Universidad del País Vasco Euskal Herriko Unibertsitatea
BILBOKO INGENIARITZA ESKOLA
ESCUELA DE INGENIERÍA DE BILBAO

INDUSTRIA ELEKTRONIKAREN ETA
AUTOMATIKAREN INGENIARITZA GRADUA

# GRADU AMAIERAKO LANA

## *IBILGAILU ELEKTRIKOEN PROPULTSIO-SISTEMEN RT-LAB PLATAFORMA BIDEZKO SIMULAZIO-PROZESUAREN AUTOMATIZAZIOA*

**Ikaslea**: Maite Aranguren Urrutia
**Zuzendaria:** Edorta Ibarra Basabe

**Ikasturtea:** <2018-2019>

**Data:** Bilbon, 2019ko uztailaren 18an

# Ibilgailu elektrikoen propultsio sistemen RT-Lab plataforma bidezko simulazio-prozesuaren automatizazioa

Maite Aranguren Urrutia

Bilbon, 2019ko uztailaren 18an

# Laburpena

Gradu amaierako proiektu honetan ibilgailu elektrikoen propultsio-sistemen RT-Lab plataforma bidezko simulazio-prozesuaren automatizazioa deskribatzen da. Proiektu hori burutzeko beharra gaur egungo egoera klimatikoaren ondorioz sortu da. Garraioa berotegi-efektuaren iturri nagusienetako bat denez, ibilgailu elektrikoen garapena sustatzea behar bat moduan ikusten da, zero emisioko ibilgailuak direla eta. Hala ere, horiek ezarri baino lehen, beharrezkoak diren simulazioak egin behar dira. Jakinda propultsio-sistema ibilgailu elektrikoen osagai nagusia dela, propultsio-sistemetan oinarritutako simulazioak garatu behar dira horien diseinu-prozesuan zehar.

Hori guztia kontuan izanik, lan-denbora aurrezteko simulazio horiek automatizatzea da aukerarik onena. Horretarako, OPAL-RTren OP4510 platarforma erabiliz, APERT ikerkuntza-taldearentzat baliagarria izango diren script desberdinak garatu dira. Alde batetik, simulazio azeleraturako automatizazio-script-a, eta bestetik, denbora errealeko simulaziorako automatizazio-script-a. Hori egindakoan, script-a inplementatu eta emaitzak aztertu dira.

# Resumen

En este trabajo de fin de grado se describe el proceso automatizado de las simulaciones del sistema de propulsión de los vehículos eléctricos utilizando la plataforma RT-LAB. La necesidad de llevar a cabo este trabajo ha surgido a raíz de la situación climática actual. El transporte es una de las principales causas del efecto-invernadero, por lo tanto, se ha encontrado la necesidad de impulsar el desarrollo de los vehículos eléctricos, debido a que son vehículos que emiten 0 emisiones. Teniendo en cuenta que el sistema de propulsión es el elemento principal de los vehículos eléctricos, hay que desarrollar simulaciones en base a los sistemas de propulsión.

Teniendo en cuenta todo lo citado anteriormente, para poder rebajar el tiempo de trabajo invertido en las simulaciones, automatizar esas simulaciones sería la mejor opción. Para ello, se han desarrollado diferentes scripts que serán útiles para el grupo de investigación APERT utilizando la plataforma OP4510 de OPAL-RT. Por un lado, se ha desarrollado el script que automatiza simulaciones en tiempo acelerado y, por otro lado, se ha desarrollado el script para automatizar simulaciones en tiempo real. A continuación, se han implementado los scripts y se han analizado los resultados.

# Abstract

In this end-of-degree project, the automation of electric vehicles' propulsion system is described, by means of RT- Lab platform simulation process. The need to carry out this project is based on today's climate situation. As transport is one of the main causes of the greenhouse effect, promoting the use of electric vehicles is seen as a real necessity, since they are zero emission means of transport. However, before implementing them, some simulations must be done. Taking into account that the propulsion system is the main part of electric vehicles, simulations based on propulsion system must be developed.

Bearing every aspect in mind and in order to save time, the best option is to automate those simulations. For this purpose, by using an OPAL-RT OP4510 platform, various scripts that will be useful for APERT research group have been developed. Firstly, the automation script for accelerated simulation, and secondly, the automation script for real time simulation. After carrying that out, the script has been implemented and the results have been examined.

# Akronimoen zerrenda

APERT: Applied Electronic Research Team
BEV: Battery Electric Vehicle
CFC: Chlorofluorocarbons
CPU: Central Processing Unit
eHS: Electric Hardware Solver
FCV: Full Cell Vehicle
FPGA: Field-programmable Gate Array
FTP: File Transfer Protocol
HEV: Hybrid Electric Vehicle
HIL: Hardware-in-the-Loop
I/O: Input/Output
MIL: Model-in-the-loop
MOSFET: Metal-oxide-semiconductor Field-effect transistor
PHEV: Plug-in Hybrid Electric Vehicle
PMSM: Permanent Magnet Synchronous Machine
PWM: Pulse Width Modulation
RCP: Rapid Control Prototiping
SIL: Software-in-the-Loop
SM-PMSM: Surface Mounted Permanent Magnet Synchronous Machine

# Aurkibidea

# Irudien zerrenda

# Taulen zerrenda

# I. atala

# MEMORIA

# 1. kapitulua

# Sarrera

Gaur egun, ingurumen arazoetako bat kutsadura atmosferikoa da. Kutsadura atmosferikoa airean pertsonentzat eta edozein ondarentzat arriskua edo kalteak ekar ditzaketen substantziak edo energia-motak egotean datza [3]. Kutsatzaile nagusiak sufrezko osagaiak ($SO_2$, $SO_3$, $H_2SO_4$ eta $H_2S$), nitrogeno oxidoak (NO, $NO_2$ eta $N_2O$), karbono oxidoak (CO eta $CO_2$), solidoak eta likidoak (aerosolak), hidrokarburoak, ozonoa eta metal astuna (Be, Cd eta Hg) dira. Horrez gain, hirietako kutsadura akustikoa ere kontuan hartu beharko litzateke.

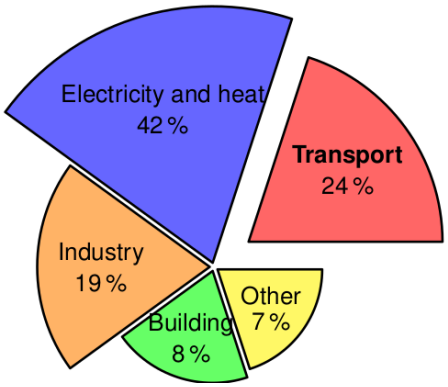
Zoruak eta ozeanoen azalak Eguzkiaren izpien xurgapenetik datorren energia termikoa dute. Horren ondorioz, lurgainak atmosferarantz erradiazioa igortzen du. Lurrak igortzen duen erradiazioa infragorria da. Atmosferako gas batzuk ($CO_2$ eta ura batez ere) erradiazio infragorria ondo xurgatzen dute, eta, ondorioz, berotu egiten dira eta hauekin atmosfera. Gas horiek egongo ez balira Lurraren batez besteko tenperatura 30°C baxuagoa izango litzateke. Beraz, fenomeno horri esker garatu ahal izan da bizia lurrean.

Erradiazio infragorria xurgatzen duten gasak ($CO_2$, ur-lurruna, ozonoa, metanoa, oxido nitrosoa eta CFCak) negutegi-efektu gasak deitzen dira [4], berotegietan kristalek duten funtzio bera dutelako. Baina, mundu mailan negutegi-gas sortzaile nagusiak aztertuko bagenitu, 1.1. irudiko grafika lortuko genuke.

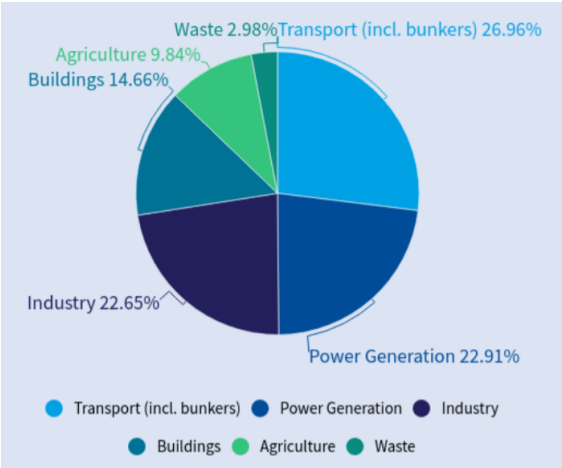
Ikusten denez, berotegi-gas sortzaile nagusia elektrizitatea lortzeko indutria da, eta bigarrena garraioa. Europar Batasuna aztertu ezkero, 2015. urtean berotegi-efektu gasak % 0.5 igo ziren [5]. Alde horretatik, 1.2. grafikoan ikusten den moduan, horien sortzaile nagusia garraioa da. Kutsadura atmosferikoak osasunean dituen epe motzeko ondorio garrantzitsuenak heriotzen gehikuntza edota urgentzietara joaten den gaixoen hazkundea da [6], batik bat, arnasketa-sistemaren gabeziengatik eta bihotzaren arazoengatik. Epe luzera, aldiz, arazo kronikoak ager daitezke.

Emisio horiei aurre egiteko zenbait konponbide bilatu dira, hala nola, 0 emisirik ez duten garraioak bilatzea eta sustatzea, beraz, ibilgailu elektrikoen garapena bermatu beharra dago. Horiek produzitzerako orduan, kontuan hartu behar da propultsio-sistema dela ibilgailu elektrikoen edota hibridoen osagai nagusia. Topologia desberdinak existitzen dira [7], honako hauek dira ohikoenak:

- Hibrido serie konfigurazioa *(series HEV, Hybrid Electric Vehicle).*
- Hibrido serie entxufagarri konfigurazioa *(PHEV, Plug-in Hybrid Electric Vehicle).*




**1.1. irudia**: Mundu mailako berotegi-efektuko gasen sortzaile nagusiak. [1]




**1.2. irudia**: Europar Batasunean negutegi-gasak isurtzen dituzten iturriak. [2]

- Hibrido paralelo konfigurazioa *(parallel HEV, Hybrid Electric Vehicle)*.
- Konfiggurazio guztiz elektrikoa *(BEV, Battery Electric Vehicle)*.
- Erregai-pilan oinarritutako ibilgailuak *(FCV, Full Cell Vehicle)*.

Bestalde, ezin da ahaztu ibilgailu elektrikoen garapen on bat egiteko simulazioak beharrezkoak direla horien diseinu-prozesuaren hasierako etapetan. Alde horretatik, lan honetan Simulink eta RT-Lab plataformei esker simulazioak gauzatu dira. Simulazioak automatizatu dira simulazio-denborak optimizatzeko, kostuak eta lan-denbora murrizteko. Lan honetan, horretarako, Python liburutegiak erabiliz helburu horiek betetzen dituen script-a garatu egin da.

# 2. kapitulua

# Testuingurua

Lan hau UPV/EHUko APERT (*Applied Electronics Research Team*) ikerkuntza-taldean burutu da. Bilboko Ingeniaritza eskolako irakasle, ikertzaile doktoregaiak zein doktoratu ostekoak eta gradu eta graduondoko ikasleek osatzen dute ikerkuntza-talde hau. Taldearen lan-ildo nagusiak hurrengoak dira:

- Administrazio publikoek finantzatutako Ikerkuntza eta Garapen (I+G) proiektuetan lankidetzak gauzatzea.
- Enpresetarako I+G proiektuak gauzatu, beti kontratupean eta taldearen ikerkuntza ildoen barne.
- Enpresetarako trebakuntza-ikastaroak, taldearen ikerkuntza lerroen inguruan.
- Aholkularitza teknikoa, azterketa teknikoak edo txostenak taldearen ikerkuntzarekin erlazionatutako gai ezberdinetan.
- Doktorego-tesiak, nazioarteko aldizkarietan argitalpenak, kongresuak eta patenteak gauzatzea.

Alde horretatik, bi dira taldearen ikerketa arloak. Alde batetik, zirkuitu birkonfigurafarriak eta System-on-Chip-ak; bestetik, energia bihurgailuentzako kontrol- eta potentzia-zirkuituak.

Zehazki, lan hau potentzia-sistemen diseinuan eta garapenean zentratzen den ikerketa-taldearen parte da. Gaur egun, APERT taldearen ikerketa-lerroak potentzia elektronikaren hurrengo esparruetan zentratuta daude:

- Ibilgailu elektrikoaren propultsio-sistemaren elektronika. Lerro honetan, potentzia-sistemen kontrola, bihurgailuen hozte-zirkuituak eta ibilgailu elektrikoan erabilitako potentzia-bihurgailuen hardwarea ikertzen dira, besteak beste.
- Korronte zuzeneko transmisioa eta banaketa. Ikerkuntza-ildo honetan, itsas energiaren transmisioa eta banaketa sistemak garatzen dira.

Zehazki, gradu amaierako lan hau ibilgailu elektrikoen propultsio-sistemetarako denbora errealeko plataforma baten simulazio-prozesua automatizatzea du helburu. Horretarako OPAL-RT fabrikatzailearen RT-Lab plataforma erabiliko da. Plataforma horren bidez

posible izango da potentzia-sistema hainbat esparrutan aztertzea (kontrola, errendimendua, galerak, etab.) gida-ziklo luzeetan zehar, parametro ezberdinak aldatuz eta horien eragina aztertuz.

# 3. kapitulua

# Proiektuaren helburuak eta irismena

Lan honen **helburu nagusia** da **APERT ikerkuntza-taldeak ibilgailuen propultsio-sistemak simulatzeko erabiltzen duen RT-Lab plataforma automatizatzea, simulazioak parametrikoki gauzatzeko.** Horrela, APERT taldearen simulazio-gaitasuna hobetuko da; era horretara posible izango dira simulazio konplexuak parametrikoki exekutatzea, garatutako diseinuen optimizazioa egin ahal izateko.

Hori guztia kontuan hartuta, hauexek dira helburu nagusia lortzeko bete beharreko **helburu espezifikoak** edo bigarren mailako helburuak:

1. Artearen egoera aztertzea, simulazioak potentzia-sistemen diseinuan dituen funtzioak ulertzeko.
2. OPAL-RTren OP4510 plataforma ezagutzea eta bere erabileran trebatzea.
3. Simulazio parametrikorako beharrezkoak diren programazio-ezagutzak lortzea. Horretarako, Python programazio-lengoaia aztertuko da, eta OPAL-RTk simulazio-plataforma kontrolatzeko garatutako komando espezifikoak ikasiko dira.
4. Automatizazio-prozesua gauzatzeko script-ak garatzea eta balioztatzea.

Beraz, proiektu honen **irismena** da **etorkizunean APERT ikerkuntza-taldearentzat baliagarria izango den denbora errealeko simulazio-plataforma automatizatzea**; horrela, taldeak posible izango du ibilgailu elektrikoen propultsio sistemaren simulazio parametrikoa gauzatzea, sistemaren optimizaziorako.

# 4. kapitulua

# Artearen egoera

## 4.1. Sarrera

Simulazio bat egiteko beharra sortzen da sistema erreala adierazten duen eredu matematikoa konplexuegia denean, edo analisi-metodorik ez edukitzearen ondorioz beste modu batean konpondu ezin denean. Adibidez, potentzia-bihurgailuak dituzten propultsio-sistemak sistema nahiko konplexuak dira, eta egoki suertatzen da horien analisia (hasierako etapetan) simulazio-erreminten bidez gauzatzea. Alde horretatik, sistema konplexuak direnez, eta kasu askotan baldintza oso berezietan simulatu nahi direnez, gaitasun handiko gailu digitalak erabili behar dira simulaziorako.

Lan honetan OPAL-RT fabrikatzailearen denbora errealeko OP4510 gailua erabili da. Gailu horrek hainbat simulazio-modalitate gauzatzeko aukera eskaintzen du. Alde horretatik, jarraian eskuragarri diren modalitateak azaltzen dira, horien ezaugarri nagusiak bistaratuz.

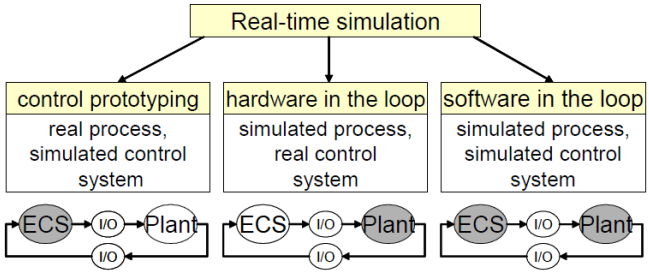
## 4.2. Simulazio-modalitateak

Simulazio-modalitateak offline edo denbora errealekoak izan daitezke [8, 9]. Offline simulazioa ordenagailu konbentzionaletan gauzatzen da. Aldiz, OP4510 bezalako gaitasun handiko gailu digitaletan bestelako modalitateak daude. Alde batetik, posible da offline simulazioen baliokideak diren denbora ez errealeko simulazioak gauzatzea. Bestalde, posible da ere simulaketak denbora errealean egitea.

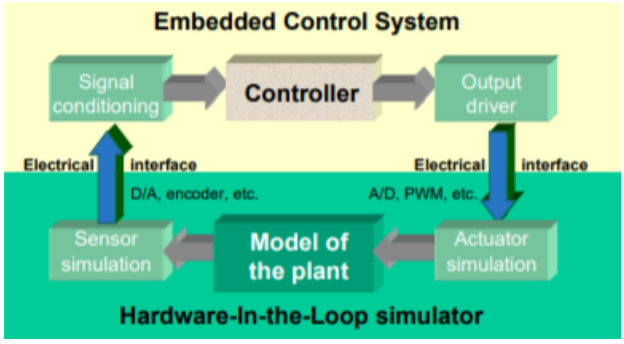
Denbora-errealeko simuladoreak simulazio fidagarriak eta zehatzak gauzatzeko gai dira [10]. Simuladore horiek konplexuak diren denbora-errealeko simulazioetan erabiltzen dira. Denbora-errealeko simuladoreen barnean 3 mota desberdineko simulazio-modalitateak gauzatu daitezke, hori da, RCP eta HIL. 4.1. irudian ikus daiteke denbora-erreala inplementatzeko modu desberdinak:

- **HIL**: Hardware-in-the-Loop

  HIL motako simulazioetan hardwarea erabiliz plantaren modeloa edo prozesua ordezkatzen da, denbora-errealean simulazioak egiteko. 4.2. irudian HIL motako simuladorearen kontrola eta elementuak ikus daitezke.




**4.1. irudia**: Debora-erreala inplementatzeko metodo desberdinak.




**4.2. irudia**: Kontrol sistema txertatua eta HIL simuladorea.

Honako abantaila hauek biltzen ditu HIL simulazioak:

- Sistema konplexuetan erabiltzen diren kontrolagailuak frogatzeko oso erabilgarriak dira [11]. Hala nola, autogintzan eta zirkulazio-kontroletan. Erreaktore nuklearren kontrolean eta industria kimikoan ere erabiltzen dira, sistema horien hutsegite batek kostu handiagoa dakarrelako HIL simuladore batean egon litzatekeen hutsegitearekin konparatuz.

- Frogak, kontrolagailua bere baldintzen mugetan egonda egin daitezke. Planta errealean gauza bera egingo balitz, arriskutsua eta kostu handikoa izango litzateke. HIL simuladoreak modeloak sortutako seinale guztiak kontuan hartzen ditu. Kontrolagailua begizta itxian frogatuz egoera errealaren portaerara gerturatzea lortzen dugu.

- Diseinua eta software-frogak aurreko fase batera mugitu daitezke, lehenengo prototipo fisiko/mekanikoa amaitutzat eman baino lehen HIL simuladore batean gauzatu daitezkeelako.

- Plantaren parametroak aldatu daitezke, planta modelatu behar den momentuan edo denbora-errealeko simulazioa martxan dagoenean.

- Hardwarea beste modelo desberdinetarako berrerabili daiteke aldaketa fisiko handirik egin gabe, aldaketarik egitekotan, erraz egin daitezke.

Ondoren, simulazio-modalitate honen desabantailak edota zailtasunak aurkezten dira:

- HIL simuladoreak ezin ditu planta errealean egin behar diren froga zehatzak ordezkatu. Baizik eta, kontrolagailuaren garapenean eta arazketan laguntzen du. Nahiz eta, plantaren modeloa oso zehatza izan, beti parametroren bat ez da kontuan hartuko.

- Simuladorea martxan dagoen bitartean ezin da simulazioa eten.

- Kontrolagailuaren portaera era orokor batean aztertzen da. Hortaz, HIL simuladoreak ez daki zer gertatzen den kontrolagailuaren barnean. Kontrolagailuaren irteerak bakarrik irakurtzen dituenez, software hutsegite bat egotekotan, zaila da jakitea kodearen zein puntu exekutatzen ari den edo barne aldagaien egoerak.

- **RCP**: Prototipatze azkarreko kontrola.

RCP modalitatea produktuaren garapen-denbora bizkortzeko teknologia garrantzitsuenetako bat da, modelo batean oinarritutako kontrolaren diseinuaren eta inplementazioaren arteko trantsizioa errazten laguntzen duelako [12]. Diseinuaren etapan erabiltzen da, hain zuzen ere, mundu errealeko dinamikari aurre egiteko diseinatutako kontrol-estrategiak azkar egiaztatzeko.

RCPak hardware modularrak diren sistemak dira eta software euskarri onak dituzte. Hardware modularitatea ezaugarri bat da sistema eraikitzen eta betebeharreko baldintzak betetzen laguntzen duena. Gutxienez, RCP simuladorearen modulu bat mikroprozesadore modulu bat izan behar da. Modulu honi esker, potentzia konputazionala eta memoria-edukiera handitzea lortzen da, modu honetan, kontrol aurreratuko algoritmoak egiaztatzen dira, kode optimizatu baten beharra izan gabe.

Software euskarri on bati esker, RCP sistema modularren produktoreak liburutegiak eta dokumentazioa hornitzea ahalbidetzen du. Liburutegi horiek aplikazioen interfaze modura erabiltzen dira, gailuen ezaugarrien erabilerarako. Interruptoreen erabileran eta analogiko digital bihurketen balioen irakurketan erabiliak dira. Hostaren kodea RCP sisteman exekutatzen da. Host zerbitzu honek denbora-errealeko hardwarearen eta host konputagailuaren arteko datu-trukaketa ahalbidetzen du. RCP sistemaren beste ezaugarri bat software bidez konfiguratu daitekeela da. Hau da, garatzen ari diren RCP sistemen erabilerak algoritmo programaketak tratatzea ahalbidetzen du.

RCP aplikazioetan denbora-errealeko simuladore bat erabiliz, planta kontrolagailu bat inplementatzen da eta planta fisiko batera konektatzen da. RCPak kontrolagailu erreal baten prototipoaren inplementazioaren aurrean abantaila asko eskaintzen ditu. Denbora-errealeko simuladore bat erabiliz garatutako kontrolagailu prototipo

bat malgutasun gehiago, inplementatzeko azkarragoa eta arazteko errazagoa izango da.

Bestalde, simulazioa azkartzeko aukera ere existitzen da horrelako gailu digitaletan. Horretarako SIL (Software-in-the-Loop) modalitatea existitzen da, zein denbora errealean exekutatu daitekeen edo ez:

- **SIL**: Software-in-the-Loop

  SIL simuladoreak prozesuaren portera frogatzen du, hardwerean edo denbora-errealeko hardware simulazioan intengratu baino lehen [13]. Gainera, SILak RPC eta HILen arteko konbinazioa baino urrunagoko hirugarren pausoa adierazten du. Potentzia nahikoa duen simuladore bat erabiliz, planta eta kontrolagailua denbora-errealeko simuladore berberean simulatu daitezke. Froga hori pauso garrantzitsua da, baina ez du HIL simulazioa guztiz ordezkatzen, azken batean, errealitatearen eta simulazioaren desberdintasunak daudelako.

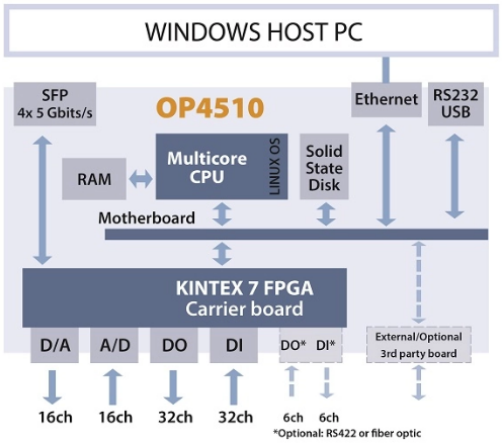
  SIL simuladorea RCP eta HIL simuladoreekin konparatuz, ez ditu sarrerak eta irteerak erabiltzen [14]. Horretaz aparte, kontrolagailua eta planta simuladore berberean egotean, kanpo munduarekin partekatutako denbora ez da kritikoa izango. Denbora-erreala baino motelago edo azkarragoa izan daiteke, emaitzetan eraginik izan gabe. Beraz, SIL simuladorea simulazio azeleratua deituriko simulazio mota bat da. Modu azeleratuan simulazioa denbora-errealean baino azkarrago exekutatzen da. Modu honetan, periodo motz batean froga kopuru handia gauzatu daitezke. Hala ere, denbora-errealean baino motelago ere exekuta daiteke.

## 4.3. Gaitasun handiko RT-Lab plataforma

Jarraian lan honetan automatizatu den OPAL-RT fabrikatzailearen OP4510 plataforma deskribatzen da. OP4510 gailuak hainbat sarrera/irteera ditu aurreko atalean azaldutako simulazio-modalitateak gauzatu ahal izateko. OPAL-RT fabrikatzaileak eskaintzen dituen ezaugarriak biltzen ditu, errendimendu altuko kontrol-ereduen eta zirkuituen hardware simulazioak sortzeko.

Azken belaunaldiko Intel Xeon prozesadorea du gailuak, eta potentzia handiko Xilinx Kintex 7 FPGA batez hornituta dago [15]. OP4510ren txasis trinkoak mahaigainerako konfigurazioetan eta bastidore muntaietarako funtzionamendu egokia du. Gainera, seinaleak egokitzen dituzten errendimendu altuko 128 kanal digital/analogiko eta hardwarearen interfazearentzako SFP-GTX abiadura handiko 4 lotura optiko eskaintzen ditu. Konfigurazio estandarrak 32 irteera digital, 32 sarrera digital, 16 irteera analogiko eta 16 sarrera digital ditu. 4.3. irudiak erakusten du gailuaren hardware arkitektura.

Malgutasun handiko arkitektura duen simuladorea da. FPGAaren arkitektura konbezionalaren eta OP4510aren CPUaren arkitekturaren azkeneko teknologiak batzen ditu, erabiltzaileak 10$\mu$s-ko eta 20 $\mu$s tarteko simulazio-denborak lor ditzan denbora-errealeko simulazioetan. Bestalde, posible da FPGAaren eta CPUaren arteko simulazio partekatua egitea, PCIexpress lotura azkar baten bidez gailuek datuak eta seinaleak elkar trukatzen dituztelako. Ezaugarri honi esker, abiadura handiko FPGAn oinarritutako modeloak ako-




**4.3. irudia**: OP4510 gailu digitalaren arkitektura.

platu daitezke, hala nola potentzia-bihurgailuak eta sistema elektrikoak; aldiz, CPUan sistema mekaniko eta elektriko motelagoak edota kontrolatzaileak simulatu daitezke.

FPGAren 10 ns-ko denbora-erresoluziari esker erabiltzaileek PMWetan oinarritutako RCP kontrolagailu garatuak gauzatu ditzakete, edota kontrolagailu horiek hardware errealean probatu daitezke.

## 4.4. Simulazioen automatizazioa

### 4.4.1. Simulazio automatizatuaren zergaitia

Simulazioen automatizazioa beharrezkoa da modeloan lortutako emaitzak parametrikoki aztertu nahi direnean. Adibidez, kontrolagailu baten kasuan interesgarria izan daiteke erreguladoreei irabazi ezberdinak ezartzea, horien menpe lortzen diren emaitzak aztertzeko eta sisteman egokienak izan daitezkeen parametroak aukeratzeko. Bestalde, sistema hainbat baldintzatan simulatu nahi denean, beharrezkoa da operazio-eszenatoki ezberdinak definitzea, eta horiek guztiak simulatzea, aztergai den planta edota kontrolagailua ahalik eta baldintza-kopuru handienean aztertu ahal izateko.

Adibidez, Matlab/Simulink-en posible da simulazio automatizatu eta parametrizatua gauzatzea “m” hizkuntzan eskuragarri dauden komandoen bidez; hori da, aldagaiak aldatu egin daitezke `for` edota `while` zikloen bidez, eta modeloaren simulazioa abiarazi egin daiteke `sim` komandoa eta bere aukerak erabiliz. Prozesu hori nahikoa sinplea da; hala ere, OPAL-RT-ren gailu digitaletan beharrezkoa da fabrikatzaileak emandako softwarearen bidez modeloekin elkarrekintza egitea. Alde horretatik, baliabide grafikoak eskaintzen ditu fabrikatzaileak, baina horiek simulazio sinpleak egiteko dira bakarrik baliagarriak. Aldiz, Python hizkuntza erabili behar da modeloengan funtzio aurreratuak gauzatzeko (hala nola parametrizazioa eta modeloen exekuzioen automatizazioa eta datuberreskuraketa). Alde horretatik, jarraian OP4510 gailua automatizatzeko eskuragarri dauden Python-baliabideak aurkezten dira, horien funtzionamendua eta ezaugarriak azalduz.

### 4.4.2. RT-Lab-en simulazioak automatizatzeko Python baliabideak

Atal honetan, kodearen egitura eta kodean agertzen diren aginduak azaltzen dira. Script-an erabiltzen diren aginduak eskuragarri edukitzeko, RT-Lab-eko liburutegia eta time liburutegia kargatu behar dira, honako bi aginduei, *import RtlabApi as r* eta *import time*, dei eginez egiten da. Horretaz aparte, RT-Lab liburutegiko izena laburbiltzeko, as agindua erabili da, kasu honetan, r izena esleitu zaio. Ondorengo tauletan script-an erabilitako funtzio guztiak ageri dira:

**Agindu-zenbakia: 1**

**Agindua:**
r.OpenProject()

**Deskribapena:**
Agindu honi esker modeloa irekitzen da.

**Sarrerak:**

- *project:* Proiektu bakar bat aktibatuta baldin badago eta argumenturik sartu ez bada, zuzenean proiektu hori irekitzen da. Argumentu horrek aktibatuta dauden proiektuetako bat baino gehiago deskribatzen baditu, mezu baten bidez ireki nahi den proiektua aukera daiteke.
- *functionalBlock:* Argumentu honi esker bloke funtzionala deskribatzen da. Bloke funtzional honek aktibatuta dagoen proiektuaren konexioa kontrolatzen du.
- *controlPriority:* Kontrolaren lehentasun maila zehazten du.
- *returnOnAmbiguity:* Proiektua aukeratzeko mezua aktibatzeko eta desaktibatzeko balio du.

**Irteerak:**

- *projectId:* Aktibatu den proiektu berriaren IDaren erreferentzia zehazten du.

**Agindu-zenbakia: 2**

**Agindua:**
r.SetStopTime()

**Deskribapena:**
Modeloaren simulazio-denbora zehazteko balio du.

**Sarrerak:**

- Zuzenean funtzioaren argumentoan sartzen da. Zenbaki baten bidez adierazten zaio funtzio honi zein den nahi dugun simulazio-denbora. Funtzioak ezarriko du simulazio-denbora hori RT-Lab plataforman.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 3**

**Agindua:**
r.Load(realTimeMode,timeFactor)

**Deskribapena:**
Modeloa kargatzeko balio du.

**Sarrerak:**

- *realTimeMode:* Simulatzeko modu desberdinak daude, RT-Lab-ak 5 modu desberdin eskaintzen ditu.
- *timeFactor:* Denbora faktorea zehazten du. Normalean 1 balioa aukeratzen da.

**Irteerak:**

- *instanceId:* Konexioa egingo den modeloaren instantzia identifikatzen da.

**Agindu-zenbakia: 4**

**Agindua:**
r.LoadConsole()

**Deskribapena:**
Konektatu dagoen modeloaren kontsola kargatzen du. Behin kargatuta dagoenean, kontsola ireki egiten da. Simulink kontsolak soilik kargatu daitezke.

**Sarrerak:**

- Ez ditu (zuzenean kargatu den modeloaren kontsola kargatzen duenez, ez da beharrezkoa argumenturik sartzea).

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 5**

**Agindua:**
r.SetParametersByName(parameterNames,parameterValues)

**Deskribapena:**
Parametro baten edo gehiagoren balioa aldatzeko balio du.

**Sarrerak:**

- *parameterNames:* Tuplaren barnean dauden parametroen helbideak edo izenak aldagai honen barruan daude.
- *parameterValues:* Esleitu nahi diren balio berriak aldagai honetan aurkitzen dira.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 6**

**Agindua:**
r.GetParametersByName()

**Deskribapena:**
Parametro baten edo lista bat osatzen duten paramentroen balioak bueltatzen ditu.

**Sarrerak:**

- *parameterNames:* Jakin nahi diren parametroen izenak aldagai honetan aurkitzen dira.

**Irteerak:**

- *parameterValues:* Parametroen balioak lista batean bueltatzen ditu.

**Agindu-zenbakia: 7**

**Agindua:**
r.Execute(timeFactor)

**Deskribapena:**
Modeloa exekutatzeko balio du.

**Sarrerak:**

- *timeFactor*-a sar daiteke, baina ez da nahitaezkoa.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 8**

**Agindua:**
r.ResetConsole()

**Deskribapena:**
Momentuan konektatuta dagoen modeloaren simulink kontsolaren azpisistema exekutatzen du. Kontsola exekutatu aurretik, kontsola kargatuta egon behar da.

**Sarrerak:**

- Ez ditu.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 9**

**Agindua:**
r.StopConsole()

**Deskribapena:**
Momentuan konektatuta dagoen modeloaren simulink kontsolaren azpisistema eteten da. Kontsola gelditu aurretik, kontsola exekutatuta egon behar da.

**Sarrerak:**

- Ez ditu.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 10**

**Agindua:**
r.ResetConsole()

**Deskribapena:**
Momentuan konektatuta dagoen modeloaren simulink kontsolaren azpisistema berrasieratzen eta deskargatzen du. Kontsola berrasierazi aurretik, kontsola kargatuta egon behar da.

**Sarrerak:**

- Ez ditu.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 11**

**Agindua:**
r.CloseProject()

**Deskribapena:**
Irekita dagoen proiektua ixten du.

**Sarrerak:**

- Ez ditu.

**Irteerak:**

- *projectId:* Proiektuaren IDa itxita dago. ID hori jada ez du balio.

**Agindu-zenbakia: 12**

**Agindua:**
GetSignalsByName()

**Deskribapena:**
Seinale baten edo zerrenda bat osatzen duten seinaleen balioak bueltatzen ditu.

**Sarrerak:**

- *signalNames:* Jakin nahi diren seinaleen izenak aldagai honetan aurkitzen dira.

**Irteerak:**

- *signalValues:* Seinaleen balioak lista batean bueltatzen ditu.

**Agindu-zenbakia: 13**

**Agindua:**
time.sleep()

**Deskribapena:**
Python programa eteten du.
**Sarrerak:**

- Zuzenean funtzioaren argumentuan sartzen da. Zenbaki baten bidez adierazten zaio funtzio horri zenbat denbora nahi den programaren kodea geldirik egotea.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 14**

**Agindua:**
r.GetNodeCpu()

**Deskribapena:**
Modeloaren azpisistemen CPUa zerrenda batean bueltatzen ditu. CPUaren esleipena NT targeterako ez dago eskuragarri.
**Sarrerak:**

- *subsystemNames:* Azpisistema baten edo zerrenda bat osatzen duten azpisistemen CPUa aldagai honetan aurkitzen dira.

**Irteerak:**

- *cpuOptions:* Azpisistema bakoitzari esleitutako CPUa aldagai honen barruan zerrenda batean bueltatzen da. -1 balioa bueltatzen badu, modu automatikoan gaudela esan nahi du, beraz, azpisistemari zuzenean CPU bat esleitzen zaio.

**Agindu-zenbakia: 15**

**Agindua:**
r.SetNodeCpu(("subsysNames",),(cpuOptions,))

**Deskribapena:**
Zerrenda batean aukeratutako modeloaren azpisistemei CPU bat esleitzeko balio du. Hurrengo load agindua bete ondoren CPUaren esleipena eguneratuko da. CPUaren esleipena NT targeterako ez dago eskuragarri.
**Sarrerak:**

- *subsysNames:* CPUa esleitu nahi zaien azpisistemen izenen zerrenda aldagai honetan aurkitzen dira.
- *cpuOptions:* Azpisistema bakoitzari esleitutako CPUa aldagai honetan zehazten da.

**Irteerak:**

- Ez ditu.

**Agindu-zenbakia: 16**

**Agindua:**
r.GetModelState()

**Deskribapena:**
Modeloaren egoera jakiteko balio du.

**Sarrerak:**

- Ez ditu.

**Irteerak:**

- *OP MODEL STATE:* Modeloak 9 egoera posible eduki ditzake. Hurrengo taulan 9 egoera horiek azalduta daude:

| Egoera | Definizioa |
|---|---|
| MODEL NOT CONNECTED (0) | Modeloa ez dago konektaturik |
| MODEL NOT LOADABLE(1) | Modeloa ez dago kargaturik |
| MODEL COMPILING (2) | Modeloa konpilatzen ari da |
| MODEL LOADABLE (3) | Modeloa konpilatuta dago eta kargatzeko prest |
| MODEL LOADING (4) | Modeloa kargatzen ari da |
| MODEL RESETTING (5) | Modeloa berrasierazten ari da |
| MODEL LOADED (6) | Modeloa targetean kargatuta dago |
| MODEL PAUSED (7) | Modeloa targetean kargatuta eta exekutatuta dago |
| MODEL RUNNING (8) | Modeloa targetean kargatuta eta exekutatuta dago |
| MODEL DISCONNECTED (9) | Modeloa deskonektatuta dago |

- *OP REALTIME MODE:* RT-Lab-ak 5 simulazio modu eskaintzen ditu. Ondorengo taulan 5 moduak aurkitzen dira:

| Simulazio modua | Definizioa? |
|---|---|
| HARD SYNC MODE (0) | Hardware bidezko sinkronizazio modua. (WIN 32 targetean ez dago eskuragarri). Targetean I/O taulak tenporizadorea eduki behar du. |
| SIM MODE (1) | Simulazio modu azkarrena |
| SOFT SIM MODE (2) | Software bidezko sinkronizazio modua |
| SIM W NO DATA LOSS MODE (3) | Jada ez da erabiltzen |
| SIM W LOW PRIO MODE (4) | Lehentasun maila baxua edukita simulazio modu azkarrena |

# 5. kapitulua

# Automatizazioaren inplementazioa eta balioztapena

## 5.1. Sarrera

Lan honetan SIL modeloen (4.2. atala) automatizazioa landu da. Alde batetik, denbora errealean exekutatzeko beharrik ez duten SIL modeloentzako automatizazioa gauzatu da; azkenik, denbora errealeko exekuziorako script-a ere prestatu da. Ezaugarri ezberdinak dituztenez, bi script-mota garatu dira lan honetan modeloen simulazioa automatizatzeko. Alde batetik, simulazioak azeleratzeko denbora ez-errealeko CPU bidezko simulazioak gauzatzeko script-a; bestalde, denbora errealean (CPU bidez edota CPUak eta FPGA koordinatuz) simulazio anitz parametrizatuta gauzatzeko script-a.

Jarraian bi script horiek nola inplementatu diren azalduko dira, kode-lerroak jarriz eta horien deskribapen zehatza emanez.

## 5.2. Simulazio azeleraturako automatizazio-script-a

Lehenik eta behin, aurreko atalean aipatu dudan moduan, beharrezkoa da OPAL-RTren Python funtzioen liburutegia kargatzea, eta baita modeloaren simulaketa-denbora definitzea ere, hurrengo lerroetan erakusten den bezala:

```
01 import RtlabApi as r
02 import time
03 DURACION_SIMU = 1190
04 print "Modelo bakoitzaren simulazio-denbora honako hau da: "+ str(DURACION_SIMU)
```

Alde horretatik, 4. lerroan den aginduan ikusten den moduan, erabiltzaileak definitutako simulazio-denbora RT-Lab-eko interfazean (kontsolan) ikusten da.

Ondoren, *for* agindua erabili behar da, eta parametrizatu nahi den aldagaiaren hasierako balioa, bukaerako balioa eta horren gehikuntza definitu behar da. Adibide honetan 6 simulazio gauzatzeko programatu da Python script-a. Hasierako balioa 1 izango da, eta azkeneko balioa 7 bat, eta begizta bakoitzean unitate bateko gehikuntza egingo da. Kontuan hartu behar dugu azkeneko balioa ez dela inoiz exekutatzen eta hori argi izan behar du programatzaileak.

Behin hori guztia definituta dagoenean, nahi izanez gero, *for* aginduaren barneko kodea bi zatitan bana daiteke. Alde batetik, *try* aginduaren barnean arazoak eman ditzakeen kode zatia idatzi behar da (zeroz zatiketak, etab.), eta *finally* aginduaren barnean beti exekutatuko den kode-zatia idazten da, modu honetan, kodean arazoren bat egotekotan kontsola beti gelditu eta berrabiaraziko da.

```
05 for i in range(1,7,1): #int,end,increment. end balioa ez da exekutatzen
```

Hurrengo pausoan, ireki nahi den proiektua definitu behar da. Bi aukera ditugu horretarako, proiektua aldagai batean sartu eta ondoren 9. lerroan dagoen agindua erabili, edo zuzenean agindu horretan proiektuaren izena definitu. *print* aginduari esker jakin nahi ditugun aldagaien balioak kontsolan bistaratuko dira modeloaren exekuzioa gertatu ahala.

```
06     try:
07         #Proiektua ireki
08         project="Endika_multisimulation"
09         r.OpenProject(project)
10         print"Aukeratutako proiektuaren izena honako hau: "+project
11         print str(i) + ". simulazioa"
```

Behin proiektua irekita dagoela, modeloa osatzen duten azpisistemei nahi dugun CPUa esleitu diezaiokegu (simulazio paraleloa posible baita RT-Lab gailuan). Horretarako, `r.SetNodeCpu` agindua erabili behar da. Gure kasuan, bakarrik 2 CPU erabil ditzakegu lizentzia-kontuak direla eta[1]. CPUa esleituta daukagunean modeloaren egoera ezagutu behar da. Hori lortzeko `r.GetModelState` agindua erabili behar da eta ondoren modeloaren egoera bistaratu.

```
12         #CPUa esleitu
13         r.SetNodeCpu(("SM_Master",),(1,))
14         #Modeloaren egoera konprobatu
15         (modelState, realTimeMode) = r.GetModelState()
16         print ("Modeloaren egoera honako hau da: "+ str(modelState))
```

Definitutako simulazio-denbora konfiguratu behar da. Honen ondoren, modeloa eta MATLAB-eko kontsola kargatu behar dira. Modeloa eta kontsola ondo kargatzeko denbora eduki ditzaten 5 segunduz kodea gelditu behar izan da.

```
17         #Modeloaren simulazio-denbora konfiguratu
18         r.SetStopTime(DURACION_SIMU)
19         #Modeloa kargatu
20         r.Load(r.SIM_MODE, 1)
21         #MatLAB-eko kontsola kargatu
22         r.LoadConsole()
23         time.sleep(5)
```

Kasu partikular honetan, `for` aginduan definitutako balioak modulazio-teknika bakoitzaren selektorea eta irteera fitxategi bakoitzaren izenaren id-a (identifikazio-zenbakia)[2] definitzeko erabiliko dira.

[1] Gehienez 3 CPU erreserbatu daitezke gailu honetan simulazioa banatzeko, laugarren CPUa denbora errealko sistema eragilearentzat erreserbatuta baitago.

[2] Emaitzak ezberdindutako izenak eta identifikagarriak diren .mat formatuko fitxategietan gordetzeko simulaketa bakoitza amaitzen denean.

```
24          #Modulazio-teknika bakoitzaren selektorea konfiguratu
25          select="SynRM_SiC_endika/SM_master/PLANT/Modulation/select_modulation/Value"
26          r.SetParametersByName(select, i)
27          #Irteera fitxategi bakoitzeren izenaren id-a konfiguratu
28          id="SynRM_SiC_endika/SM_master/to workspace for calculations/simulation_id/Value"
29          r.SetParametersByName(id, i)
```

Berriro ere modeloaren egoera konprobatu eta bistaratu behar da. Hori jakin ondoren, modeloa eta erabiltzaile-kontsola (Simulink modelo baten bidez gauzatua) kargatu behar dira. Azkenik, modeloaren egoera berriz ere ezagutu behar da. Alde horretatik, `while` aginduari esker modeloaren egoera `MODEL RUNNING` den bitartean modeloaren simulazio-denbora eta simulazioaren portzentajea bistaratuko dira. Modeloaren egoera aldatzekotan, kodea `while` begiztatik irtengo da. `while` instrukzioaren exekuzio bakoitza amaitzean 30 segundoz gelditukо da kodea, prozesamendu guztia ondo gauzatzeko.

```
30          #Modeloaren egoera konprobatu
31          (modelState, realTimeMode) = r.GetModelState()
32          print ("Modeloaren egoera honako hau da: "+ str(modelState))
33
34          #Modeloa exekutatu
35          r.Execute()
36          #MatLAB-eko kontsola exekutatu
37          r.ExecuteConsole()
38          #Modeloaren egoera konprobatu
39          (modelState, realTimeMode) = r.GetModelState()
40          print ("Modeloaren egoera honako hau da: "+ str(modelState))
41          while  modelState == 8:
42
43              #Modeloaren egoera konprobatu
44              (modelState, realTimeMode) = r.GetModelState()
45              if modelState != 8:
46                  print ("\rModeloaren egoera honako hau da: "+ str(modelState))
47              else : #Modeloa exekutatzen ari den bitartean...
48                  #Simulazio-denbora konprobatu
49                  timeValue = r.GetSignalsByName("SynRM_SiC_endika/SM_master/Fleet-BEV-Cycle/Driving cyc
50                  porcentaje=(timeValue/DURACION_SIMU)*100
51                  print ("Simulazio-denbora: "+ str(timeValue) + " portzentajea:"  + str(porcentaje)) +
52
53              time.sleep(30)
```

Azkenik, kontsola gelditu eta berrabiarazi behar da. Modeloaren egoera berriro konprobatu eta proiektua itxi behar da.

```
54      finally:
55          r.StopConsole()
56          time.sleep(5)
57          r.ResetConsole()
58          time.sleep(5)
59  (modelState, realTimeMode) = r.GetModelState()
60  print ("Modeloaren egoera honako hau da: "+ str(modelState))
61  r.CloseProject()
62  print "Proiektua itxi da"
```

## 5.3. Denbora errealeko simulaziorako automatizazio-script-a

Berriz ere, Python liburutegia kargatzen da simulaketa-denborarekin batera, denbora errealeko simulazioak gauzatu nahi direnean:

```
01 import RtlabApi as r
02 import time
03 SIMU_DURATION = 785
```

Ondoren, simulazio-kopurua zehazten da parametrizatu nahi den aldagaiaren balioak zehaztuz, kasu honetan, 3 simulazio egiteko programatu da Python script-a eta parametrizatutako aldagaia ibilgaiuaren bateriaren korrontea da. Bateriaren korrontearen hasierako balioa 300 izango da, azkenekoa 360 eta begizta bakoitzean 20 unitateko gehikuntza egongo da. Aurreko script-ean aipatu den moduan, 360 balioa ez da inoiz exeketutako.

```
04 for i in range(300,360,20):
```

Script honetan berriro ere, `for` aginduaren barneko kodea bi zatitan bana daiteke. 7. lerroan ikusten den moduan, `print` aginduari esker bateriaren korronte maiala bistara daiteke. Honen ondoren, irekiko den proiektua aukeratu behar da.

```
05    try:
06      print("\rSIMULATION FOR Vbatt = "+ str(i))
07      #Open model
08      project="SM_PMSM_eHS_SiC_MOSFET"
```

Behin proiektua aukeratuta dagoenean, proiektua ireki behar da. Ondoren, proiektuaren egoera ezagutzeko, `r.GetModelState` agindua erabiliz lor daiteke.

```
09      r.OpenProject(project)
10      print"Aukeratu den proiektua honako hau da: "+project
11
12      #Check model state
13      (modelState, realTimeMode) = r.GetModelState()
14      print("Modeloaren egoera honako hau da: "+ str(modelState))
```

Hurrengo pausoan, simulazio-denbora konfiguratu behar da. Modeloa kargatzerako orduan `HARD_SYNC_MODE` modua aukeratu da, hau da, hardware bidezko simulazio modua. Modeloa kargatu ostean, MATLAB-eko kontsola kargatu behar da. Biak ondo kargatzeko 5 segunduz kodea gelditu behar izan da, `time.sleep(5)` agindua erabiliz etenaldia programatu da.

```
15      #Configure model simulation time
16      r.SetStopTime(SIMU_DURATION)
17
18      #Load model
19      r.Load(r.HARD_SYNC_MODE, 1)
20      #Load Matlab user interface
21      r.LoadConsole()
22      time.sleep(5)
```

Jarraian, `for` aginduan definitutako balioak bateriaren korrontearen balioa eta irteera fitxategi bakoitzaren izenaren id-a (identifikazio-zenbakia)[3] definitzeko erabiliko dira. Kasu partikular honetan, maisu bat eta morroi bat ditugunez, bi irteera fitxategi edukiko ditugu.

```
23      #Configure selector of each Vbatt
24      vbat="SM_PMSM_eHS_SiC_MOSFET/SM_master/Battery model/Constant2/Value"
25      r.SetParametersByName(vbat, i)
26
27      #Set output file name id
28      id="SM_PMSM_eHS_SiC_MOSFET/SM_master/to workspace/Simu_ID/Value"
29      id2="SM_PMSM_eHS_SiC_MOSFET/SS_slave/to workspace/Simu_ID/Value"
30      r.SetParametersByName(id2, i)
31      r.SetParametersByName(id, i)
```

Konfigurazio guztiak egin ondoren, modeloa eta erabiltzaile-kontsola kargatu behar dira. Behin hori eginda, berriro ere modeloaren egoera konprobatu eta bistaratu behar da. Alde horretatik, `while` aginduari esker modeloaren egoera `MODEL RUNNING` den bitartean modeloaren simulazio-denbora eta simulazioaren portzentajea bistaratuko dira. Modeloaren egoera aldatzekotan, kodea `while` begiztatik irtengo da. `while` instrukzioaren exekuzio bakoitza amaitzean 100 segundoz gelditu ko da kodea, prozesamendu guztia ondo gauzatzeko.

```
32      #Run model
33      r.Execute()
34      #Execute Matlab user interface
35      r.ExecuteConsole()
36
37      #Check model state
38      (modelState, realTimeMode) = r.GetModelState()
39      print("Model state: "+ str(modelState))
40      while  modelState == 8:
41
42          #Check model state
43          (modelState, realTimeMode) = r.GetModelState()
44          if modelState != 8:
45              print ("Modeloaren egoera honako hau da: "+ str(modelState))
46          else :
47              #Check time value
48              timeValue =
     r.GetSignalsByName("SM_PMSM_eHS_SiC_MOSFET/SM_master/Fleet-BEV-Cycle/Driving cycle/Clock/port1")
49              percentage=(timeValue/SIMU_DURATION)*100
50              print ("Current time:  "+ str(timeValue) + "portzentajea:" + " %"  + str(percentage))
51          time.sleep(100)
```

Amaitzeko, kontsola gelditu eta berrabiarazi, eta proiektua itxi behar da.

```
52  finally:
53     r.StopConsole()
54      time.sleep(5)
55      r.ResetConsole()
```

[3]Emaitzak ezberdindutako izenak eta identifikagarriak diren .mat formatuko fitxategietan gordetzeko simulaketa bakoitza amaitzen denean.

```
56      time.sleep(5)
57
58  r.CloseProject()
59  print("Proiektua itxi da")
```

## 5.4. Lortutako emaitzak

### 5.4.1. Automatizazio-emaitzak

Bi simulazioen emaitzak aztertu baino lehen, garrantzitsua da RT-lab programaren leiho guztiak ondo ezagutzea. 5.1. irudian 4 leiho eta 2 botoi azpimarratu direla ikus daiteke. 1. leihoan RT-lab plataforman kargatu diren proiektuak ageri dira. Lan honetan, *Endika_multisimulation* eta *SM_PMSM_eHS_SIC_MOSFET* proiektuak landu dira. *SM_PMSM_eHS_SIC_MOSFET* proiektuaren ondoan agertzen den U8028833 serie-zenbakiak laborategian dagoen OP4510 simuladoreari erreferentzia egiten dio. Landu nahi den proiektuaren egoera hobeto ezagutzeko, proiektuaren izenean bi klik eginez, 5.2. irudian ikusten den leihoa irekiko da. Eta bertan, nahi izanez gero, eskuzko exekuzioa edo berrabiarazketa gauzatu daiteke [16].

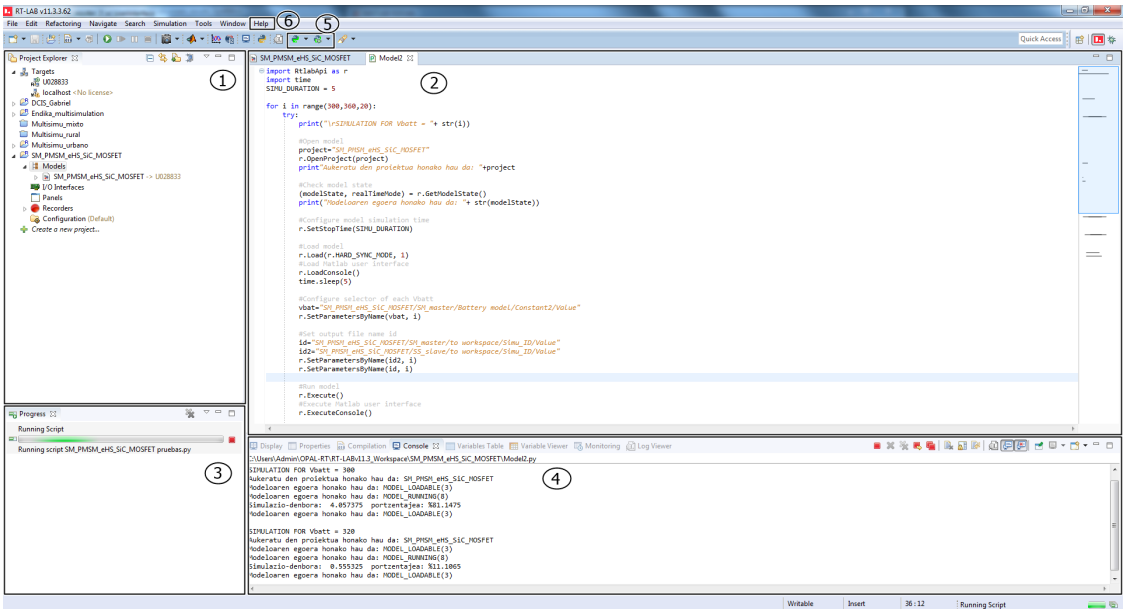
2. leihoan SM_PMSM_eHS_SIC_MOSFET izeneko proiektuari zuzendutako script-a ikus daiteke. Ikusten denez, pantaila-argazki hori egiterako orduan, 5 segundoko simulazio-denbora definitu egin da, nahiz eta, benetako simulazio-denbora 785 segundokoa izan. 3. leihoan martxan dauden proiektuen script-en egoerak ikus daitezke. Kasu honetan, bakarrik proiektu bat matxan dagoenez, proiektu horren script-aren egoera ikus daiteke. 4. leihoan erabiltzaile-kontsola ikus daiteke, bertan ikusten diren datuak script-ean programatu dira.

5. puntuan ageri diren bi botoiek script-aren arazketa eta script-a martxan jartzeko balio dute. 5.3. irudian ikusten den moduan ezkerreko botoia sakatuz gero, martxan jar daitezkeen 5 script-ak agertuko dira. Martxan jarri nahi den script-a horietako bat ez izatekotan, `run configurations` sakatu behar da eta ondoren, urdinez dagoen `browse...` botoia klikatuz, aukeratu nahi den script-a ordenagailuan bilatu behar da.

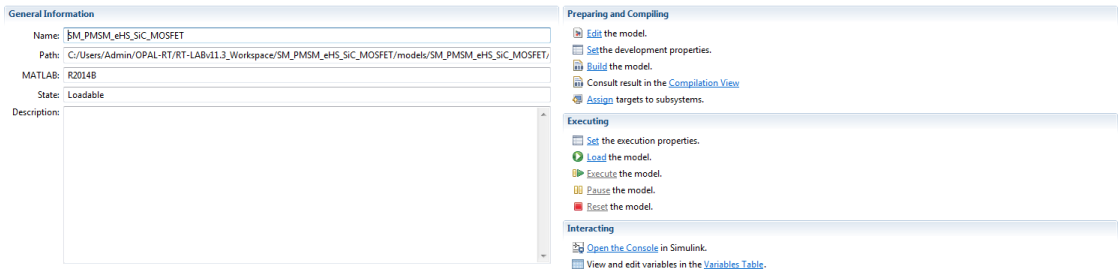
Azkenik, arazorik edukitzekotan, 6. puntuko `Help` botoia sakatu behar da. Aurreko atalean, bi proiektuen script-ak azaldu dira. Hala ere, 5.4. irudian lehenengo simulazioaren script-a fluxu-diagrama baten bidez laburtuta dago eta 5.5. irudian erabiltzaile-kontsolan bistaratutako datuak ageri dira.

### 5.4.2. Ibilgailuaren propultsio-sistemaren denbora errealean lortutako emaitzak simulaketa konkretu batetan

Jarraian, modeloa denbora errealean baldintza konkretu batetan (erregulatzaileen irabaziak zehaztuz eta modulazio-teknika jakin bat aukeratuz) simulatzean lortutako emaitzak erakusten dira. Alde horretatik, simulazio-baldintzak zehazterakoan gida-ziklo estandarizatu jakin bat kargatu da. Alde horretatik, motorraren biraketa-abiadura erakusten du 5.6. irudiak; aldiz, baldintza horien menpe motor elektrikoak (iman iraunkorreko motor sinkrono bat) sortzen duen momentu elektromagnetikoa erakusten du 5.7. irudiak. Azkenik, hori guztia ahalbideratzen duen korronte-kontrolaren emaitzak erakusten dituzte 5.8. eta 5.9. irudiek.

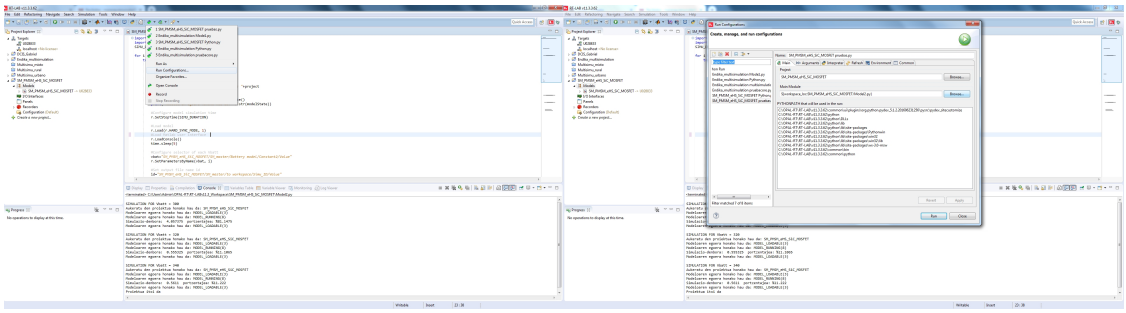



**5.1. irudia**: RT-Lab plataforma.

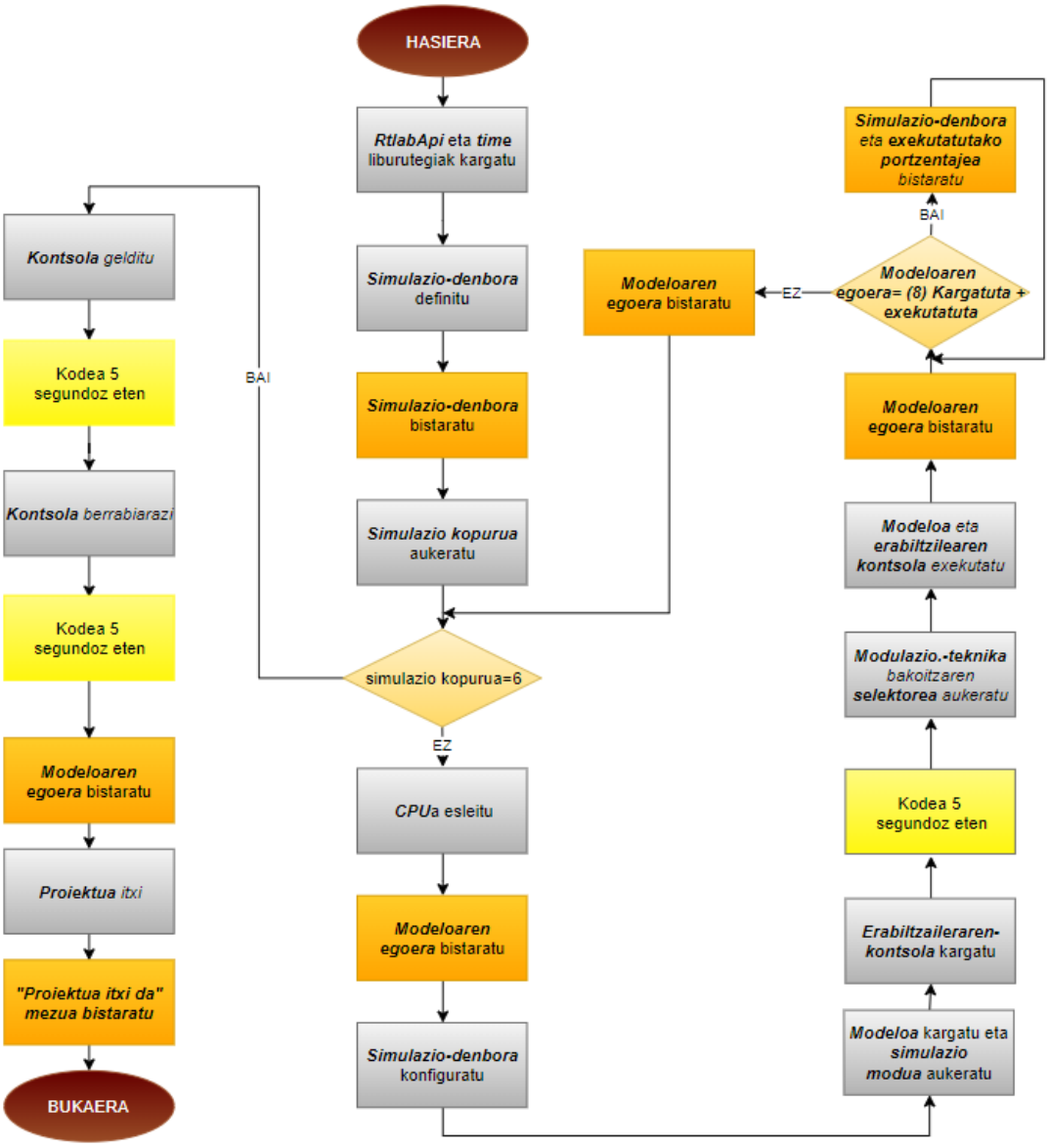



**5.2. irudia**: SM_PMSM_eHS_SIC_MOSFET proiektuaren informazio orokorra.

Beraz, automatizazioa ongi egiten da, eta simulaketa bakoitzean lortutako emaitzak gailuaren disko gogorrean gordetzen dira, simulaketa guztiak amaitzerakoan FTP protokolo bidez OP4510 gailutik Ethernet konexioa erabiliz erabitzailearen ordenagailu pertsonalera deskargatzeko.



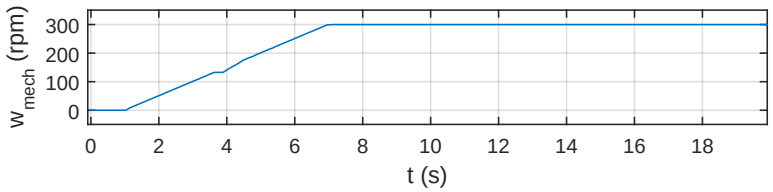
**5.3. irudia**: Script-a martxan jartzean erabiltzaile-kontsolan lortutako emaitzak.



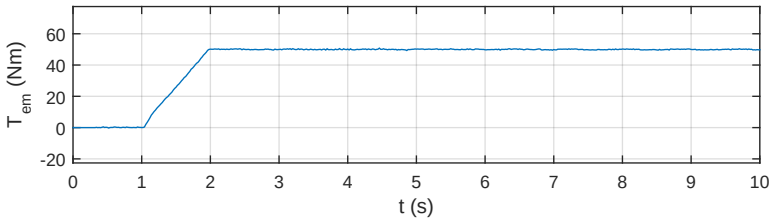

**5.4. irudia**: Simulazio azeleraturako fluxu-diagrama.

```
Modelo bakoitzaren simulazio-denbora honako hau da: 0.5

Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
1. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.051516 portzenjea: %10.3032
Simulazio-denbora: 0.42581 portzenjea: %85.162

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
2. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.00782 portzenjea: %1.564
Simulazio-denbora: 0.385541 portzenjea: %77.1082

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
3. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007859 portzenjea: %1.5718
Simulazio-denbora: 0.384747 portzenjea: %76.9494

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
4. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007786 portzenjea: %1.5572
Simulazio-denbora: 0.384226 portzenjea: %76.8452

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
5. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007635 portzenjea: %1.527
Simulazio-denbora: 0.383379 portzenjea: %76.6758

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
3. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007859 portzenjea: %1.5718
Simulazio-denbora: 0.384747 portzenjea: %76.9494

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
4. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007786 portzenjea: %1.5572
Simulazio-denbora: 0.384226 portzenjea: %76.8452

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
5. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007635 portzenjea: %1.527
Simulazio-denbora: 0.383379 portzenjea: %76.6758

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Aukeratutako proiektuaren izena honako hau da: Endika_multisimulation
6. simulazioa
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_PAUSED(7)
Modeloaren egoera honako hau da: MODEL_RUNNING(8)
Simulazio-denbora: 0.007921 portzenjea: %1.5842
Simulazio-denbora: 0.385436 portzenjea: %77.0872

Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Modeloaren egoera honako hau da: MODEL_LOADABLE(3)
Proiektua itxi da
```

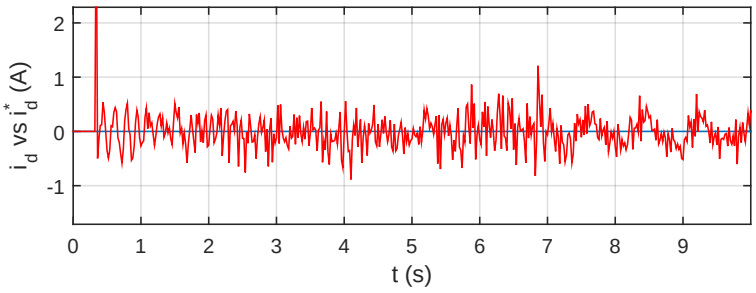
**5.5. irudia**: Simulazio azeleraturako erabiltzaile-kontsolatik jasotako erantzunak.



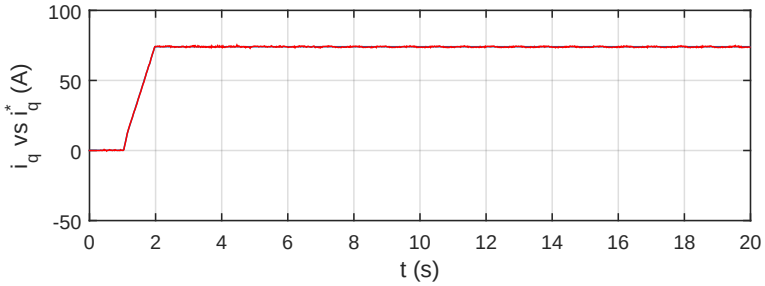

**5.6. irudia**: Abiadura angeluarraren bilakaera denbora errealeko simulazioan.




**5.7. irudia**: Momentu elektromagnetikoaren bilakaera denbora errealeko simulazioan.




**5.8. irudia**: d-ardatzeko korronte-kontrolaren emaitzak.




**5.9. irudia**: q-ardatzeko korronte-kontrolaren emaitzak.

# 6. kapitulua

# Ekarpenak

Lan honetan egindako garapen teknikoez gain, artikulu zientifiko baten erredakzioan parte hartu da, non egindako ekarpena azaltzen den, besteak beste. Artikulu zientifiko hori nazioarteko kongresu batetara bidali da, eta onartu egin dute:

**Autoreak:** Markel Fernandez, Edorta Ibarra, Endika Robles, Oihane Cuñado, Maite Aranguren, Iñigo Kortabarria, Yahia Bouzid.

**Izenburua**: FPGA and CPU based real-time simulation platform for EV propulsion system analysis under driving cycles.

**Kongresua:** Conference on Design of Circuits and Integrated Circuits 2019 (DCIS)

**Agerkaria:** Proceedings of DCIS'19

**Tokia:** Bilbo, Data: 2019/11/20-2019/11/22

# 7. kapitulua

# Ondorioak

Lan honetan azaldu den bezala, ibilgailu elektrikoen teknologia beharrezkoa izango da etorkizunean aldaketa klimatikoari aurre egiteko. Alde horretatik, ibilgailu elektrikoen arazoetako bat autonomia da. Bestalde, kontrol algoritmo egokiak diseinatu behar dira ibilgailu horietarako. Testuinguru horretan simulazioa beharrezkoa da diseinu-prozesuaren lehenengo etapetan kontrol-algoritmoak arazteko, eta baita ere ibilgailu horien propultsio-sistemak ongi aztertzeko eta diseinu edota dimentsionamendu erroreak aurkitzeko.

Ibilgailu elektriko baten propultsio-sistemaren modelatzea eta simulazioa konplexua da. Gainera, kasu batzuetan denbora luzeak simulatu nahi dira, eta hori ez da posible ordenagailu normalak edota MIL prozedurak erabiltzen direnean. Horregatik, gaitasun altuko gailu digitalak erabiltzen dira gero eta gehiago. Gure kasuan, OPAL-RT-ren OP4510 gailua erabili da. Hala ere, ikusi denez, egokia da simulazioak automatizatzea, baina hori OP4510 plataforman egiteko beharrezkoa da Python lengoaia erabiltzea.

Garatutako lanak erakutsi digu Python bidezko automatizazioa gauzatzea erraza dela hasieratik beste goi mailako lengoaia informatikoak menperatuz gero. Alde horretatik, ikasleak esperientzia du C++ programazio-lengoaian eta, horri esker, inplementazioa nahikoa azkarra izan da.

Bestalde, OPAL-RTk eskaintzen duen Python liburutegiari esker posible da automatizazio-prozedimendua asko sinplifikatzea. Hala ere, zailtasunak aurkitu dira ez automatizazioan bertan, baina bai interfaze grafikoa eta horrek erabiltzailearekin duen datu trukaketa programatu nahi izan denean. Ondorioz, komenigarria litzateke OPAL-RT fabrikatzaileak eskaintzen duen liburutegiko funtzionalitateak gehitzea, interfaz grafikoa hobetzeko.

Azkenik, esan beharra dago finkatutako helburuak lortu direla, eta hemendik aurrera APERT taldeak erabilgarri duen automatizazio-programa batzuk dituela. Lan hau gida bezala erabiliz eta sortutako programetan aldaketa minimoak gauzatuz, posible da taldeak dituen automatizazio-beharrizanak betetzea.

# Bibliografia


[1] International Energy Agens (IEA) (Ed.), $CO_2$ Emissions from Fuel Combustion , International Energy Agens (IEA), 2018.

[2] W. Todts, E. D., $CO_2$ emisions from cars: the fact, Tech. rep., European Federation for Transport and Environment AISBL (2018).

[3] CEIDA, Uraren eta airearen kutsadura, CEIDA, 2017, Ch. 3, pp. 2–74.

[4] CEIDA, Arazo globalak: kutsadura atmosferikoaren ondorioak, CEIDA, 2017, Ch. 13, pp. 2–33.

[5] EU greenhouse gas emissions from transport increase for the second year in a row (2017).

[6] F. Balleste, Vigilancia de riesgos ambientales en Salud Pública. El caso de la contaminación atmosférica, Unidad de Epidemiología y Estadística. Escuela Valenciana de Estudios en Salud-EVES.

[7] L. Kumar, S. Jain, Electric propulsion system for electric vehicular technology: A review, Renewable and Sustainable Energy Reviews (29) (2014) 924–940.

[8] A. R. Plummer (Ed.), Model-in-the-Loop Testing, 2006.

[9] V. Rau Aparow and K. Hudha and Z. Abd Kadir and M. Mansor and N. Hafizah Amer, Model-In-the-Loop Simulation of Electronically Controlled Pitman Arm Steering Mechanism for Armored Vehicle, International Symposium on Control Systems (ISCS).

[10] S. S. Noureen, N. Shamim, V. Roy, S. B. Bayne, Real-Time Digital Simulators: A Comprehensive Study on System Overview, Application, and Importance, IJRE.

[11] P. M. Visser, M. A. Groothuis, J. F. Broenink (Eds.), FPGAs as versatile configurable I/O devices in Hardware-in-the-Loop Simulation*) , 2004.

[12] Z. Tong, H. Zhang, Z. Ye, J. Han, RCP-based HIL Simulation and Control for 2-DOF Tracking Robot of Maneuvering Target, in: International Asia Conference on Informatics in Control, Automation and Robotics, 2009, pp. 121–125.

[13] W. Chaaban, M. Schwarz, B. Batchuluun, H. Sheng, J. Börcsök, A Partially Automated HiL Test Environment for Model-Based Development Using Simulink and OPC Technology, IEEE.

[14] J. Belanger, P. Venne, J.-N. Paquin, The what, where and why of real-time simulation, pp. 37–49.

[15] O.-R. Technologies, OP4510 RT-Lab-RCP/HIL Systems, User Guide, azken bisita: 2018/10/24 (2016).
URL https://www.opal-rt.com/wp-content/themes/enfold-opal/pdf/L00161_0413.pdf

[16] Opal-RT, Quick Start Guide. OPAL-RT Technologies.
URL https://www.opal-rt.com/wp-content/themes/enfold-opal/pdf/L00161_0582.pdf

# II. atala

# LANAREN GARAPENEAN JARRAITUTAKO METODOLOGIA

# 8. kapitulua

# Eginbeharren, faseen, ekipamenduaren eta prozeduren deskribapena

Jarraian, lanaren helburua lortzeko beharrezkoak diren eginbeharrak eta lortu beharreko mugarriak aipatzen dira.

**Espezifikazio eta eskakizunen definizioa**

- **Deskribapena**: Ataza honetan RT-Lab plataforma erabiliz sortuko den script-aren baldintzak finkatuko dira.
- **Zeregina**: Proiektuaren baldintza funtzional guztiak zehaztea.
- **Entregagarriak**: Espezifikazioen dokumentua (proiektuaren zuzendariak entregatu beharrekoa).

**Artearen egoera**

- **Deskribapena**: Simulazio moduak aztertu eta aplikaziorako egokiena zein den aukeratu. Horretarako, literatura zientifikoa erabiliko da, hala nola, unibertsitateko bibliografia zientifikoen datu-baseetarako harpidetza garrantzitsuenak (IEEEXplore eta Scopus).
- **Zeregina**: Simulazio mota egokiak aukeratu.
- **Entregagarriak**: Artearen egoerari buruzko txostena, gradu amaierako lan honetan kapitulu gisa agertzen dena.

**Formakuntza**

- **Deskribapena**: Proiektu hau aurrera eramateko, ezinbestekoa da ikasitako irtenbideen balidatzea eta aplikazioa garatzeko beharrezkoak diren laneko tresnak ezagutzea. Formakuntzaren helburua RT-Lab plataforman eta OpalRTren OP4510 simulazio tresnetan trebetasun aurreratuak garatzea da.

- **Zeregina**: Trebetasun aurreratuak lortzea RT-Lab plataforman eta OpalRTren OP4510an. Horretarako, RT-Lab-eko Yahia Bouzid adituaren formakuntza jasoko da.
- **Entregagarriak**: Ez dago.

**Automatizazio-script-en inplementazioa**

- **Deskribapena**: Modelatuko diren soluzioak definituko dira, behin artearen egoera aztertuta, ondoren, RT-Lab plataformari esker eta Python lengoaia erabiliz beharrezko bi script-ak garatuko dira.
- **Zeregina**: Bi simulazioen script-ak RT-Lab plataforman garatu.
- **Entregagarriak**: Egindako script-ak eta jarraitutako pausuen txostena (dokumentuan zehar azalduta).

**Egiaztapena**

- **Deskribapena**: Denbora errealeko eta simulazio azeleraturako automatizazio-scripten simulaketa.
- **Zeregina**: Simulazioen konprobaketa eta emaitzen lorpena.
- Entregagarriak: Emaitzen simulazioa (dokumentuan zehar azalduta).

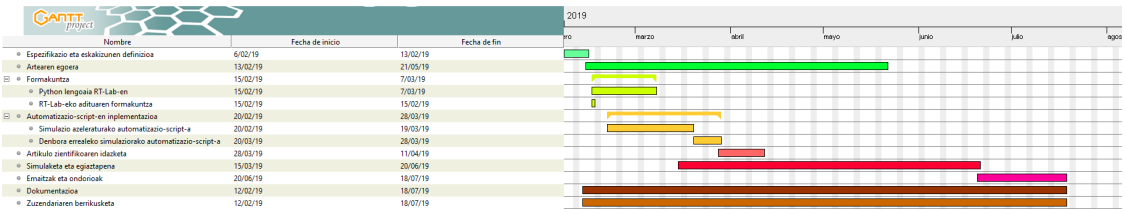
**Dokumentazioa**

- **Deskribapena**: Gradu amaierako lana dokumentatzea eta artikulu bat idazten laguntzea du helburu betebehar honek. Hurrengo pasuak bete ahala egingo da.
- **Zeregina**: Gradu amaierako lana eta artikulua dokumentatu.
- **Entregagarriak**: Gradu amaierako lanaren dokumentua eta artikulua.

# 9. kapitulua

# Gantt-diagrama

Lan hau 2019ko otsailaren 6an hasi zen eta uztailaren 18an bukatutzat eman da. Aurrera eramaterako orduan, 24 aste erabili dira, guztira 320 ordu. Eta aste bakoitzean 14 orduko bataz besteko lan-ordu bete dira. 8. kapituluan zehaztutako faseetatik, artearen egoeraren eta simulaketa eta egiaztapenaren faseak iraupen luzeenak izan dituzten faseak izan dira, hain zuzen ere, 14 aste iraun dute. Hori guztia dokumentazioaren fasea eta zuzendariaren berrikusketaren fasea kontuan hartu barik. Zuzendariarekin egindako bilerak bi astetik behin egin dira. Gainera, otsailaren 15ean, telematikoki RT-Lab-eko Yahia Bouzid adituaren formakuntza jaso zen, formakuntza honetan Python lengoaiaren oinarrizko aginduak eta eskuragarri dauden egiturak landu ziren. 9.1. irudian ikus daiteke lan honen fase guztien epeak. Hala ere, 9.1. taulan fase guztien asteak zehaztu dira.




**9.1. irudia**: Proiektuaren Gantt diagrama.

**9.1. taula**: Fase bakoitzari esleitutako aste-kopurua.

| **Fasea** | **Iraupena** *(asteak)* |
|---|---|
| Espezifikazio eta eskakizunen definizioa | 1 |
| Artearen egoera | 14 |
| Formakuntza | 3 |
| Automatizazio-script-en inplementazioa | 5 |
| Artikulo zientifikoaren idazketa | 2 |
| Simulaketa eta egiaztapena | 14 |
| Emaitzak eta ondorioak | 5 |
| Dokumentazioa | 23 |
| Zuzendariaren berrikusketa | 23 |

# 10. kapitulua

# Proiektuaren aurrekontua

Hurrengo tauletan proiektua gauzatzeko beharrezkoa den aurrekontua laburbiltzen da, bertan, langileen eta materialen kostuak zehazten dira. 10.3. taulan ikusten den moduan, lan honen kostu totala 39.757,24 eurokoa izan da.

**10.1. taula**: Barne orduak.

| **Langileak** | **Ordu tasa (euro/h** | **Orduak (h)** | **Kostua** | |
|---|---|---|---|---|
| Ingeniari Teknikoa | 30 | 320 | 9.600,00 | |
| Proiektu zuzendaria | 50 | 25 | 1.250,00 | |
| | | | **Guztira** | 10.850,00 euro |

**10.2. taula**: Amortizazio taula.

| **Inbert./Akt. finkoa** | **Hasierako kost. (euro)** | **Bizitza erabilg. (urte)** | **Erabiltze denb. (aste)** | **Amortizazioa** |
|---|---|---|---|---|
| Ordenagailua | 800 | 4 | 24 | 92,3 |
| Latex | 115 | 1 | 22 | 48,65 |
| Matlab/Simulink | 1980 | 3 | 18 | 228.46 |
| OP4510 | 28500 | 4 | 22 | 3014,42 |
| | | | **Guztira** | 3383,83 |

**10.3. taula**: Proiektuaren kostu osoaren taula.

| **Kontzeptua** | **Kostua** |
|---|---|
| Eskulana | 10.850,00 |
| Tresneria | 31.395 |
| Amortizazioak | 2.487,76 |
| **Guztira** | 39.757,24 |