_To_ExternOS()) provided from FSW.	Stakeholder Request Object Identifier = 000aa5e0_4675	QCU_ISW_O100
000aa647_288	Information	To guarantee atomic copy-operations, locking mechanisms (ACTION_INFR_DisableTaskSwitchingFast(), ACTION_INFR_EnableTaskSwitchingFast()) from BSW/OS are used.	Stakeholder Request Object Identifier = 000aa5e0_4675	QCU_ISW_O100
000aa647_289	Information	For redundant data read/write/check, macros from BSW/OS shall be used.	Stakeholder Request Object Identifier = 000aa5e0_4675	QCU_ISW_O100
000aa647_290	Requirement	Before invocation of copy-functions, external interface signals shall be prepared.	Stakeholder Request Object Identifier = 000aa5e0_4675	QCU_ISW_O100
000aa647_291	Requirement	After processing and invocation of the copy-function, the external interface signals shall be distributed.	Stakeholder Request Object Identifier = 000aa5e0_4675	QCU_ISW_O100

Fig. 21 Exemplu de cerințe introduse în aplicația Rational DOORS

## **5. Proiectarea software-ului pe baza cerințelor**

### **5.1. Design și arhitectură**

Design-ul unui proiect reprezintă explicarea detaliată a cum urmează să fie implementat un proiect. Sunt detaliate atât arhitectura și interfețele, cât și funcțiile, algoritmi și chiar variabilele utilizate. Tot aici este descris și modul de interacțiune între diferitele elemente ale proiectului, precum și ordinea de apelare a funcțiilor sau ordinea de efectuare a prelucrărilor.

#### **5.1.1. Tool-uri utilizate**

Design-ul arhitecturii și al modulelor componente s-a realizat în urma finalizării fazei de analiză, redactare și validare a cerințelor proiectului.

În aceasta fază, fiecare cerință a proiectului a fost luată în considerare, iar arhitectura și fiecare modul în parte au fost concepute astfel încât să respecte cerințele formulate în pasul anterior.

Documentele de design sunt introduse, asemenea codului, în aplicația MKS Integrity. Aceasta este o aplicație ce permite echipelor de dezvoltare software să urmărească evoluția tuturor aspectelor legate de proiecte, cum ar fi documente de design, fișiere sursa sau rapoarte de testare, în cadrul unui singur produs. MKS Integrity oferă posibilități de trasabilitate, automatizare a proceselor, raportare și analiză, precum și versionare.

#### **5.1.2. Structură**

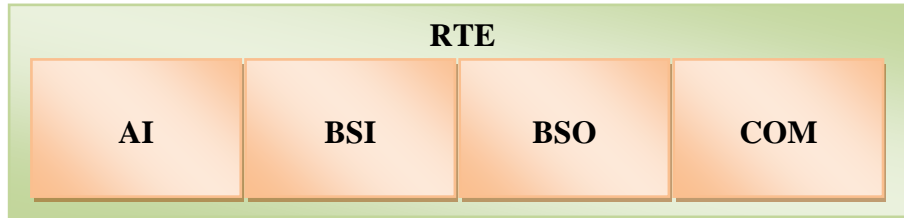
În capitolele anterioare am prezentat, într-o manieră simplificată, arhitectura proiectului. El este alcătuit, în mare, dintr-un software de bază (BSW - Basic Software) și un software aplicativ (FSW - Functional Software). Cele două comunică, conform cerințelor AUTOSAR, prin intermediul RTE-ului cu rol de interfață.

Deoarece, conform variantei de implementare adoptate, doar o parte a BSW-ului este compatibilă cerințelor AUTOSAR, proiectul se încadrează în categoria ICC1, deci RTE-ul face parte din BSW.

În continuare va fi prezentat designul adoptat pentru implementarea RTE-ului.

## A. Arhitectura RTE-ului

RTE-ul este alcătuit din mai multe module, fiecare din ele servind un anumit rol:



**Fig. 22** Componenta RTE-ului

### a. Modulul AI

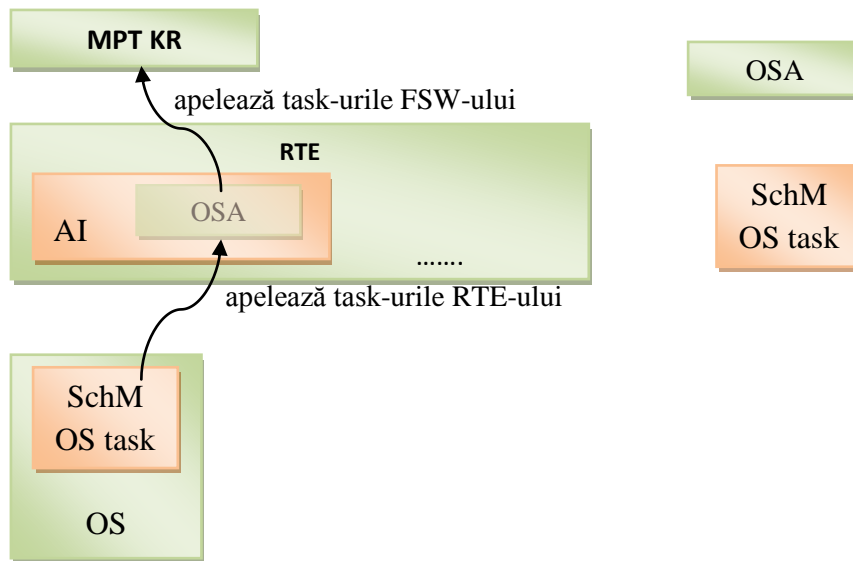
Acest modul se ocupă de interfațarea între sistemul de operare al BSW-ului și funcționalitățile FSW-ului. Acestea au fost definite în documentul de cerințe:

- Task\_init() - task-ul de inițializare
- EDB\_Init() - task-ul de inițializare a modulului EDBV
- Task\_ST() - se va apela cu o recurență de 2 ms
- Task\_MT() - se va apela cu o recurență de 10 ms
- Task\_Shutdown() - task-ul de finalizare a activității FSW-ului
- EDB\_PressureMonitoring() - se va apela cu o recurență de 10 ms, după apelarea Task\_MT()

Mai exact, luând ca exemplu task-ul Task\_ST(), pentru care în cerințele clientului este specificată o recurență de 2 ms, se efectuează următoarele operații:

- în task-ul de 2 ms al sistemului de operare al BSW-ului se apelează task-ul de 2 ms al RTE-ului
- în task-ul de 2 ms al RTE-ului se apelează task-ul de 2 ms al FSW-ului - Task\_ST()
- tot în task-ul de 2 ms al RTE-ului, înainte și după apelul funcției Task\_ST(), se apelează funcțiile de pregătire și copiere a valorilor semnalelor cu rol de interfață între BSW și FSW, precum și funcția de distribuție a acestor valori

Schematic, interacțiunea dintre BSW și FSW este ilustrată mai jos:



**Fig. 23** Interacțiunea dintre BSW și FSW

Mai departe sunt prezentate task-urile RTE-ului și modul lor de funcționare, conform cerințelor.

### 1. Task-ul de inițializare

Este apelat din task-ul de inițializare al sistemului de operare. Are rolul de inițializare atât a componentelor BSW-ului, cât și ale FSW-ului:

BSW:

- interfața de achiziție și distribuție a datelor: IoHwAb\_Init()
- emularea memoriei flash: Fee\_Init()
- managerul pentru memoria NVRAM: NvM\_Init()

FSW:

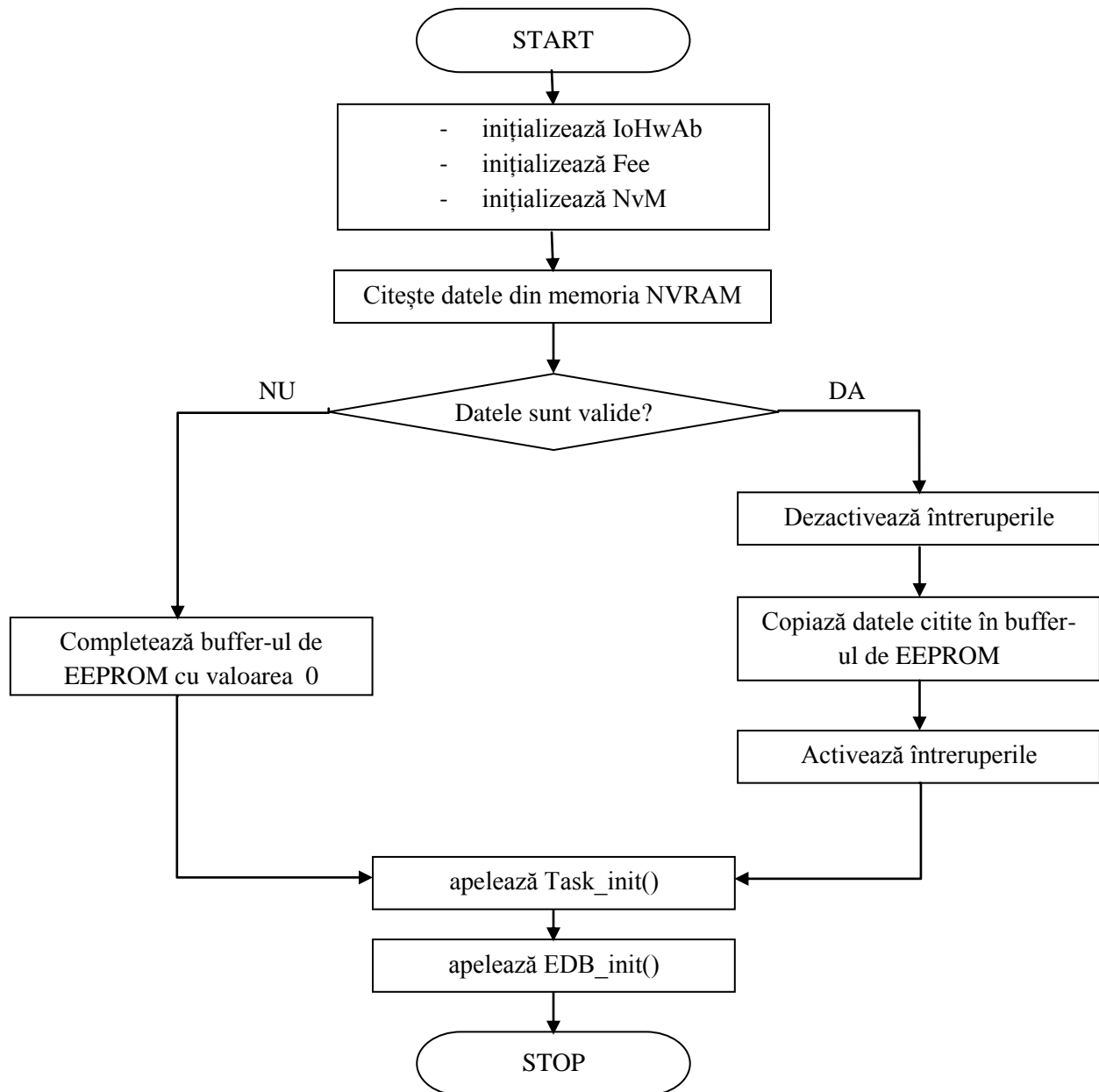
- modulul EDBV: EDB\_Init()
- apelează task-ul de inițializare al FSW-ului: Task\_init()

Datele necesare de la o sesiune de lucru la alta sunt salvate în memoria NVRAM. La începutul sesiunii de rulare, pentru un acces mai rapid la aceste date, ele sunt copiate într-un buffer - EEPROM\_chbuffer[]. La sfârșitul sesiunii, buffer-ul este copiat în memoria NVRAM. Task-ul



de inițializare al RTE-ului efectuează și această operație de pregătire a bufferului EEPROM cu datele citite din NVRAM. Dacă datele sunt invalide, buffer-ul este completat cu valoarea 0.

Funcționarea task-ului de inițializare este ilustrată schematic în figura de mai jos:



**Fig. 24** Schema logică de funcționare a task-ului de inițializare

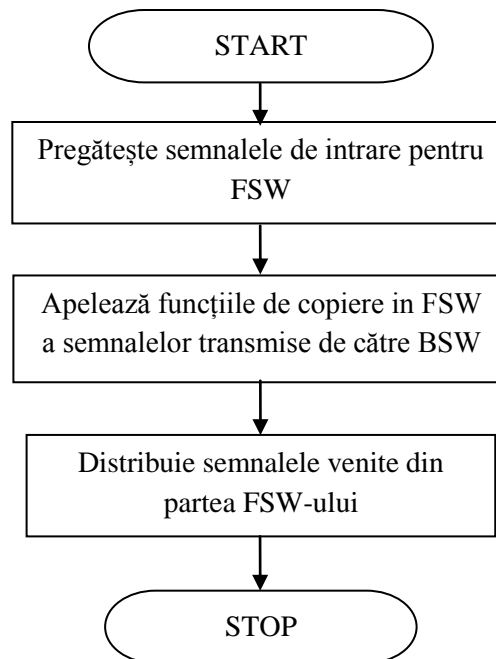
## 2. Task-urile recurente de 2 și 10 ms

Acestea sunt apelate din task-urile corespunzătoare de 2, respectiv 10 ms ale sistemului de operare al BSW-ului și efectuează următoarele operații:

- pregătesc semnalele necesare task-urilor de 2 și 10 ms ale FSW-ului: acestea pot fi semnale analogice care sunt citite de la convertorul analog-numeric sau semnale digitale, în cazul cărora se citește valoarea unui anumit port/pin; acestea se prelucrează conform specificațiilor și se salvează valoarea obținută în variabilele de interfațare cu FSW-ul;
- după ce toate semnalele au fost prelucrate și salvate în interfață, se apelează funcțiile de copiere din FSW care au rolul de a prelua valorile semnalelor din interfață și de a le salva local în FSW
- apelează task-ul corespunzător de 2 sau 10 ms al FSW-ului
- apelează funcții din FSW cu rol de copiere a valorilor semnalelor din FSW în variabilele din interfață
- distribuie semnalele corespunzătoare task-urilor de 2 și 10 ms ale FSW-ului: RTE-ul va efectua operațiunile necesare în funcție de valorile fiecăruia dintre semnale

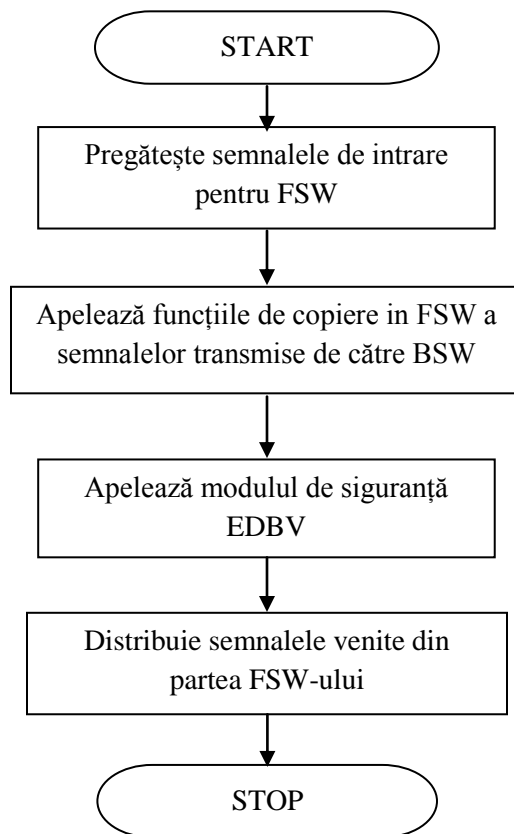
Pregătirea și distribuția semnalelor se realizează prin intermediul unor funcții ale RTE-ului. Fiecare task dispune de o funcție de pregătire a semnalelor și una de distribuție. În cadrul acestora, se folosesc funcții de citire/scriere a porturilor și a convertoarelor analog-numerice.

Funcționarea task-ului de 2 ms a RTE-ului este redată schematic în figura de mai jos:



**Fig. 25** Schema logică a funcționării task-ului de 2 milisecunde

Funcționarea task-ului de 10 ms al RTE-ului este prezentată schematic în figura de mai jos:



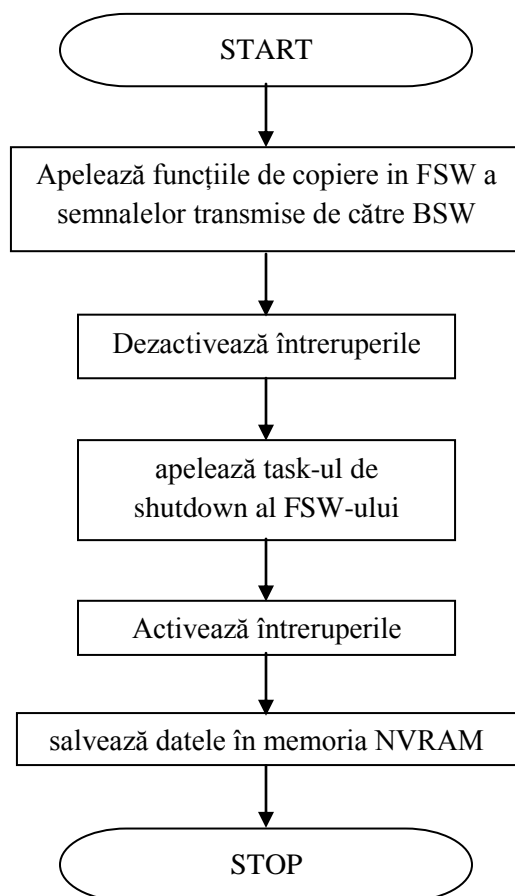
**Fig. 26** Schema logică a funcționării task-ului de 10 milisecunde

### 3. Task-ul de shutdown

Acesta este apelat din task-ul de background dacă sistemul de operare al BSW-ului se află în starea “POWER OFF” și are rolul de a opri funcționarea componentelor FSW-ului și de a realiza operații de salvare a datelor necesare pentru rulările ulterioare. În cadrul acestui task se efectuează următoarele operații:

- se apelează funcțiile de copiere ale FSW-ului pentru a înregistra local valorile tuturor semnalelor din interfață
- se dezactivează întreruperile
- se apelează task-ul de shutdown al FSW-ului
- se activează întreruperile
- se salvează datele în memoria NVRAM

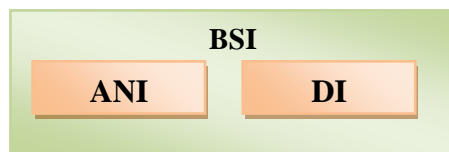
Funcționarea task-ului de shutdown este prezentată schematic în figura de mai jos:



**Fig. 27** Schema logică a funcționării task-ului de shutdown

### b. Modulul BSI - Basic Software Inputs

Acest modul are rol de interfață în achiziția de semnale analogice și digitale de la BSW și este structurat ca în figura de mai jos:

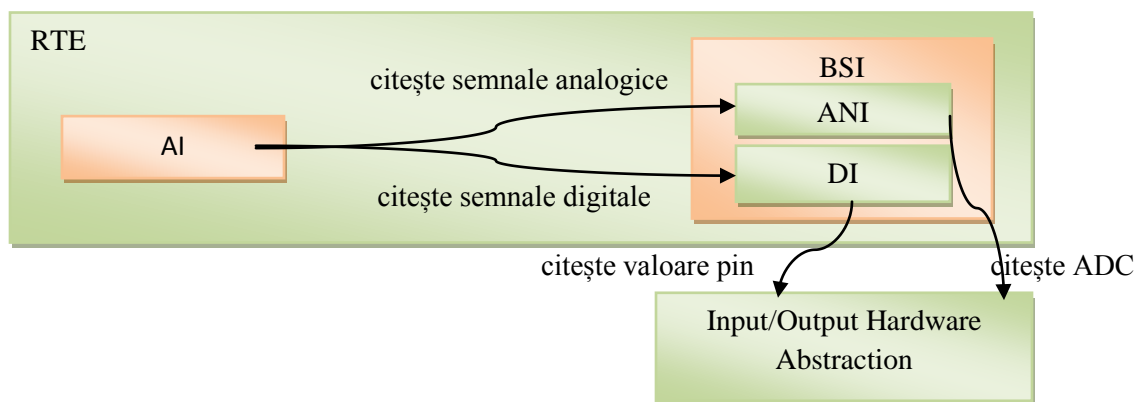


**Fig. 28** Componenta modulului de achiziție a datelor

Pentru achiziția de semnale, se apelează funcțiile corespunzătoare din modulul Input/Output Hardware Abstraction al BSW-ului.

Pregătirea semnalelor necesare unui task al FSW-ului se face prin următoarea secvență de apeluri:

- din corpul task-ului corespunzător al RTE-ului se apelează funcția de pregătire a semnalelor
- această funcție apelează, la randul ei, funcțiile din modulul BSI corespunzătoare fiecărui semnal
- în fiecare funcție a modulului BSI se apelează funcțiile corespunzătoare de achiziție de date analogice, respectiv digitale din modulul Input/Output Hardware Abstraction și se prelucrează aceste date conform specificațiilor fiecărui semnal



**Fig. 29** Funcționarea achiziției de date

#### ○ Modulul ANI - Analogue Inputs

Acest “sub”-modul se ocupă cu achiziția de semnale analogice de la BSW. Deoarece semnalele din interfață au caracteristici diferite, acest modul conține funcții dedicate fiecăruia dintre ele, iar semnalele care prezintă caracteristici identice sunt tratate de aceeași funcție.

#### ○ Modulul DI - Digital Inputs

Acest “sub”-modul se ocupă cu achiziția de semnale digitale de la BSW. La fel ca și în cazul modulului ANI, el conține funcții dedicate semnalelor sau grupurilor de semnale care prezintă caracteristici identice.

### c. Modulul BSO - Basic Software Outputs

Modulul BSO este interfața de distribuție a semnalelor primite de la FSW și este structurat conform figurii de mai jos:

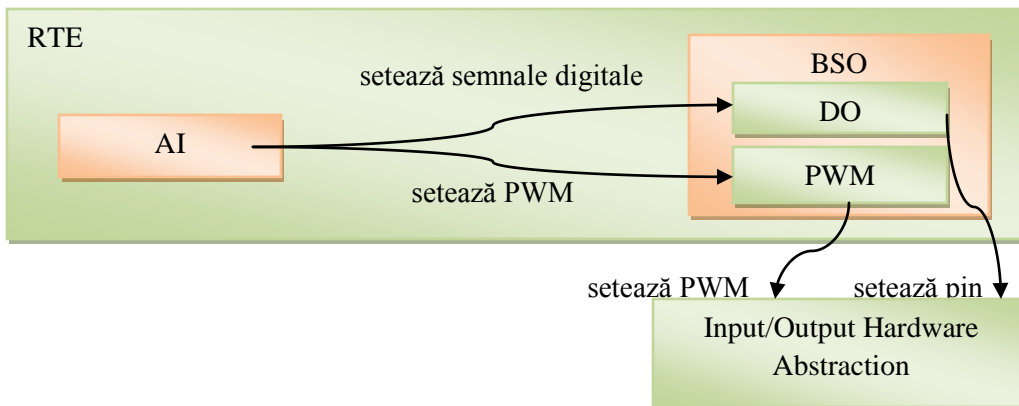


**Fig. 30** Componenta modulului de distribuție a datelor

Pentru distribuția semnalelor, se apelează funcțiile corespunzătoare din modulul Input/Output Hardware Abstraction al BSW-ului.

Distribuția semnalelor primite de la FSW se realizează prin următoarea secvență de apeluri:

- din corpul task-ului corespunzător al RTE-ului se apelează funcția de distribuție a semnalelor
- această funcție apelează, la rândul ei, funcțiile din modulul BSO corespunzătoare fiecărui semnal
- în fiecare funcție a modulului BSO se apelează funcțiile corespunzătoare de distribuție a datelor digitale, respectiv de setare a semnalului PWM din modulul Input/Output Hardware Abstraction
- Hardware Abstraction

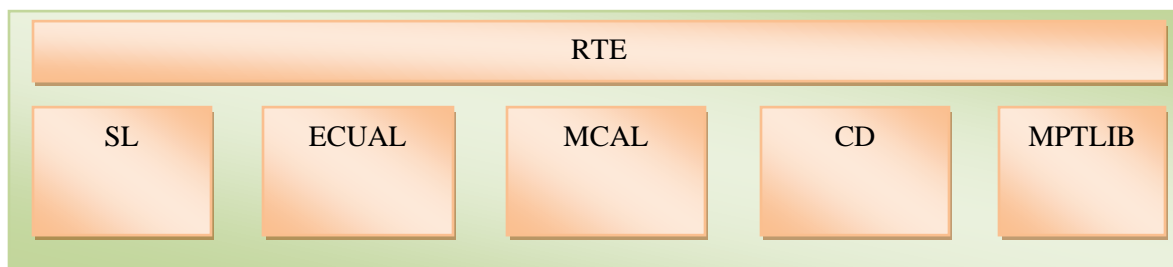


**Fig. 31** Funcționarea distribuției de date

## B. Arhitectura BSW-ului

Am precizat mai devreme că arhitectura adoptată pentru proiect este ICC1 (Implementation Conformance Class 1), adică BSW-ul nu este compatibil cu specificațiile AUTOSAR, cel puțin nu în totalitate. În acest caz, RTE-ul face parte integrantă din BSW.

Astfel, pe lângă modulul RTE, BSW-ul cuprinde și alte module, fiecare din ele specializat în îndeplinirea unor anumite funcționalități. O imagine de ansamblu asupra BSW-ului o oferă figura de mai jos:



**Fig. 32** Componenta BSW-ului

Se observă că s-a încercat proiectarea BSW-ului cât mai apropiată de modelul oferit de AUTOSAR: arhitectura nu este una modulară, ci una bazată pe straturi, fiecare din aceste straturi având rolul definit de specificațiile AUTOSAR, denumirile acestora respectând, de asemenea, standardul.

Modulele SL (Services Layer), ECUAL (ECU Abstraction Layer) și MCAL (Microcontroller Abstraction Layer) au mai fost prezentate într-un capitol anterior și nu ne vom mai opri asupra lor.

### a. Modulul CD - Complex Drivers

Acest modul oferă posibilitatea integrării unor funcționalități speciale, cum ar fi driverele unor dispozitive care nu sunt specificate în cadrul AUTOSAR sau care presupun constrângeri mari de timp. În cazul acestui proiect, modulul CD conține driver-ul necesar funcționării dispozitivului ATIC107, care controlează generarea semnalului PWM pentru comanda motorului.

Implementarea și interfața superioară pot fi dependente de aplicație, microcontroller sau de caracteristicile hardware ale ECU-ului.

## b. Modulul MPTLIB

Am menționat într-un capitol anterior că, pentru integrarea BSW-ului și a FSW-ului, o serie de fișiere header trebuie furnizate de către Magna. Modulul MPTLIB cuprinde toate aceste fișiere, precum și librăria de funcții a FSW-ului.

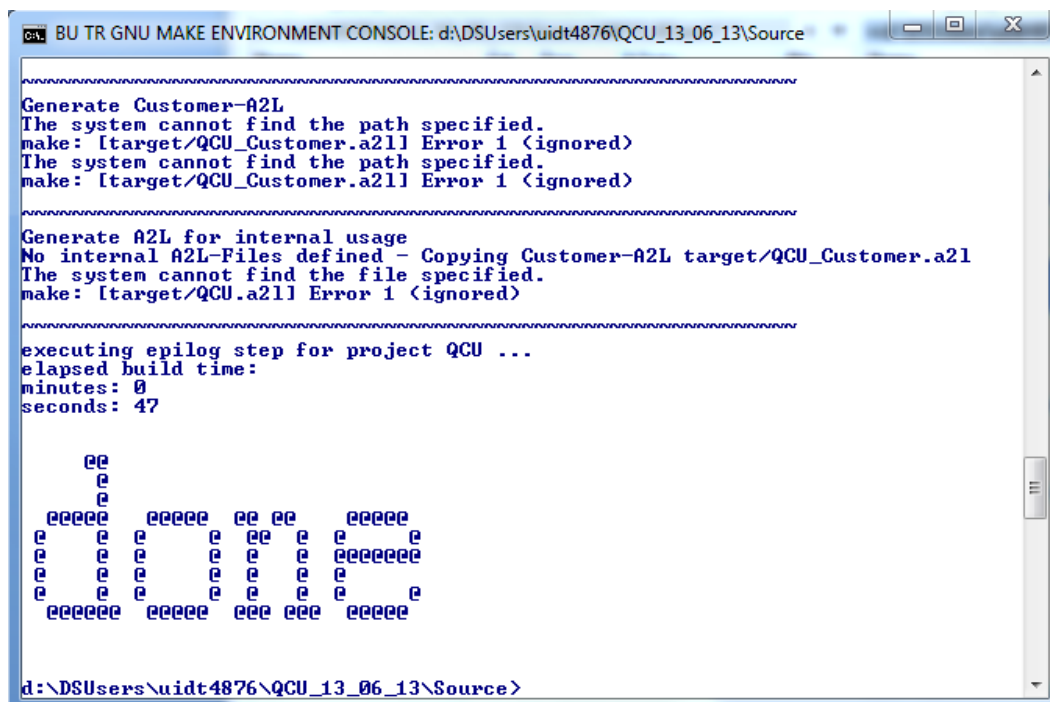
Pentru că software-ul pentru FSW (dezvoltat de Magna) și cel pentru BSW (dezvoltat de Continental) s-au elaborat în paralel, pentru început, modulul MPTLIB a fost populat cu fișiere “dummy”, urmând ca ele să fie înlocuite ulterior cu fișierele livrate de Magna.

## 5.2. Implementare

Implementarea proiectului presupune scrierea propriu-zisă a codului și compilarea lui. Așadar, codul a fost dezvoltat în mediul CodeWright utilizând limbajul de programare C, iar pentru compilare s-a utilizat un script dedicat proiectului.

În ceea ce privește implementarea, toate modulele prezentate anterior trebuie văzute ca directoare în cadrul proiectului, conținând fișiere \*.c și \*.h, sau alte directoare, în funcție de ramificațiile funcționalităților pe care trebuie să le îndeplinească fiecare modul în parte.

Pentru compilare s-a rulat un script dedicat în care s-au precizat modulele care vor fi incluse. Pentru fiecare modul în parte s-au precizat opțiunile de compilare, precum și căile la care se găsesc eventualele fișiere necesare.



```

GNU MAKE ENVIRONMENT CONSOLE: d:\DSUsers\uidt4876\QCU_13_06_13\Source

Generate Customer-A2L
The system cannot find the path specified.
make: [target/QCU_Customer.a2l] Error 1 (ignored)
The system cannot find the path specified.
make: [target/QCU_Customer.a2l] Error 1 (ignored)

Generate A2L for internal usage
No internal A2L-Files defined - Copying Customer-A2L target/QCU_Customer.a2l
The system cannot find the file specified.
make: [target/QCU.a2l] Error 1 (ignored)

executing epilog step for project QCU ...
elapsed build time:
minutes: 0
seconds: 47

      ee
      e
    eee
  eeeeee  eeee  ee ee  eeeee
e  e  e  e  e  e  e  e  e  e
e  e  e  e  e  e  e  e  e  e
e  e  e  e  e  e  e  e  e  e
  eeeee  eeeee  eee  eee  eeeee

d:\DSUsers\uidt4876\QCU_13_06_13\Source>
  
```



Întregul proiect este încărcat în aplicația MKS Integrity. Aceasta este o aplicație client/server dezvoltată de către MKS Inc. și aparută în anul 2001, al cărui rol principal este cel de gestiune a procesului de dezvoltare a proiectelor, de la managementul cerințelor, managementul cererilor de schimbare sau controlul reviziilor până la implementarea software-ului și a diferitelor rapoarte atașate.

În imaginea de mai jos se poate observa structura de fișiere din cadrul aplicației, fiecare fișier în parte având specificată versiunea corespunzătoare în urma tuturor modificărilor care i-au fost aplicate până în momentul curent:

Name	Working Rev.	Member Rev.	Locked	Labels	State
project.pj					
Source\project.pj					
AFR\project.pj					
al\project.pj					
CD\project.pj					
cfg\project.pj					
ECUAL\project.pj					
MCAL\project.pj					
MPTLIB\project.pj					
RTE\project.pj					
AI\project.pj					
OSA\project.pj					
opt\project.pj					
src\project.pj					
osa_10ms.c	1.12	1.13			planned
osa_2ms.c	1.18	1.19			planned
osa_init.c		1.13			planned
osa_shutdown.c		1.3			planned
osa.c		1.1		[QCU_ISW_O100] [QCU_IS...	planned
osa.h		1.2		[QCU_ISW_O100] [QCU_IS...	planned
component.def		1.8		[QCU_ISW_O100] [QCU_IS...	planned
BSI\project.pj					
ANI\project.pj					
opt\project.pj					
src\project.pj					
bsi_ani_Cfg.h		1.1		[QCU_ISW_O100] [QCU_IS...	planned
bsi_ani.h		1.8			planned
bsi_current.c	1.7	1.8			planned
bsi_pressure.c		1.4			planned
bsi_temperature.c		1.5			planned
bsi_voltage.c	1.6	1.7			planned
component.def		1.3		[QCU_ISW_O100] [QCU_IS...	planned
DI\project.pj					
SIC\project.pj					
BSO\project.pj					
DO\project.pj					
opt\project.pj					
src\project.pj					
bso_do_Cfg.h		1.1		[QCU_ISW_O100] [QCU_IS...	planned
bso_do.c		1.3			planned
bso_do.h		1.2		[QCU_ISW_O100] [QCU_IS...	planned
component.def		1.2		[QCU_ISW_O100] [QCU_IS...	planned
PWM\project.pj					
SOC\project.pj					
com\project.pj					
SIP\project.pj					
SL\project.pj					
Test\project.pj					

Fig. 33 Arhitectura de fișiere a proiectului

## 6. Testarea software-ului

Deoarece părțile software componente ale proiectului au fost dezvoltate în paralel, pentru testarea RTE-ului s-a folosit o simulare a librăriei FSW-ului. Pe baza specificațiilor primite din partea companiei Magna Powertrain cu privire la funcționalitățile FSW-ului care trebuie integrate în task-urile BSW-ului, s-a elaborat o librărie care să testeze funcționarea corectă a RTE-ului.

Această testare a funcționării corecte a RTE-ului înseamnă, de fapt, pe de-o parte, testarea faptului că datele achiziționate de către BSW de la senzori, convertoare sau porturi sunt transmise FSW-ului conform specificațiilor, iar pe de altă parte, testarea faptului că toate comenzile transmise de către FSW sunt interpretate corect, iar motorul și valvele sunt acționate în consecință.

### 6.1. Utilitare folosite

Aplicația a fost testată într-un laborator specializat din cadrul companiei Continental. Codul a fost încărcat pe un TCU propriu-zis, conectat la un diferențial, pentru testarea acționării acestuia și la un calculator, pentru a testa achiziția datelor de pe rețeaua Flexray simulată.

Datorită librăriei de testare implementate în acest scop, se pot simula diferite contexte de funcționare a TCU-ului și se pot modifica variabile în timp real pentru a determina luarea unor anumite decizii de către acesta.

Utilitarele folosite pentru simularea contextelor de funcționare dorite și pentru testarea și urmărirea funcționării aplicației sunt CANape și CANalyzer, dezvoltate de compania Vector.

Librăria elaborată pentru testare este folosită și în cadrul prezentării practice. S-a implementat o aplicație “demo” care să ruleze pe unitatea de control în așa fel încât să se observe modificarea acțiunilor comandate de software în funcție de potențialele cereri venite din partea FSW-ului. Astfel, cu ajutorul aplicației CANape se pot modifica manual valorile unor variabile cu scopul de a provoca schimbările dorite pentru a testa sau demonstra funcționarea corectă a software-ului.

## **6.2. Testarea propriu-zisă**

S-a precizat mai devreme în lucrare că pe baza cerințelor proiectului extrase din cererile clientului și din standardul AUTOSAR se vor elabora teste corespunzătoare fiecăreia din cerințe.

Așadar, s-au testat în mare, următoarele funcționalități:

- citirea datelor necesar a fi transmise FSW-ului de către RTE:
  - o citirea mesajelor și a semnalelor de pe rețeaua Flexray
  - o citirea datelor furnizate de convertoarele analog numerice
  - o citirea stărilor diferiților pini ai porturilor TCU-ului
- calculul corect al datelor ce necesită anumite conversii înainte de a fi transmise
- acționarea corectă a motorului și a valvelor
- integrarea task-urilor FSW-ului în cadrul BSW-ului

## **7. Concluzii**

### **7.1. Rezultatele proiectului**

Proiectul a fost dus la capăt cu succes, respectându-se în același timp planificarea inițială. Implementarea software a primei livrări, care urmărește integrarea BSW-ului și a FSW-ului, s-a realizat la timp, dispunând de o perioadă suficientă pentru testarea acesteia. Rezultatele testării au fost favorabile, aplicația funcționând corect, în parametrii normali, urmând a fi testată ulterior și într-o rețea reală FlexRay.

În dezvoltarea proiectului s-a urmărit parcurgerea fazelor ciclului V, după încheierea fiecărei faze realizându-se o revizie a documentelor elaborate sau a modificărilor survenite de la ultima revizie. Astfel, toate componentele proiectului, de la analiza cererilor clientului până la implementarea codului și elaborarea testelor, au fost supuse unor revizii astfel evitând apariția erorilor.

### **7.2. Direcții de dezvoltare**

Am prezentat mai devreme în lucrare că pentru prima livrare a aplicației se va urmări integrarea celor două software-uri principale (BSW și FSW) prin intermediul RTE-ului. Aceasta presupune integrarea task-urilor FSW-ului în task-urile sistemului de operare al BSW-ului și implementarea anumitor semnale de interfața vitală pentru funcționarea unității de control, cum ar fi semnale de pornire/oprire a unității sau de acționare a motorului și valvelor.

Așadar, din punctul de vedere al proiectului în ansamblu, acesta va mai fi dezvoltat funcțional pe baza cerințelor clientului. Vor mai fi implementate cerințe referitoare la aspecte legate de siguranță, memorie EEPROM și celelalte semnale nevitale care nu au fost necesare pentru prima livrare.

De asemenea, în capitolul 2 (Cerințe) am prezentat o analiză a cerințelor clientului și a cerințelor AUTOSAR, precum și diferitele variante de dezvoltare, în final alegându-se cea de mijloc: BSW variantă clasică și RTE AUTOSAR.

Deoarece tendința în domeniul automotive este de a dezvolta software care să respecte standardul AUTOSAR, o direcție principală de dezvoltare ar fi implementarea integrală a software-ului urmând cerințele standardului. Acest lucru nu va fi dificil de realizat, ținând cont de cunostințele și experiența dobândite în urma implementării AUTOSAR a RTE-ului.

### **7.3. Utilitatea proiectului**

Utilitatea proiectului consta in principal in avantajele prezentate de respectarea normelor standardului AUTOSAR. Aceste avantaje au fost prezentate mai devreme in lucrare, dintre care cele mai importante fiind posibilitatea reutilizarii componentelor si configurarea mai usoara a acestora.

De asemenea, acest proiect va sta la baza altor proiecte ce se doresc a fi implementate cu respectarea standardului AUTOSAR. Toate cunoștințele și experiența vor fi foarte utile în dezvoltarea proiectelor viitoare, ținând cont și de faptul că se dorește dezvoltarea conform AUTOSAR a tuturor componentelor unui autovehicul.

## 8. Bibliografie și studiu bibliografic

[Et. 1] Nanu Sorin - “Sisteme cu microcontrollere si microprocesoare” - notite de curs, lucrari de laborator

[Et. 2] Lacramioara Stoicu-Tivadar - “Programarea calculatoarelor” - notite de curs, lucrari de laborator

[Et. 3] <http://www.autosar.org/>

[Et. 4] <http://en.wikipedia.org/>

[Et. 4] Writing Good Requirements - document intern Continental

[Et. 5] C Coding Rules - document intern Continental

## ANEXA 1 - Lista de figuri

Fig. 1 Tipuri de tracțiune .....	7
Fig. 2 Diferențialul.....	9
Fig. 3 Eliminarea subvirajului cu ajutorul diferențialului sport.....	10
Fig. 4 Menținerea stabilității în curbe succesive cu ajutorul diferențialului sport .....	10
Fig. 5 Privire de ansamblu asupra arhitecturii AUTOSAR.....	12
Fig. 6 Arhitectura AUTOSAR în detaliu .....	14
Fig. 7 Arhitectura unui proiect din clasa de conformitate ICC1 .....	15
Fig. 8 Sistemul de transmisie al autoturismului Audi Quattro Sport .....	16
Fig. 9 Schema bloc a proiectului.....	17
Fig. 10 Motorul comandat în punte H.....	18
Fig. 11 Valva comandată în semipunte.....	19
Fig. 12 Arhitectura simplificată a sistemului software .....	21
Fig. 13 Arhitectura detaliată a sistemului software.....	22
Fig. 14 Funcționarea proiectului deja existent (QHCU) .....	25
Fig. 15 Diferențe între proiectul vechi și cel nou.....	26
Fig. 16 Funcționarea proiectului deja existent (QHCU) .....	26
Fig. 17 Schimburile de informații ce intervin în funcționarea proiectului.....	28
Fig. 18 Diagrama Gannt a planificării activității .....	32
Fig. 19 Arhitectura proiectului cuprinsă în cererile clientului .....	35
Fig. 20 Cerințele standardului AUTOSAR luate în considerare.....	39
Fig. 21 Exemplu de cerințe introduse în aplicația Rational DOORS.....	45
Fig. 22 Componenta RTE-ului.....	47

Fig. 23 Interacțiunea dintre BSW și FSW.....	48
Fig. 24 Schema logică de funcționare a task-ului de inițializare .....	49
Fig. 25 Schema logică a funcționării task-ului de 2 milisecunde .....	50
Fig. 26 Schema logică a funcționării task-ului de 10 milisecunde .....	51
Fig. 27 Schema logică a funcționării task-ului de shutdown .....	52
Fig. 28 Componenta modulului de achiziție a datelor .....	52
Fig. 29 Funcționarea achiziției de date .....	53
Fig. 30 Componenta modulului de distribuție a datelor.....	54
Fig. 31 Funcționarea distribuției de date.....	54
Fig. 32 Componenta BSW-ului.....	55
Fig. 33 Arhitectura de fișiere a proiectului .....	57