Titlu Proiect: VIrtual Platform for real time testing of electric vehicles with improved EneRgetic performances

Nr. contract: BG 38/2016

Adresă website: www.viper.utcluj.ro

Organism implementare: Universitatea Tehnică din Cluj-Napoca

Beneficiar cercetare: Siemens Industry Software SRL (Brașov)

Echipă proiect:

Daniel FODOREAN (director)
Lorand SZABO (cercetător)
Ioana-Cornelia GROS (cercetător)
Claudia Violeta POP (doctorand)
Tamas GYORGY (doctorand)
Alexandru-Mihai DĂRĂMUȘ (doctorand)
Cătălin Rareș NACU (masterand)
Adam KIS (masterand)

Raport de activitate 2017

Realizarea interfeței virtuale și a modelelor subcomponente

CUPRINS

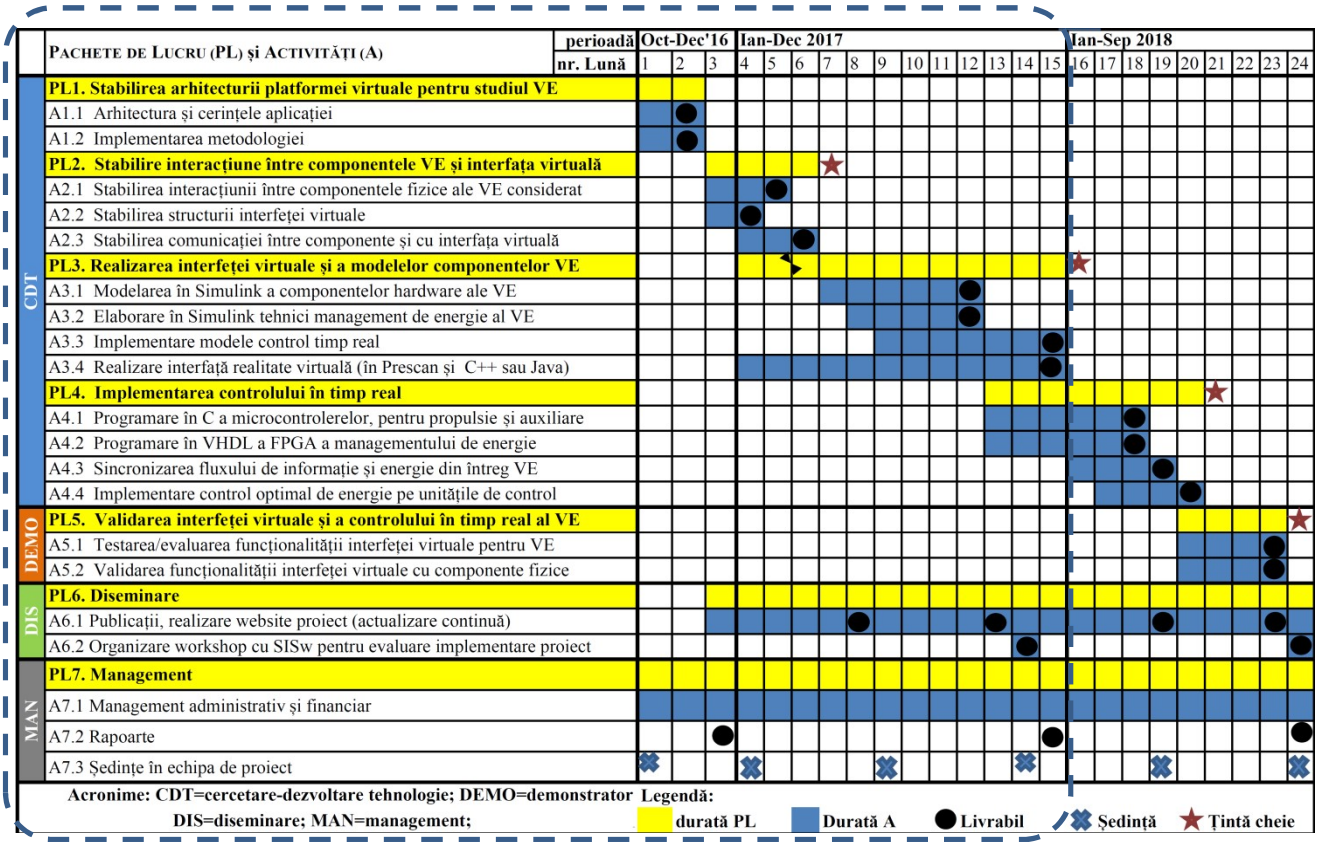
I. Rezumatul Etapei

1. Obiectivele cercetării din etapa a II-a de implementare a proiectului VIPER

ETAPA a II-a: Realizarea interfeței virtuale și a modelelor sub-componente
II.1 Stabilirea interacțiunii între componentele VE și interfața virtuală
II.2 Realizarea interfeței virtuale șia modelelor componentelor VE

2. Preambul al cercetării pe anul 2017 implementate în proiectul VIPER

Activitatea de cercetare în 2017 s-a derulat între lunile a 4-a și a 15 de implementare a proiectului VIPER, Pentru evaluarea stadiului implementării cercetării se vor readuce în discuție activitățile de întreprins specificate la propunerea de proiect în diagrama Gantt:



Obiective și activitățipropuse în proiectul VIPER pe durata primelor 15 Luni de derulare.

Realizările principale ale cercetării în ultimul an de derulare vor fi detaliate în continuare. Aici se prezintă doar rezumatul acestora:

- S-au elaborat modele matematice pentru componente de tracțiune și direcție electrică, implementate în Simulink, precum și convertizoarele statice asociate pentru alimentarea mașinilor electrice și a regulatoarelor necesare controlului.
- S-au realizat circuite rutiere virtuale, folosind automobile și scutere folosind software-ul Prescan
- S-a realizat conexiunea între Simulink și softwareul Prescan de realitate virtuală pentru pregătirea conexiunii cu platforme de timp real.
- S-a realizat o noua interfață de realitate virtuală, care să integreze modelul mecanic impus de configurația drumului (inclusiv elevație) și modelul propulsiei electrice, programarea acesteia

realizându-se în Simulink. Aici s-au considerând circuite rutiere urbane din Cluj-Napoca și Brașov.

- Ambele platforme, atât cea elaborată în Prescan cât și cea nou realizată au fost testate cu sistem volan/pedalier achiziționat în proiect.
- Pentru caracterizarea (dezvoltarea adecvată de modele) a bateriei, s-au achiziționat celule de baterie de tip LiFePO4, precum și circuite de echilibrare aferente, precum și sisteme de monitorizare a consumului energetic (acestea fiind principalele achiziții ale proiectului VIPER).
- S-a realizat programarea unei unități FPGA pentru implementarea modelului propulsiei, model folosit și în software-ul Prescan, dar și în a doua platformă de realitate virtuală (nou elaborată). Astfel, s-a putut evalua funcționarea în timp real a modelului de control al propulsiei electrice.
- S-a realizat interfațarea hardware a două sisteme de propulsie: unul de mare viteză, iar al doilea în structură inversată (motor roată), precum și interfața corespunzătoare pentru detecția vitezei cu traductor de tip resolver.
- Cu ajutorul membrilor companiei SISw s-a elaborat o platformă de simulare a propulsiei electrice folosind software-ul companiei, SISw, ce se dorește a fi integrat în platforma de timp real în vederea evaluării bilanțului energetic în condiții cât mai aproape de realitate a aplicației studiate.
- S-au efectuat două stagii la compania SISw (al treilea programat la depunerea proiectului urmând a se derula în luna Decembrie, dar la Cluj, când membri companiei SISw vor dori să evalueze și testeze platformele virtuale obținute și să evalueze posibilitatea conectării acestora la o aplicație de tip hexapod dezvoltată de companie, pentru simularea condițiilor de rulare a vehiculelor)
- S-au publicat 3 articole de conferință indexate IEEE, dintre care unul în colaborare cu membrii SISw.

S-a început și partea de implementare a funcționării în timp real a platformei virtuale, subiect pe care va fi concentrată activitatea de cercetare în ultima etapă a proiectului VIPER. În continuare se vor prezenta mai pe larg realizările prezentate mai sus.

II. Implementarea activităților de cercetare în proiectul VIPER

1. Dezvoltarea platformei virtuale folosind software-ul Prescan

Prescan, un produs al companiei TASS International din Olanda a ajuns să fie achiziționat, în luna August a.c., chiar de către Siemens, urmând a fi integrat în portofoliul companiei SISw. Această conjunctură favorabilă va permite, în viitor (specificîm acest lucru, deoarece procedura de integrare a Tass International în ceea ce înseamnă afacerea Siemens este un proces ce se va intinde, se pare pe durata a 18 luni), o mai bună integrare a produselor companiei SISw în platforma de realitate virtuală ce se dorește a fi elaborată în proiectul SISw. În continuare se va detalia modul de lucru cu Prescan și realizările cercetării implementate.

Prin intermediul programului *Prescan*, putem proiecta/importa un scenariu de condus care să surprindă elemente de infrastructură, marcaje rutiere, condiții meteo, actori (autovehicule, pietoni) și senzori, iar toate aceste se pot mai apoi simula. Astfel, putem analiza comportamentul autovehiculelor din punctul de vedere al propulsiei, dinamicii și sistemelor de siguranță. Capacitatea de a cosimula cu *Matlab-Simulink* îi conferă un atu în implicarea acestuia în platfroma de simulare, unde Prescan este responsabil cu susținerea realității virtuale. Transferul de date dintre *Prescan* și *Matlab-Simulink* ține strict de poziționarea GPS a autovehiculului în realitatea vrituală, elementele de dinamică, propulsie, comunicare senzori, simulându-se în *Simulink*. Deoarece acumulatorii autovehiculelor electrice reprezintă o componentă fundamentală în sistemul de propulsie, o strategie cat mai eficientă de BMS (baterry management system) este prioritară a fi implementată. În acest sens, modelarea/simularea BMS și a acumulatorilor se va dezvolta în programul AMESIM, un produs al companiei SISw care permite și conexiunea cu Simulink. Vizual, cele de mai sus se pot prezenta astfel, ca în Fig.1 :

Fig. 1 Fluxul de date al cosimulării folosind Prescan, Simulink și Amesim.

Spre a atinge obiectivul cosimulării între cele trei programe, s-au parcurs următoarele etaăe. Primul pas a fost realizarea unui scenariu care conține un circuit pe care să se modeleze și simuleze comportamentul unui vehicul și conceptul de cosimulare. Pentru a replica condițiile cât mai real posibil, s-a introdus un HID (Human Interface Device, Fig 2b), de tip volan-pedalier, cu care s-a controlat accelerația, frânarea, direcția și cutia de viteze a vehiculului. S-a rulat cosimularea și răspunsul în realitatea virtuală la manevrele exercitate la HID a fost prompt.

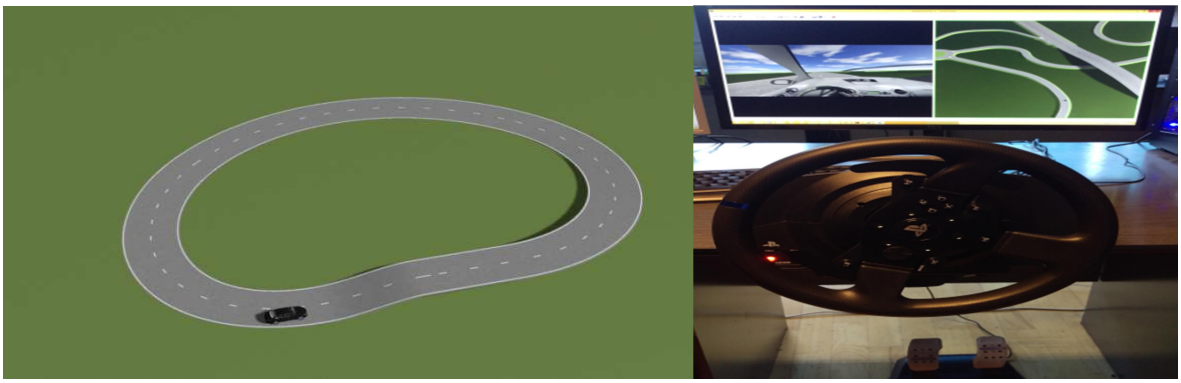


Fig. 2 a) Primul circuitul de test; b) conducere cu volan și pedalier (de tip HID).

Următorul pas a fost să dezvoltăm un circuit amplu (aprox 1km^2), care să conțină mai multe categorii de drum cum sunt: alei urbane, regionale, autostrăzi, și să fie implicate marcaje rutiere, actori, V2I (Vehicle to Infrastructure Interaction) după cum se poate observa in figura Fig. 3. Datorită complexității modelului, timpul de cosimulare este sporit, cu toate că resursele calculatorului sunt utilizate în proporție de doar 20%. Prin urmare, este necesară o metodă de accelerare a simulări pentru cazurile în care se dorește cosimularea unui model mai complex. Totodată, vom fi dornici să reținem această limitare în perspectiva realizării propriei interfețe virtuale de co-simulare!

Înainte de a soluționa problema co-simulării lente pentru modele complexe, datorită limitărilor în răspuns a sistemului (cu toate că PC-ul pe care este instalat Prescan este cu procesor i7-octocor, cu 64Gb RAM, HDD dublu, adică unul rapid pentru sistemulde operare, iar placa video este de top, cu 2Gb memorie), s-a revenit la cosimulare pe circuitul din Fig. 2a). S-a introdus un model de propulsie cu mașină sincronă cu magneți permanenți în modelul autovehiculului din Simulink, având ca referință de viteză pedala de accelerație a HID iar Mr (cuplu rezistent) aplicat la arbore a fost calculat din forțele de rezistență la rulare. S-a rulat simularea, la un pas corespunzător de simulare (10 kHz) - cu observația că la circuitul rutier mai complex simularea ruleaza foarte lent la această frecvență de eșantionare. Am făcut un compromis pentru a putea observa comportamentul modelului de propulsie. În Fig.4 este prezentat subsistemul de cosimulare elaborat în Simulink.

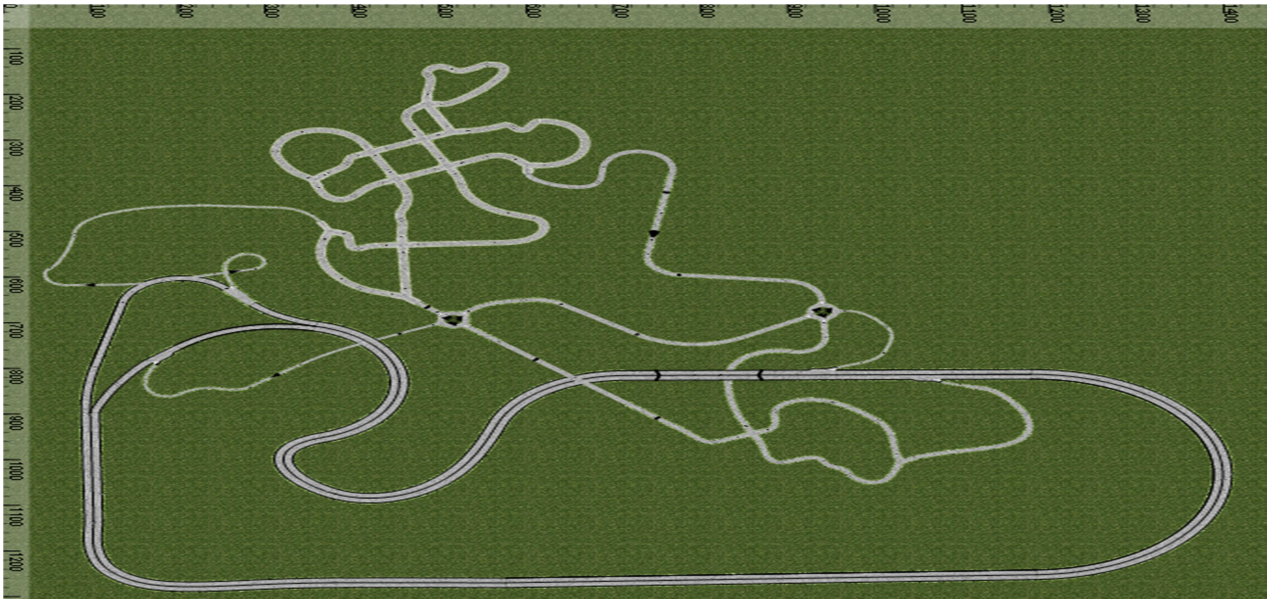


Fig. 3 Circuitul complex pentru testare

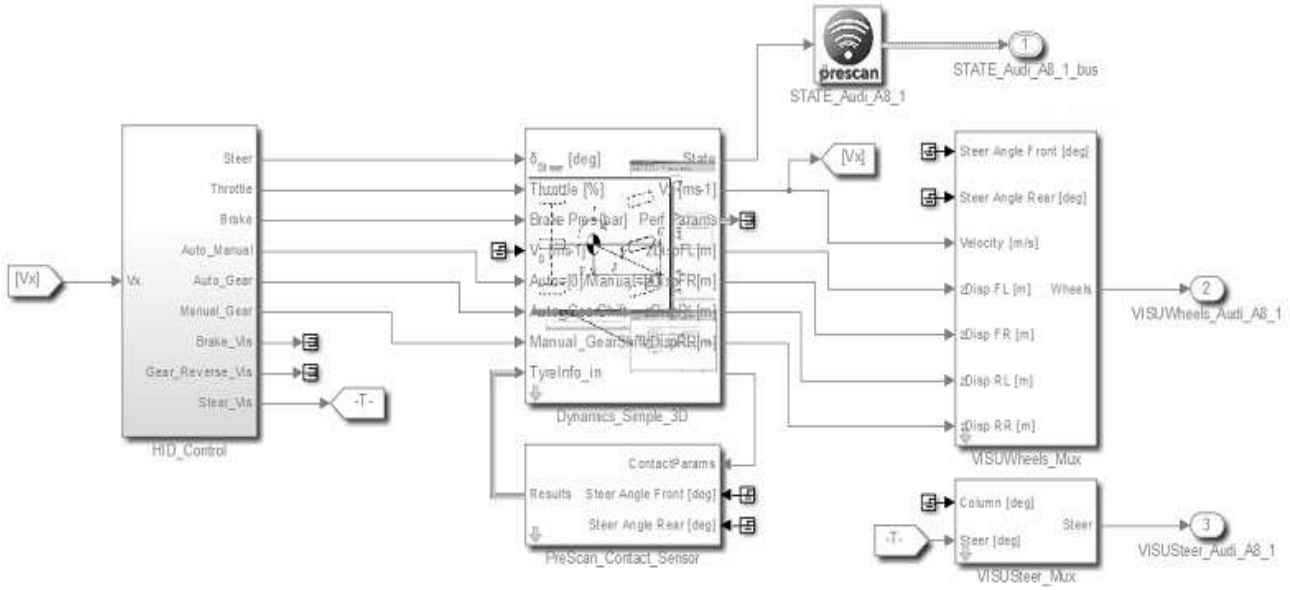


Fig. 4 Modelul autovehiculului în *Simulink.*

Vehiculul simulat este ușor, de tip scuter sau motocicletă. Mas vehiclului, cu conducător, a fost considerată de 200 kg. (Pentru acest caz, în momentul rulării cosimulării, unghiul Φ de înclinare transversală a vehiculului variază cu frecvență de 5Hz între valorile ±60°, ceea ce este ireal. Un motiv posibil pentru această variație este neadaptarea momentelelor de inerție (Jzz, Jyy, Jxx) pe cele 3 axe). Ca și exemplificare, se prezintă modalitatea de parametrizare a scuterului electric în pregătirea co-simulării Prescan-Simulink, vezi Fig. 5.

Pașii următori constau în calcularea momentelor de inerție, îmbunătățirea vitezei de cosimulare și începerea dezvoltării strategiei de BMS în programul AMESIM. După ce toate acestea se vor finaliza, modelele vor fi încărcate pe platforme de dezvoltare pentru a putea rula *Real Time.*

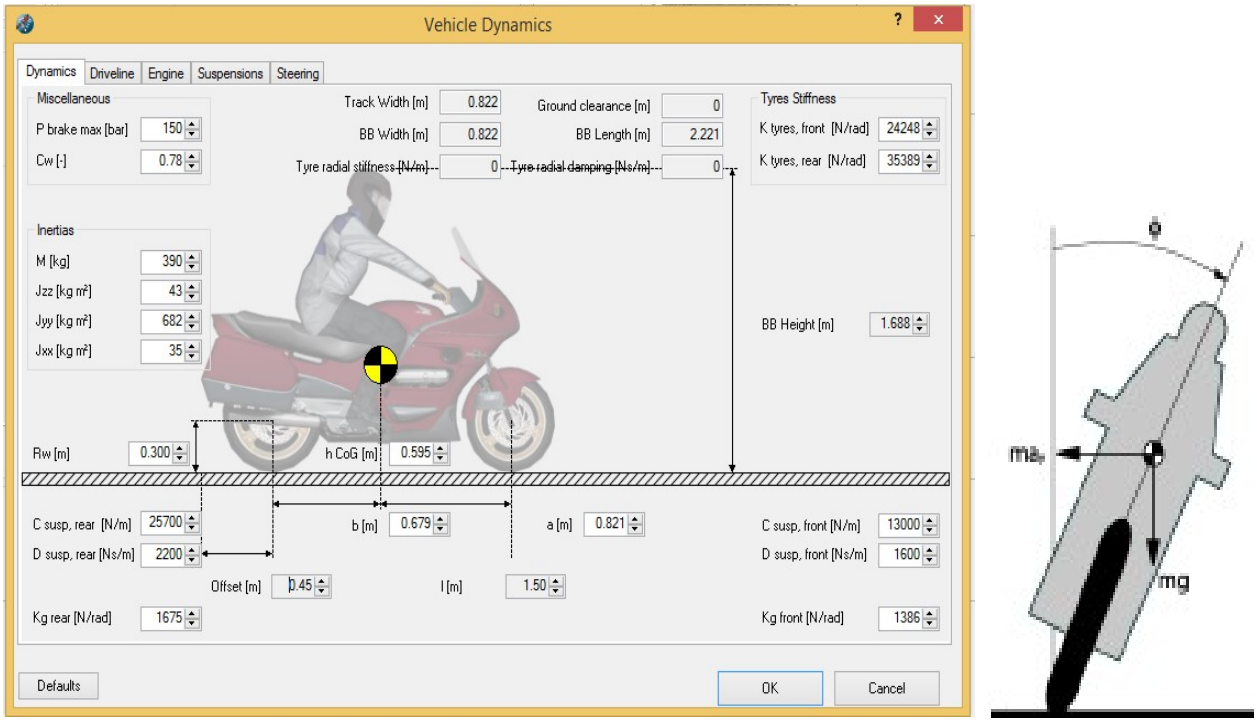


Fig. 5 Parametrizarea condițiilor de rulare ale motociceltei (scuterului) electric.

2. Realizarea unei noi interfețe virtuale, dezvoltată de membrii proiectului Viper

Pentru realizarea sistemului de software de elaborat este necesară folosirea câtorva programe, care realizează câte un task separat, respectiv este necesară interconectarea acestora folosind librării existente sau prin protocoale de comunicație bine definite.

După o analiză consistentă a fost creată arhitectura software-ului, prezentată în Fig.6. Pentru a crea o interfață nouă de realitate virtuală, a trebuit să se găsească o platformă de dezvoltare 3D. Cum reiasă și din Fig. 6, comunicarea este bidirecțională între utilizator și software-ul de realitate virtuală, inputul spre VR este un volan cu pedale sau direct claviatura, outputul fiind sunetul și realitatea creată pe monitor sau cască audio.

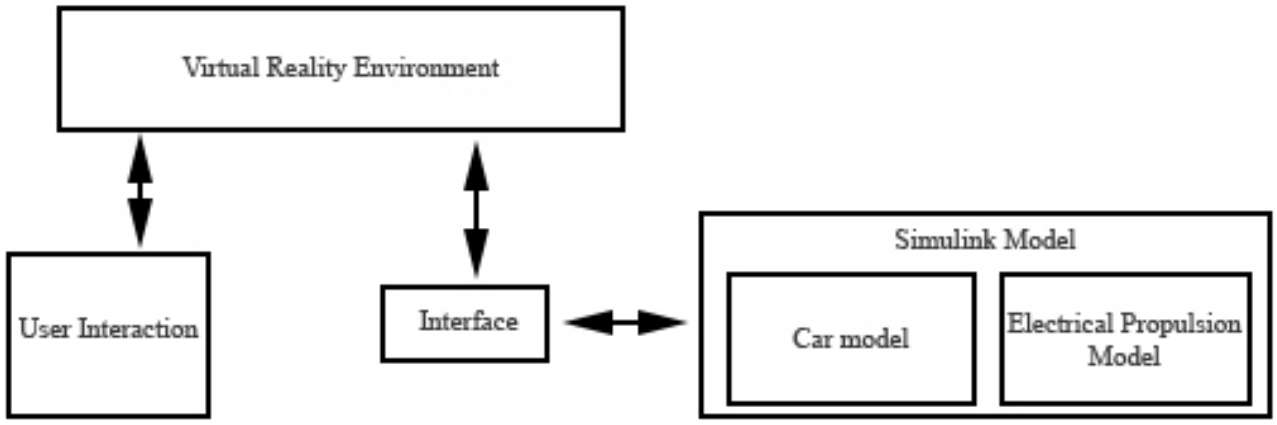


Fig. 6. Arhitectura software-ului de simulator propulsie electrică.

Între VR și simulare este necesară o interfață prin care se realizează comunicarea bidirecțională. De la VR înspre simulator sunt trimiși parametri ca și poziția pedalei, înclinația mașinii față de orizontală, respectiv parametrii incintei în care se află mașina simulată (condiții de drum, calitatea drumului). Din direcția simulatorului înspre VR sunt transmise viteza și accelerația mașinii, respectiv starea de încărcare a bateriilor.

Modelul mașinii și al controlului acesteia au fost realizate folosind Matlab Simulink. Arhitectura acestuia este compus din două clase, unul fiind modelul matematic a mașinii, celălt fiind modelul de propulsie în care este încorportată bateria, algoritmul de comandă și control, respectiv motorul electric. Comunicarea între cele două module este realizată intern, folosind interfețele de Simulink predefinite.

Realitatea virtuală referă la ambianțe artificial create pe calculator care oferă o simulare a realității atât de reușită, încât utilizatorul poate căpăta impresia de prezență fizică aproape reală, atât în anumite locuri reale, cât și în locuri imaginare. Pentru realizarea acestei impresii, am ales să folosim platforma de dezvoltare Unity 3D pentru simplitatea conexiunii cu platofma de realitate virtuală. Pentru definirea VR-ului folosind Unity3D este necesar definirie claselor care cum ar fi clasa de *mașină*, clasa de *hartă* respectiv trebuie adăugată și poziția de unde este văzut obiectul respectiv de utilizator.

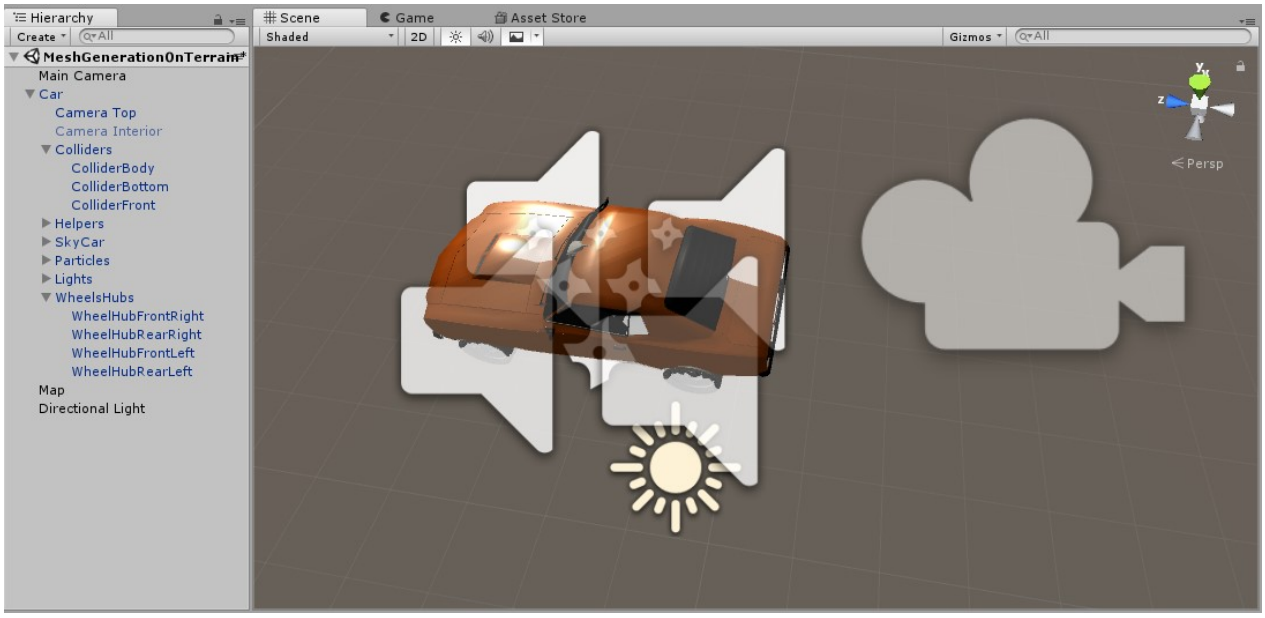


Fig. 7 Obiectele folosite în realitatea virtuală.

După cum reiese din Fig. 7, sunt folosite niște subclase moștenite de la clasa mamă, care predefinesc comportamentul obiectului în lumea virtuală generată. Clasa Car are ca și subclase două camere, cu diferite poziții anume, trei clase de Colliders care definește comportamentul mașinii în lumea virtuală, cum ar fi reacția mașinii la ciocniri frontale, să ruleze pe roți si nu pe șasiu, etc. Așadar venicului este compus din mai multe elemente, cum ar fi șasiul, roțile, habitaclul mașinii și motorul, fiecare având parametrii de sunet și lumină predefinit. Ca și exemplu, pentru fiecare roată sunt adăugate sunetele specifice în funcție de direcția în care rulează mașina respectiv inputul de la utilizator. La accelerare se aude zgomotul anvelopelor (zgomotul motorului electric este nesemnificativ), la o frânare bruscă se aude un zgomot specific a frânării.

Pentru a realiza interacția cu volan/claviatură, s-a elaborat un algoritm, care în funcție de input, returnează o variabilă între [0..1] de tip float, prin care este determinată poziția actuală a volanului/pedalei în funție de extremele acestora. Algoritmul respectiv este prezentat pe Fig. 8.

```
using System;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

namespace UnityStandardAssets.Vehicles.Car
{
    [RequireComponent(typeof (CarController))]
    public class CarUserControl : MonoBehaviour
    {
        private CarController m_Car; // the car controller we want to use


        private void Awake()
        {
            // get the car controller
            m_Car = GetComponent<CarController>();
        }


        private void FixedUpdate()
        {

            // pass the input to the car!
            float h = CrossPlatformInputManager.GetAxis("Horizontal");
            float v = CrossPlatformInputManager.GetAxis("Vertical");
#if !MOBILE_INPUT
            float handbrake = CrossPlatformInputManager.GetAxis("Jump");
            m_Car.Move(h, v, v, handbrake);
#else
            m_Car.Move(h, v, v, 0f);
#endif
        }
    }
}
```

Fig. 8. Algoritm pentru realizarea conexiunii între volan/claviatură și VR.

Ca să fie cât mai real studiul propriuzis, s-a folosit librăria MapBox din Unity 3D. Librăria respectivă conține o interfață între Unity3D respectiv OpenStreetsMap, o hartă similară cu Google Maps. Pentru generarea hărții, într-o primă fază se determină poziția vehiculului pe hartă, după care se stabilește originea harții în funcție de coordonatele GPS, mărimea hărții și librăria Map din Unity care defapt este clasa de interfață a MapBox-ului. Harta generată în acest mod va fi una plană (2D0, fără modelele clădirilor incorporate în aceasta. Pentru a adăuga clădirile, respectiv să fie considerată și direcția verticală pe hartă, trebuie folosit clasa TerrainFactory din Unity. Generarea hărții este prezentată în Fig. 9 iar lucrul în modulul Terrain este prezentat în Fig. 10.

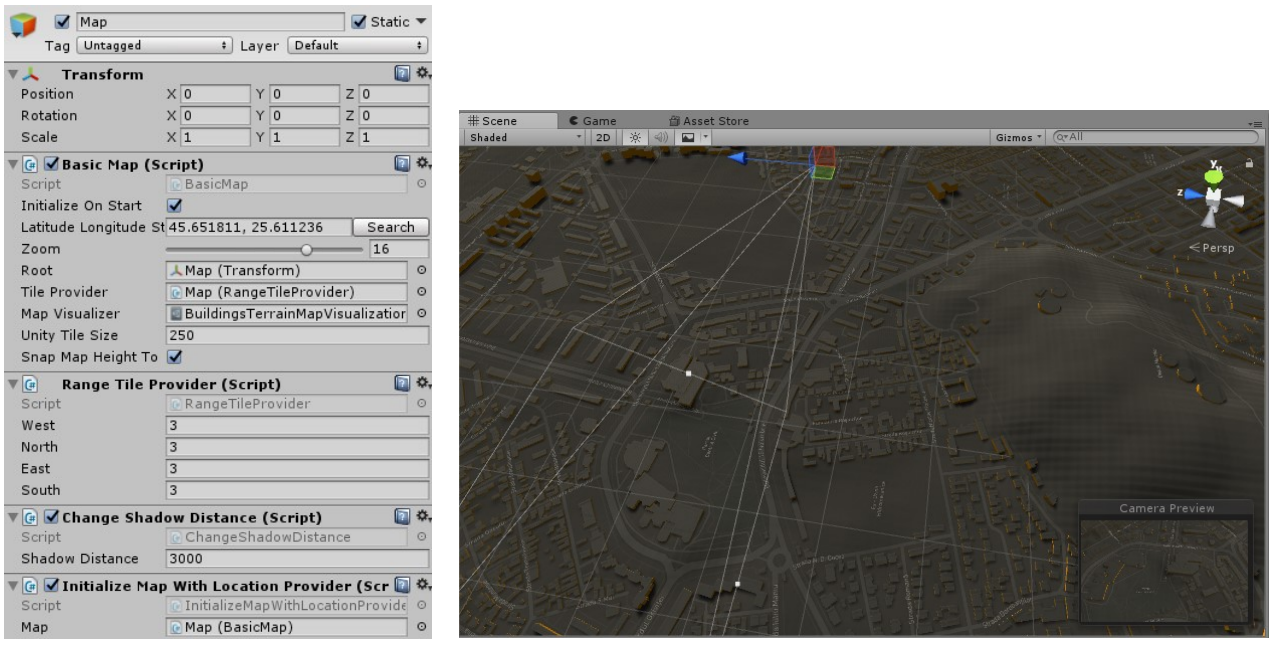


Fig. 9. Interfața de generarea ale hărții.

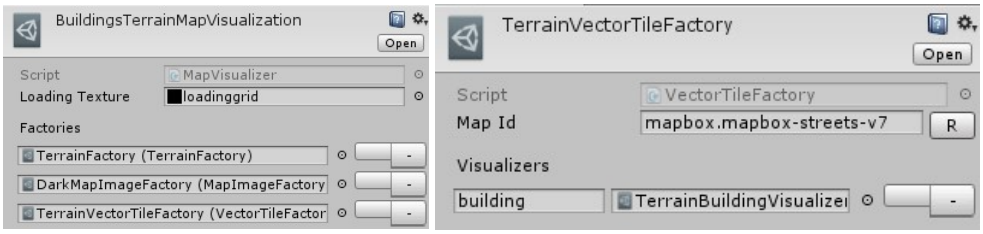


Fig. 10. Realizarea locației folosind clasa *TerrainFactory*.

După setarea tipului de hartă folosit, respectiv fațada, înălțimea și materialul, se generează harta, vezi Fig.11 unde este prezentat centrul orașului Brașov.

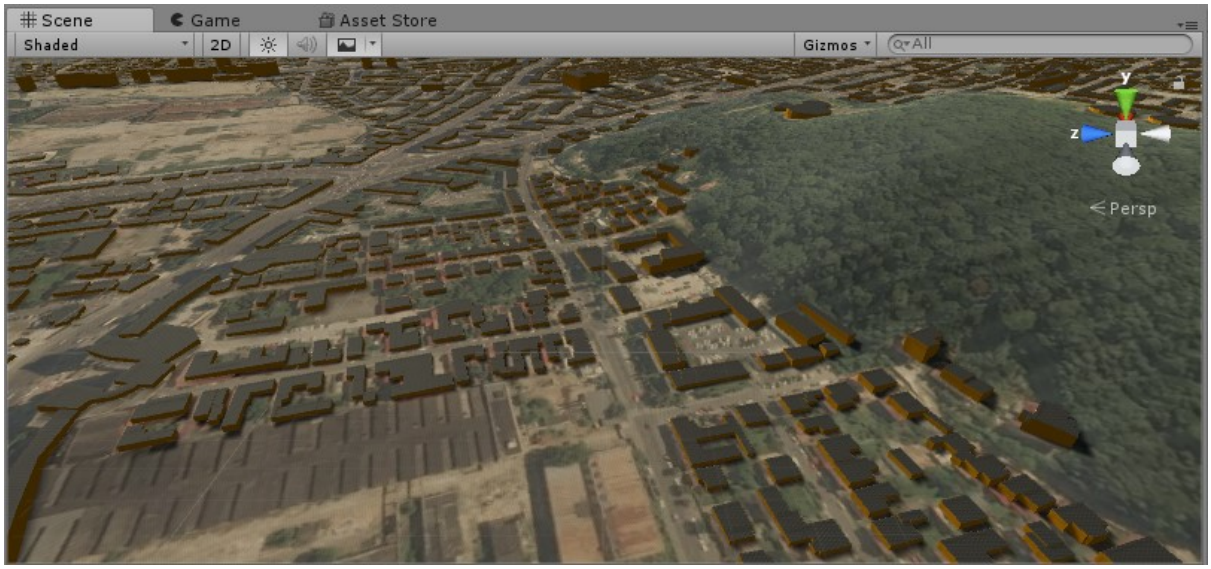


Fig. 11. Incinta generată de VR în care se deplasează mașina – oraș Brașov, coordonate GPS: 45.651811, 25.611236.

Pentru o funcționare a sistemului compus, este necesară realizarea/folosirea unei interfețe de comunicare prin care Unity3D și Matlab/Simulink pot comunica. Pentru acesta s-a făcut o analiză prin care a avut ca și rezultat că protocolul de comunicație *TCP/IP* sau *Shared Memory* trebuie folosit.

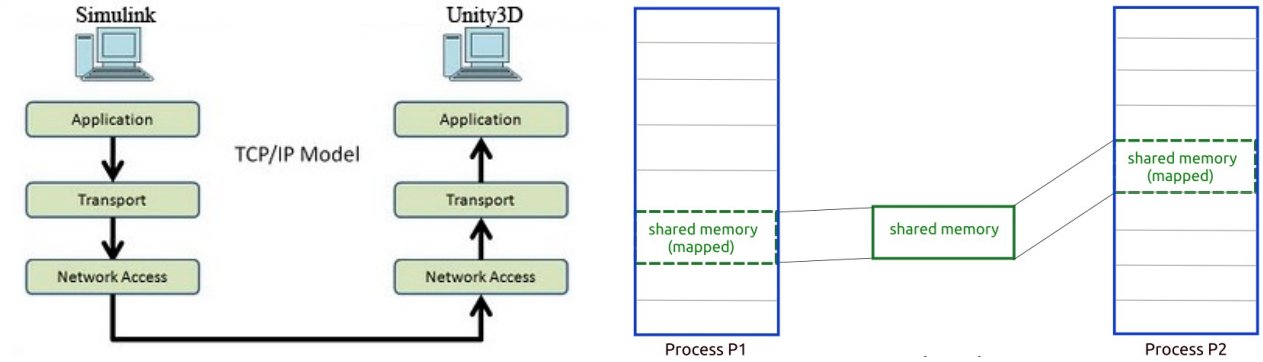


Fig. 12. Protocolul de *TCP/IP* respectiv descrierea funcționalității *Shared memory.*

Diferențele principale între TCP/IP și Shared Memory sunt enumerate mai jos:

- TCP/IP poate să facă conexiunea între două aplicații instalate pe aceeași PC sau PC-uri diferite. Acest lucru este imposibil de folosit cu memoria comună.

- Există deja protocolul de TCP/IP în Unity3D respectiv Simulink, librăriile de bază pentru memorie comună trebuie scrise.
- Timpul de scriere/citire pentru memorie comună este aproximativ 0.1 ms față de 20-50 ms în cazul TCP/IP.

În faza curentă a proiectului comunicarea de TCP/IP a fost aleasă deoarece și placa de dSpace pe care se va implementa comanda în timp real a hardware-ului are protocolul de TCP/IP. Pentru realizarea conexiunilor este nevoie un *server* și un *client*, în cazul de mai sus serverul este software-ul VR iar clientul este Matlab/Simulink. Pentru realizarea serverului, trebuie creat un obiect nou, legat la o cameră „ascunsă", respectiv un modul de comunicare trebuie adăugat pentru obiectul *Car*. Codul prin care este realizată comunicarea este prezentată în Fig. 13. Partea de Matlab, are un *Listener*, care la momentul în care intră un nou bloc de date, întrerupe simularea, citește valorile a vitezei , a accelerației și a memoriei din Simulink, modifică valorile de intrare în Simulink conform datelor obținute din Unity și repornește procesul de simulare.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityStandardAssets.CrossPlatformInput;

public class VehicleSpeed : MonoBehaviour {

    public Rigidbody rb;
    public string speedStr;
    public float speed;

    /* Start listening to server data. */
    void Start()
    {
        rb = GetComponent<Rigidbody>();
    }

    void Update()
    {
        /* Get speed data from the Simulink model */
        speedStr = GameObject.Find("Main Camera").GetComponent<socketScript>().serverSays;
        speed = float.Parse(speedStr, System.Globalization.CultureInfo.InvariantCulture.NumberFormat)/3.6f;
        rb.velocity = rb.transform.forward * speed;
    }
}
```

Fig. 13. Interfața de comunicare, Unity3D.

```
while(1)
    set_param('Motor_Model','SimulationCommand','pause');
   % get_param('Motor_Model','AAA',
   % get_param
    data = membrane(1);
    rawwData = fscanf(tcpipServer);
    intermediate = strsplit(rawwData,'|');
    height1 = intermediate(1,1);
    accel1 = intermediate(1,2);
    accel = str2double(accel1);
    height = str2double(height1);
    if (height > 180.0)
        height = height - 360.0;
    end
    angle = degtorad(height);
    set_param('Motor_Model/Motor_Controller/Gain','Gain', 'accel')
    set_param('Motor_Model/Drag_forces/Gain','Gain', 'angle')
    rto = get_param(gcb,'RunTimeObject');
    speed = rto.OutputPort(1).data;
    speed = round(speed,2);
    speed1 = sprintf('%f', speed);
    fwrite(tcpipServer, speed1);
    set_param('Motor_Model','SimulationCommand','continue');
    pause(1);
end
```

Fig. 14. Codul Matlab pentru citire/scriere parametrii Simulink și Unity.

Elaborarea modelelor de comandă și a componentelor sistemului de antrenare în Simulink este justificată prin rolul pe care îl joacă acest program: se conectează cu Prescan sau software-ul nou elaborat

în proiect pentru simularea comportamentului sursei și consumatorilor la bordul unui vehicul electric ce rulează în mediul de realitate virtuală; în al doilea rând permite conexiunea cu software-ul Amesim al companiei SISw unde bilanțul energetic este evaluat și se pot asocia elemente vibro-acustice care să însoțească simularea; și în cele din urmă permite programarea și conectarea cu platforme de calcul în timp real ca dSPACE și placa de dezvoltare Anvyl, cu Spartan 6, utilizată în proiect. În continuare vom prezenta platforma de co-simularea Amesim-Simulink dezvoltată în proiectul VIPER.

3. Realizarea unei platforme de co-simulare Amesim-Simulink

Interfețele Amesim - Simulink și Simulink - Amesim fac posibilă simularea unui ansamblu complex de antrenare a unui vehicul oarecare. Deoarece există două pachete software implicate, interfețele oferă două opțiuni principale: importul modelului Amesim în Simulink sau invers. Alegerea interfeței necesare depinde de multe criterii. Există sisteme Amesim care sunt dificil sau imposibil de rezolvat folosind soluția Simulink. În acest caz, utilizatorul are două opțiuni: utilizarea co-simulării (în Simulink) sau importarea modelului Simulink în Amesim și utilizarea solverului Amesim pentru evaluarea numerică a performanțelor sistemului. Uneori, sistemul Simulink este foarte complex și importul în Amesim este, probabil, nu cea mai bună soluție. Pe lângă motivele tehnice, trebuie să luăm în considerare și modelul pentru care urmează să fie utilizat. Dacă scopul principal este de a testa / dezvolta modelul Amesim, cea mai bună strategie ar putea fi de a importa modelul Simulink în Amesim. Dacă scopul principal este de a testa / dezvolta un controler în Simulink cu un model fizic scris în Amesim, o idee mai bună ar fi să lucrăm în Simulink. Fig. 15-sus principiul co-simulării: Amesim ca slave și Simulink ca master, program elaborat de membri partenerului industrial, SISw. Conținutul simulării în Amesim este prezentat în Fig.16, unde avem inputul șoferului (accelerația și frâna), propulsia electrică, diferiți consumatori auxiliari (direcție, compresor, frână) și dispozitivele aferente de alimentare, transmisia și alte elemente sunt introduse. Modelul controlerului elaborat în Simulink este prezentat în Fig. 17, iar performanțele electromecanice (puterea la arborele motorului de antrenare și la roata vehiculului, viteza motorului de propulsie la ax și la roata vehiculului, tensiunea de alimentare și curentul absorbit) și termo-energetice (bilanțul termic în lanțul de antrenare, starea de încărcare a bateriei de alimentare, randamentul întregului sistem) sunt prezentate, pentru un ciclu rutier prestabilit, în Fig. 18.

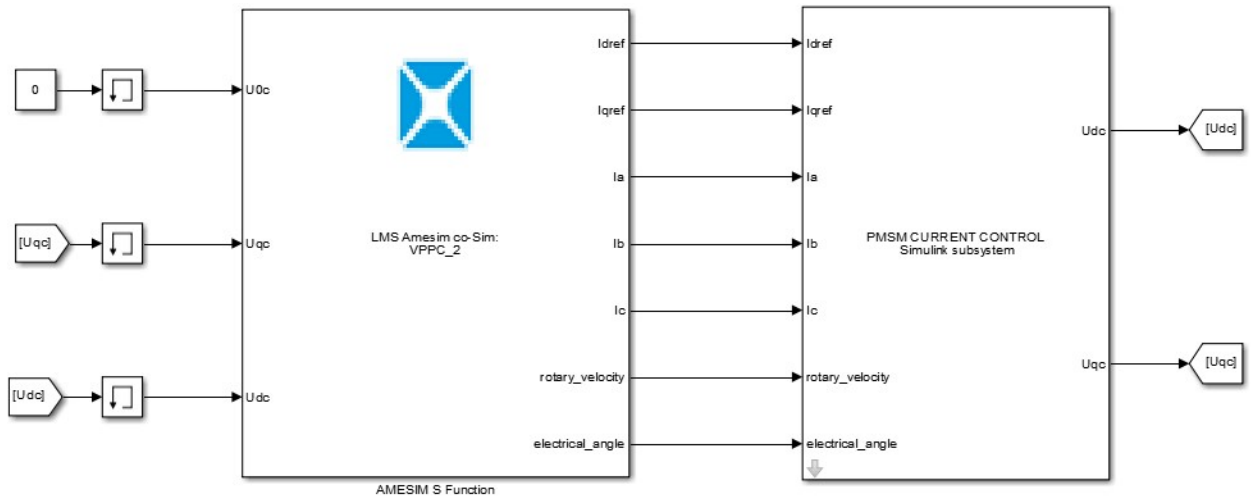


Fig. 15. Platforma de lucru în Simulink, folosind co-simularea cu Amesim, software-ul companiei SISw.

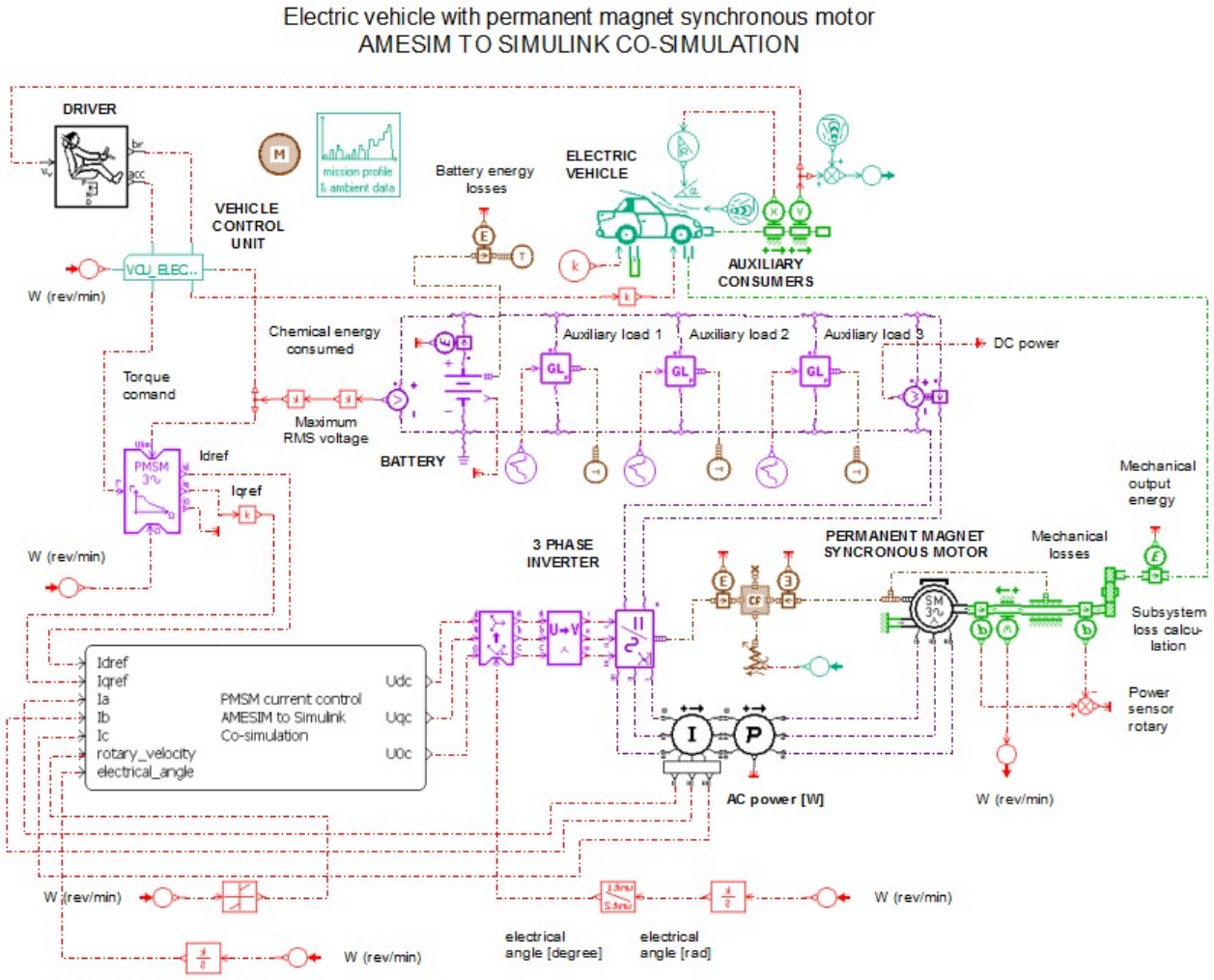


Fig. 16. Modelul Amesimi pentru modelarea vehiculului electric din perspectiva conexiunii cu Simulink.

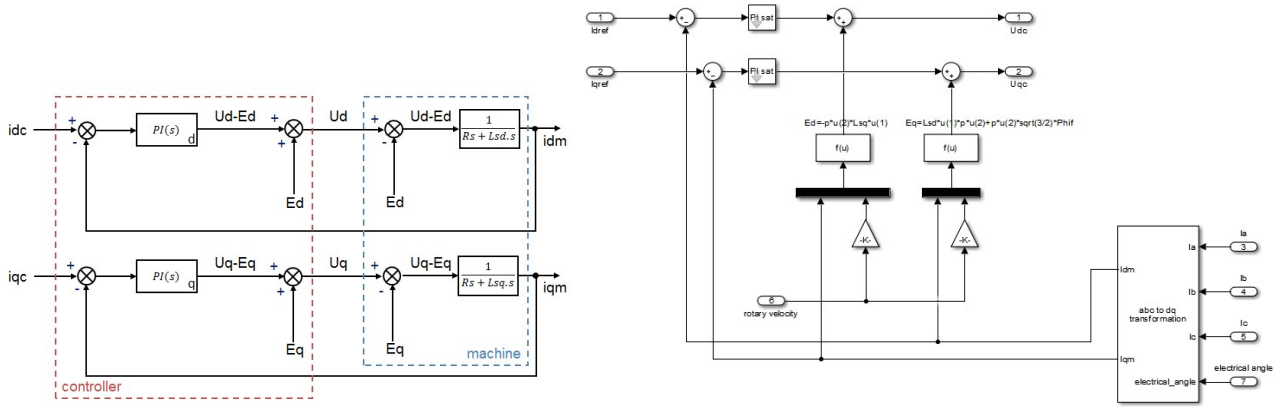


Fig. 17. Principiul controlerului PI (stânga) și elaborarea acestuia în Simulink (dreapta).

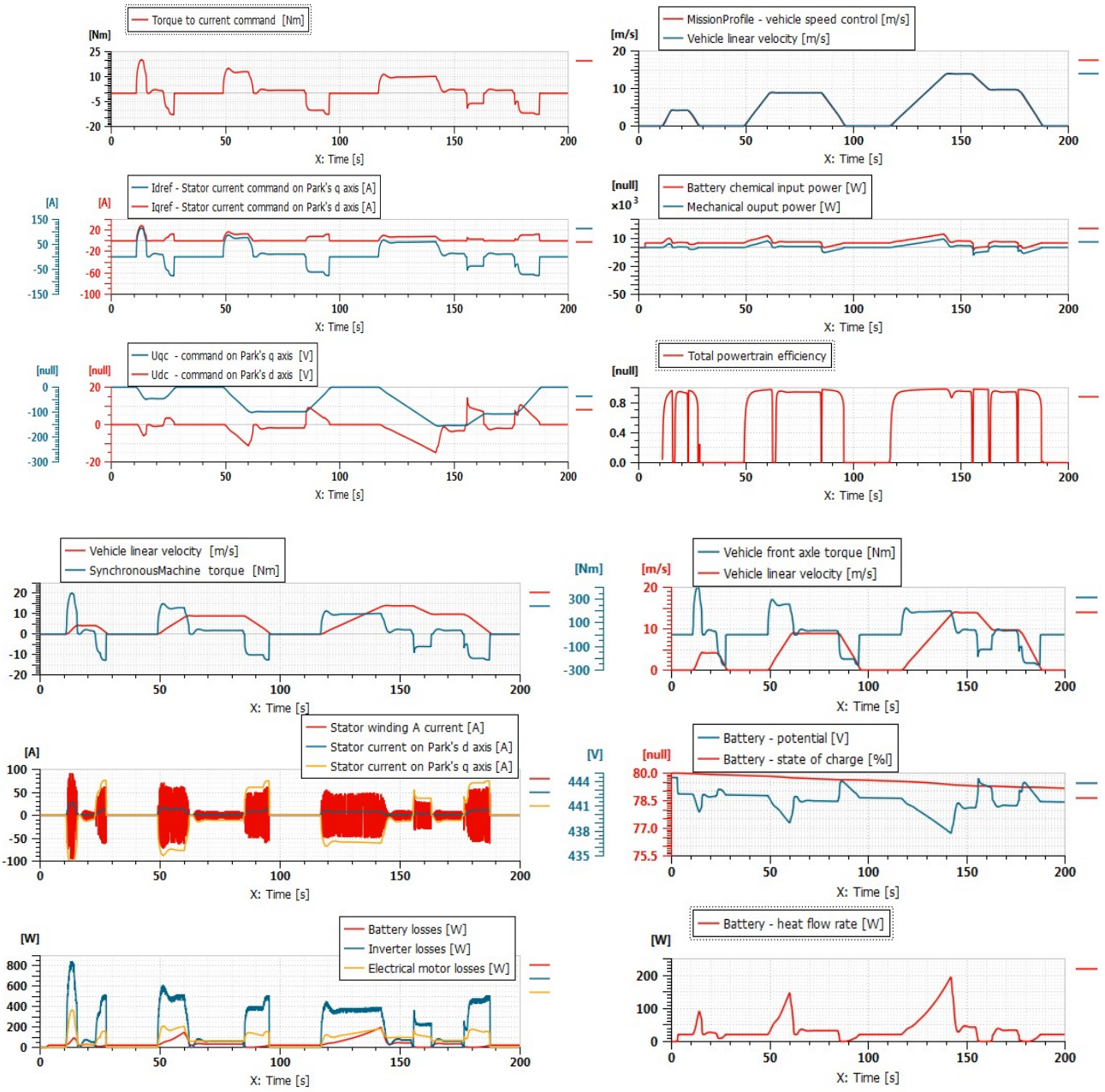


Fig. 18. Performanțele electromecanice și energetice ale VE simulat folosind co-simularea Simulink-Amesim.

4. Componentele propulsie electrică și dezvoltare echipamente/interfață hardware

4.1 Componente hardware

În proiect se vor testa mai multe tipuri de motorizări, surse generatoare de vibrații și zgomote. Acestea din urmă pot fi integrate într-o bază de date care, asociată software-ului de evaluare a managementului de energie și a interfeței virtuale conectată la platforma de funcționare în timp real vor contribui la realizarea unui mediu de testare cel mai realist posibil. Pentru atingerea acestui deziderat fost necesară interfațarea câtorva convertizoare statice și a traductoarelor de detecție a vitezei absolute, de tip resolver, instalate pe motoarele de propulsie.

Deoarece invertorul vizat în cadrul proiectului este dedicat aplicațiilor de tracțiune electrică, aceasta nu dispune de o interfață cu utilizatorul pentru a putea fi paramatrizat și comandat. Pentru

realizarea comunicației între invertor și utilizator, o cutie de conectică este necesară a fi confecționată. Aceasta cutie are rol de a interconecta calculatorul cu invertorul prin intermediul unui port serial RS232 și port CAN, dar și cu elementele de câmp cum sunt pedala de accelerație, frână, comutatorul de sens, senzori de temperatură, etc. Funcționarea unui astfel de sistem de propulsie electrică cu mașină electrică de mare viteză (22000r/min, la 20kW și 400Vcc tensiune de alimentare din baterie) este prezentată în Fig. 19. În urma stabilirii comunicației între calculator și invertor, acesta din urmă necesită a fi parametrizat conform caracteristicilor mașinii sincrone cu magneti permanenți (folosind un software adecvat). S-a realizat și cutia de control corespunzătoare pentru controlul turației mașinii de mare viteză, vezi Fig.20. În urma confecționării, s-a lansat o comunicație între un calculator și invertor pentru a-i actualiza firmwareul și a-l parametriza. Ca să putem comunica cu invertorul, am folosit un convertor *serial to usb*.

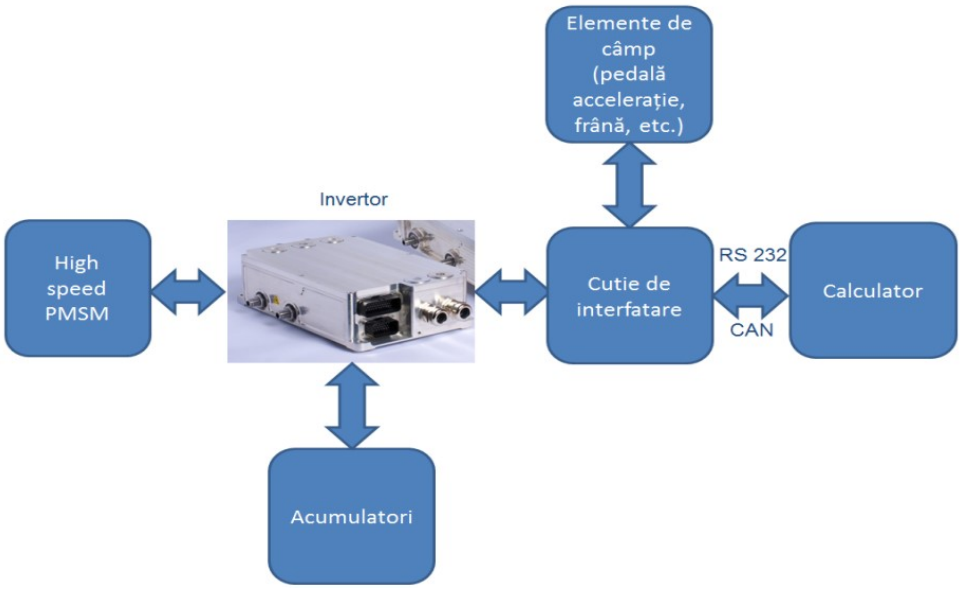


Fig. 19 Sistemul de tracțiune electrică propus/studiat.

Fig. 20 a) Cutie de interfațare și comandă; b) Conectorii invertorului; c) semnal resolver.

4.2 Control cu placă de dezvoltare pe bază de FPGA : cosimulare și evaluare performanțe

Pentru implementarea controlului în timp real se vor folosi două platforme : dSPACE și Anvyl cu miez de Spartan 6 FPGA Unit. Cum ultimul dispozitiv este mult mai ieftin și cu potențial de implementare la beneficiarul cercetării din proiectul VIPER, aici se va detalia ce conținea această placă precum și implementarea contrlului și rezultatele obținute. Lucrul cu platforma de dezvoltare “Anvyl Spartan-6 FPGA Trainer Board”, prezentată Fig.21, a presupus următoarele:

- Foloxirea de unelte software Xilinx dedicate rulării proiectelor în timp real pe platformă:
  - adaptarea versiunii ISE Design Suite cu: sistemul de operare (Windows 7, 64 biti), variante de MATLAB Simulink (Simulink 2012a, 2012b), unitate de procesare (familia Spartan-6).

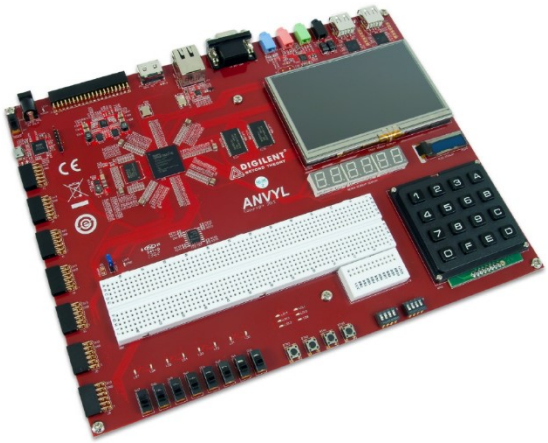


Fig. 21. Anvyl Spartan-6 FPGA Trainer Board

  - s-a ales versiunea ISE Design Suite 14.7, cu System Generator for DSP, care permite rularea în regim de co-simulare a unui proiect.
- Implementarea schemei de control vectorial pentru un motor PMSM folosit pentru propulsia unui scuter electric, folosind 'Xilinx System Generator for DSP' şi modul Co-simulare:
  - folosirea blocurillor dedicate Xilinx/Simulink pentru modelarea regulatoarelor PI din bucla de comandă a PMSM; datele sunt reprezentate în virgulă fixă, specific modului de operare a FPGA-urilor.
  - adaptarea modelului şi a parametrilor de simulare pentru rularea modului de Co-simulare; astfel, modelele matematice corespunzătoare regulatoarelor modelate cu blocurile dedicate Xilinx vor fi executate în timp real, pe nucleul Spartan 6 al platformei.
- Interpretarea rezultatelor de co-simulare pentru funcţionarea în timp real a unui scuter electric, propulsat de PMSM, într-o schema de control vectorial. S-au evaluat performanţele energetice ale sistemului, comportamentul din punct de vedere electric şi mecanic în condiţiile impuse prin schema de control.

Explicarea metodologiei de implementare a schemei de control vectorial pentru PMSM într-o schemă de control în timp real folosind 'Xilinx System Generator' şi modul 'Co-simulare hardware'

Una dintre cele mai folosite metode de modelare-simulare a aplicaţiilor de control este prin mediul MATLAB/Simulink. Din punct de vedere al implementării la normele impuse de simularea/funcţionarea în timp real, mijloacele software oferite de 'Xilinx System Generator for DSP' sunt create tocmai pentru a se adapta modelelor create prin uneltele şi librăriile specifice Simulink-ului. Algoritmii matematici corespunzători pot fi, astfel, implementaţi şi executaţi în timp real, direct pe FPGA, fără a fi nevoie de cunoştinţe de programare HDL, ci doar facând adaptarea tipurilor de date (virgulă mobilă- virgulă fixă) şi a frecvenţei de simulare.

Facilitatea oferită de acest mod de testare a unui sistem complex sau a doar a unei părţi a acestuia se poate realiza fie direct, prin generarea codului HDL corespunzător proiectului realizat în Simulink sau folosind co-simularea hardware, folosită şi în acest caz. Astfel, schema Simulink modelată prin blocuri dedicate Xilinx este interpretată ca un modul hardware independent, folosit în co-simulare, prin transmiterea intrărilor şi citirea ieşirilor.

Folosind setul de blocuri dedicate Xilinx din MATLAB/Simulink, un element important este reprezentarea adecvată a datelor. Blocurile Xilinx *Gateway In* şi *Gateway Out* permit tranziţia de la

reprezentarea în virgulă mobilă specifica Simulink-ului la reprezentarea în virgulă fixă şi invers. Timpul de eşantionare este, de asemenea, un element important, fiind definit de către utilizator ca multiplu întreg al perioadei de simulare din modelul Simulink.

Principalii paşi şi componentele proiectului în tipul de modelare co-simulare hardware sunt prezentate în Fig. 22, pentru schema de control vectorial al PMSM.

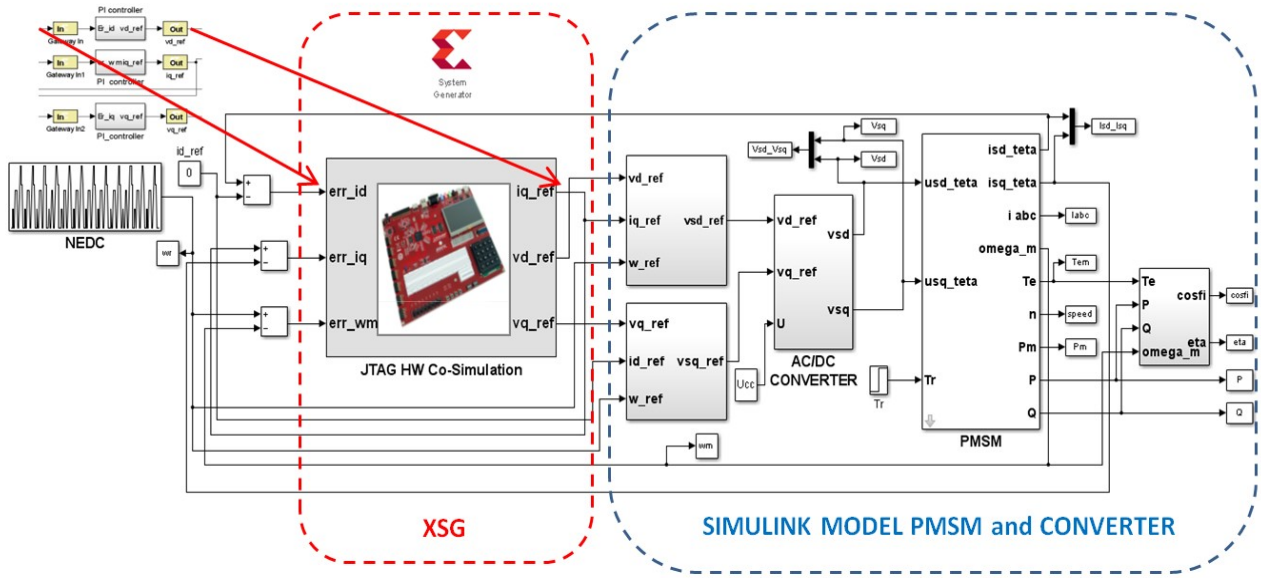


Fig. 22. Detalierea proiectului în modelarea co-simulare hardware

Modul de funcţionare este următorul:

- Modelarea ansamblului PMSM (în sistem de referinţă *d-q*) şi convertor, folosind blocuri Simulink şi reprezentarea datelor în virgulă mobilă – Fig.23;
- Implementarea regulatoarelor PI corespunzătoare controlului curenţilor *d-q* şi pentru bucla de viteză; blocurile dedicate Xilinx din Simulink sunt folosite într-o abordare matematică simplă, cu operatori matematici, cum se vede în Fig. 24.

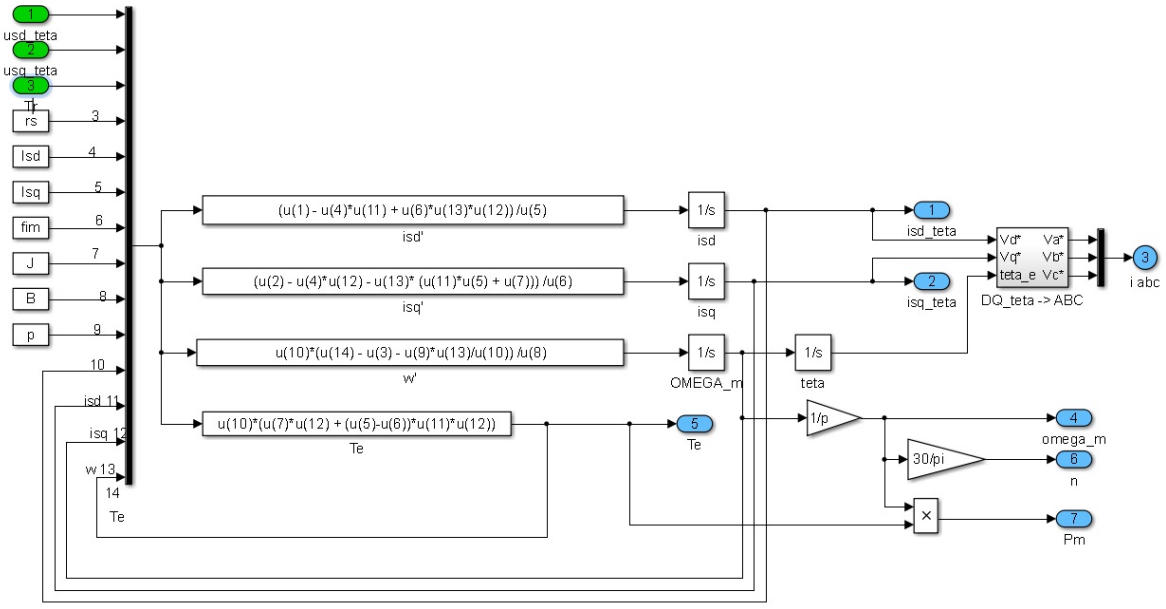


Figura 23. Modelul Simulink a PMSM

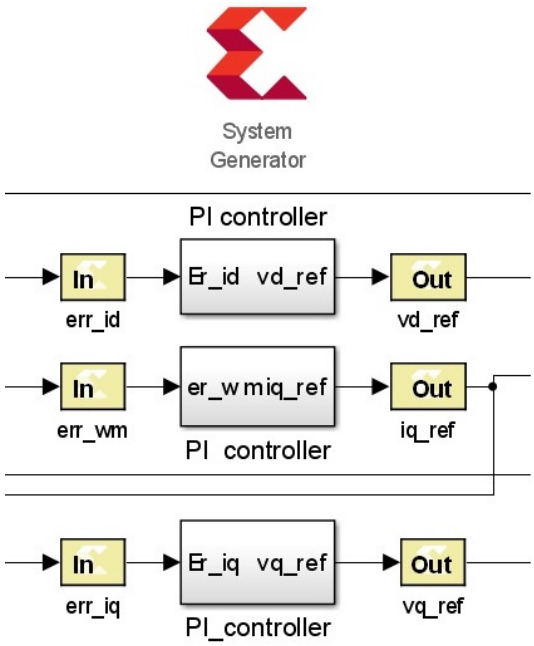


Fig. 24. Regulatoare PI proiectate cu blocuri Xilinx System Generator din Simulink

- Adăugarea blocurilor GatewayIn şi GatewayOut pentru interfaţarea modulelor Simulink realizate în virgulă mobilă cu cele realizate în virgulă fixă; funcţionarea optimă a întregului sistem este asigurată de intervenţia asupra preciziei binare alese pentru aceste blocuri.
- Interfaţarea întregului model cu Xilinx System Generator şi Simulink se realizează prin elementul *'System Generator token'*, pentru stabilirea elementului principal de procesare (FPGA Spartan 6) şi tipul de analiză *Hardware Co-Simulation*.

Sistemul de acţionare PMSM- control vectorial proiectat este destinat aplicaţiei cu scuter electric; s-a folosit ca referinţă de viteză profilul de oraş "New European Duty Cycle", până la maximum 42 rad/sec, cu durata de simulare 200 s.

Rezultatele sunt prezentate mai jos. Fig. 25 relevă că viteza măsurată/simulată urmăreşte viteza impusă; cuplul de sarcină impus după 1s este 35.4 N·m. Din punct de vedere energetic, rezultatele de simulare relevă componentele P şi Q ale puterii, după cum se observă în Fig. 26. Randamentul maşinii este ridicat (Fig. 25), iar strategia de control de tip factor de putere unitar este implementat cu succes, atingând un maxim de 99%.

În urma acestui studiu de dezvoltare din cadrul proiectului, în afara rezultatelor de simulare, care confirmă funcţionarea corectă a schemei de control în timp real implementate pe platforma FPGA Spartan 6 , se poate concluziona că folosirea mediului Simulink cu mediul dedicat Xilinx System Generator reprezintă o metodă accesibilă, eficientă şi rapidă de a urmări performanţele unui sistem de control şi performanţele acestuia în timp real, fără a fi nevoie de componente hardware sau de cunoştinţe de programare HDL şi poate substitui sau preceda implementarea experimentală a oricărei aplicaţii industriale și că ar putea înlocui cu succes platforme dedicate, gen dSPACE, extrem de constisitoare de achiziționat, în special pentru companiile private.

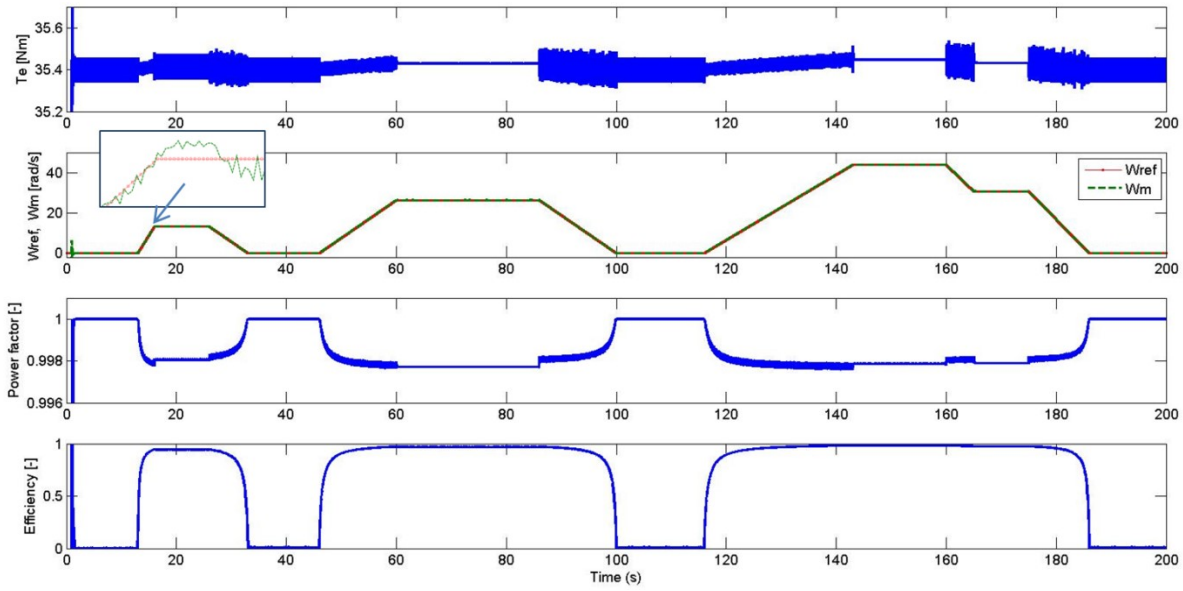


Figura 25. Parametrii mecanici şi energetici ai sistemului simulat

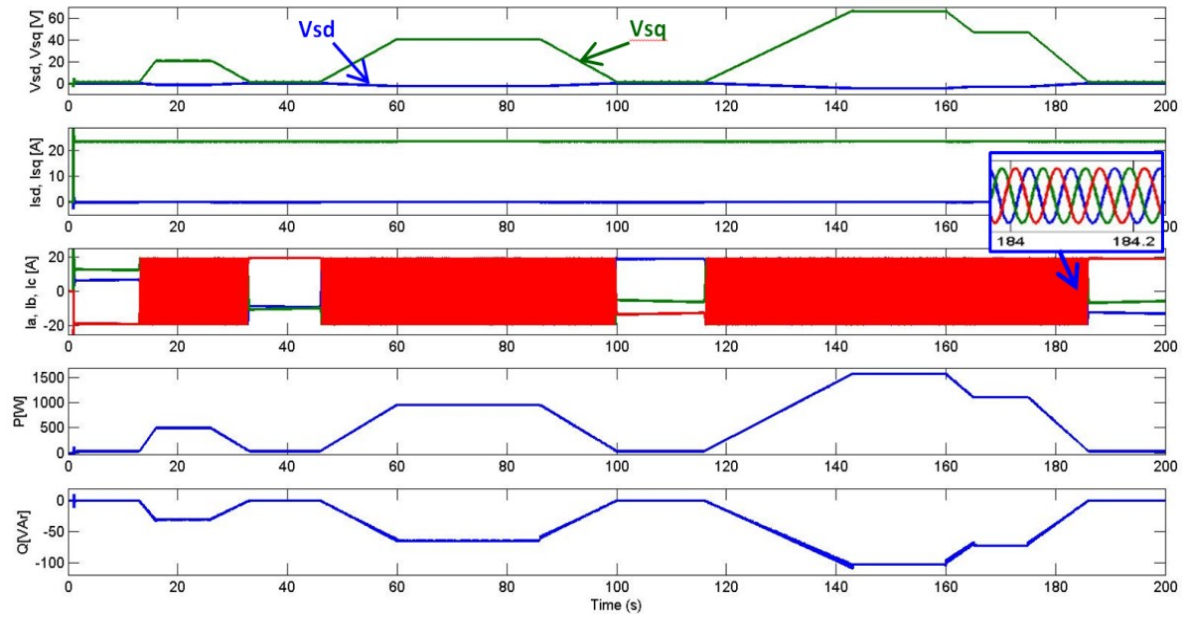


Fig. 26. Parametrii electrici ai sistemului simulat

În etapa următoare a studiului se va trece la conectarea platformei de realitate virtuală la un sistem hardware de proulsie electrică ce conține diverse tipuri de motorizare electrică (motor de mare viteză, motor roată, motor clasic de propulsie electrică, convertizorul asociat alimentării motoarelor, bateria), platforma cu FPGA și o a doua cu placă dSPACE pentru achiziția datelor și monitorizarea bilanțului energetic, sistem de direcție real precum și soluția cu volan și pedalier ce poate fi ușor adaptată pentru transferul tehnologic la partenerul industrial SISw. Testarea întregului ansamblu va permite oferirea de soluții de estimare a consumului energetic, într-o manieră cât mai realistă, incluzând și elemente vibro-acustice pentru variantele de propulsie studiate.

III. Alte rezultate semnificative – publicații

În decursul anului 2017 munca de cercetare prestată în cadrul proiectului VIPER a permis publicarea principalelor rezultate în două articole de conferință idexată *IEEE* (cu perspectiva de indexare in ISI WoS), precum și un alt articol, tot indexat *IEEE*, în colaborare cu compania beneficiară a proiectului, SISw. Sunt în pregătire două articole de revistă ISI (*IEEE Transactions on Vehicular Technology* și *IEEE Transactions on Industrial Electronics*).

In continuare se prezintă articolele publicate, subliniindu-se numele co-autorilor din postura de membrii ai proiectului VIPER, iar numele subliniate și boldate aparțin membrilor partenerului industrial.

1. G. Tamas, D. Fodorean, ***Model in the Loop Simulation of an Electric Propulsion System Using Virtual Reality***, IEEE UPEC 2017, Crete, Greece, 29 August – 1 September 2017.
2. Ioana Gros, C.Mărginean, D. Fodorean, ***FPGA Real-Time Implementation of a Vector Control Scheme for a PMSM used to propel an Electric Scooter***, IEEE ISEEE, Galati, Romania, 20-22 October 2017.
3. **Cristi Irimia, Mihail Grovu, Calin Husar**, Antonya Csaba, D. Fodorean, ***High-Speed Electrical Machine***, IEEE VVPC'17, 11-14 December, Belfort-France 2017.

Anexa I – Site WEB

Situl web actualizat se găsește la adresa: www.viper.utcluj.ro/home.

Certificăm rezultatele obținute:

Reprezentant legal Coordonator
Rector UTCN
Prof.dr.ing. Vasile ȚOPA

Reprezentant legal partner
Director general:
Dr.ing. Petru Cristinel IRIMIA

Director financiar:
Ec. Cristina RAȚIU

Director proiect
Conf.dr.ing. Daniel FODOREAN

Responsabil SISw:
Dr.ing. Călin Husar