eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Gradu Amaierako Lana

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

RailDriver eta ATO sistemen integrazioa Train Simulator 2021 bideo-jokoan

Josu Loidi Astiazaran

Zuzendariak

Rosa Arruabarrena Santos

Jon Goya Mendiluce

2021eko irailaren 13a

Laburpena

Proiektu hau enpresak dituen sistemak probatzeko lagungarria izan daitekeen produktu bat sortzeko helburuz burutuko da, izan ere merkatuan dagoen tren simulatzaile errealistena integratuko baita bertako produktu propioekin. Honela, proiektu honetan teknologia desberdinen arteko zubiak sortzeko eta inplementatzeko soluzioa proposatzen da, hain zuzen ere, *Train Simulator 2021* bideo-jokoaren eta *ATO* (Automatic train operation) sistemaren artean. Honetaz gain, erabilgarriak diren tresna gehiago ere gehitzen dira zubi honetan; batetik, *RailDriver* kabina kontroladorea, eta bestetik, ikusmen artifizialeko *PER* eredua.

Lan hau *CAF Signalling* enpresan garatutako proiektu bat da, eta proiektuaren nondik norakoak, lan hau burutzeko eman diren pausoak, lortutako helburuak eta proiektuaren kudeaketa osoa azalduko da txosten honetan.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1 Sarrera

Dokumentu hau Josu Loidi Astiazaranen Informatikako Ingeniaritzako Gradu Amaierako Lanaren memoria da, *CAF Signalling* enpresan burututakoa, Rosa Arruabarrena tutore akademikoarekin gidaritzapean eta Euskal Herriko Unibertsitateko (EHU) Donostiako fakultatean landuta.

Bertan aurkituko dira proiektua aurrera eramateko beharrezkoak izan diren pausu guztiak. Hala, dokumentu honen helburua proiektuaren fase guztiak ahalik eta xehetasun handienez azaltzea izango da, eta batez ere, erabili diren metodologia eta estrategiak deskribatzea.

1.1 Testuingurua

Produktu industrial seguru eta eraginkorra sortzeko beharrezkoa da kalitatea bermatzea, eta horretarako probak derrigorrezkoak dira. Gainera, sektore askotan, burutzen diren proba horiek baliagarriak izan daitezke produktuaren etorkizuna baldintzatzeko, edo beste modu batera esanda, derrigorrezkoa da proba txukun eta ikusgarriak egitea sortzen den produktua salgarria izan dadin.

1.2 Tren industria

Tren industria arlo handi eta zabala da: trenak ekoiztu, mantendu eta eguneratu behar dira etengabe, eta zentzu horretan, gaur egun oso presente dagoen alorra automatizazioa da. Trenen automatizazioa, izenak dioen bezala, trena gidaririk gabe gidatzean datza, eta sistema oso ahaltsua behar da horretarako, bai software nahiz hardware aldetik.

1.3 *CAF Signalling*, *ATO* eta *PER*

Proiektu hau *CAF Signalling* enpresan garatu da. *CAF Signalling CAF* (*Construcciones y Auxiliar de Ferrocarriles*) Taldearen filiala da, eta trenbide-seinaleztapen eta -kontrolerako irtenbide integralak diseinatzen eta eskaintzen ditu, azpiegituren eremua eta ontziratutako materiala barne [1]. Enpresa honek, berezko teknologiako

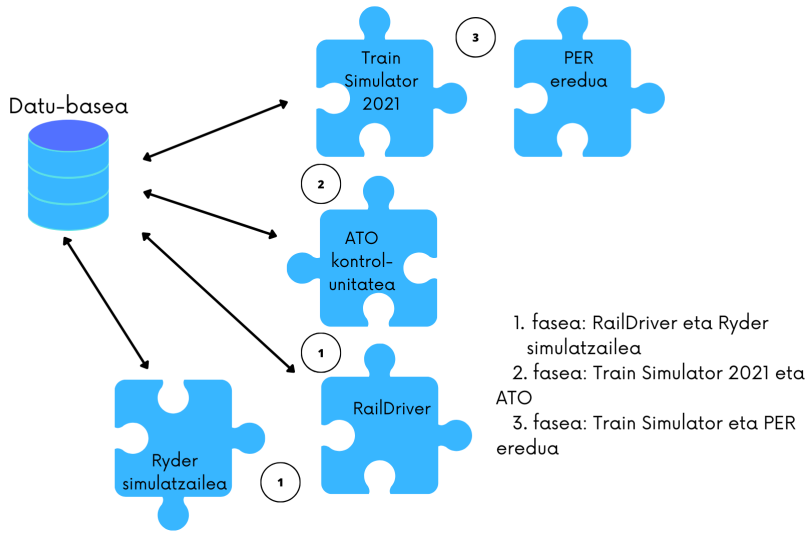
produktu aurreratuak ditu, ezagutza itzela dauka trenbide-zirkulazioko seinalez-tapenean eta proiektu konplexuak kudeatzeko eta gauzatzeko gaitasun handia erakusten du.

Hortaz, enpresako ezagutza teknologikoa handia da, eta horren froga da *ATO* (Automatic Train Operation) sistema [2]. *ATO*a trenaren funtzionamendua automatizatzen laguntzen duen gailu bat da, eta hainbat software integratuz lortzen da. Gainera, trena gidatzeaz arduratzeaz gain, trenaren abiatzea, gelditzea eta ateen funtzionamendua automatizatzeaz ere arduratzen da; eta nahiz eta definizioz ez den segurtasunaz arduratzen, gaur egungo sistema moderno askok *ATC* (Automatic Train Control) eta *ATP* (Automatic Train Protection) sistemak integratuta dituzte, hots, segurtasunaz ere arduratzen direnak. Gaur egun 4 *ATO* automatizazio maila daude (ikus 1.1 taula), UITP edo Garraio Publikoaren Nazioarteko Batasunaren arabera, eta nazioartean erabiltzen denez, maila bakoitzak hainbat estandar jarraitu behar ditu.

Bestalde, *PER* (ingeleseko *perception*etik) sistema edo eredua enpresan garatutako ikusmen artifizialeko eredu bat da, eta tren bati jarritako kameretako irudiak hartuz gai izan nahi da inguruko oztopo eta seinaleak irakurtzeko, ondoren *ATO*ak trena ondo gidatu dezan. Izan ere, sistema hau gabe, trena aurredefinitutako agin-duen baitan mugitzen da, inguruari arretarik jarri gabe.

1.4 Proiektuaren deskribapena

Orokorrean, proiektua hiru fasetan banatuko da, eta fase bakoitzak ingurune desberdin batean eragingo du eta sistema desberdinak lotuko ditu, 1.1 irudian ikusten den moduan.



1.1 Irudia: Proiektuaren zati desberdinen loturak

Hala ere, egin beharrekoa azaltzen hasi aurretik, beharrezkoa izango da aipa-

tuko diren hainbat termino eta tresna zer diren azaltzea. Gainera, esan *ATO* eta *PER*en garapena eta mantenua *CAF Signalling* enpresako izen bereko azpi taldeek kudeatzen dutela, eta beraz, proiektuaren garapenean bi azpi talde hauekin egingo dela lan. Hala ere, *ATO*ak izango du garrantzirik handiena, eta beraz, lanik gehiena talde honen barruan burutuko da.

Hona hemen proiektuan konbinatu eta integratuko diren hainbat tresnen deskribapen laburra:

1. **ATO**. Trenak automatikoki gidatzeko sortu den gailu bat da. Hainbat software zati integratuta osatzen da eta bi zatitan banatzen da, *ATO trackside* eta *ATO on board*. *ATO on board* trenean kokatzen den software eta hardware zatia da, eta jasotako agindu eta profilen arabera trena gidatzeaz arduratzen da. Bestetik, *ATO trackside* bulego zentralean kokatzen da, eta *ATO on board*i gidatzeko aipatutako beharrezko informazioa bidaltzeaz arduratzen da. Funtziona dezan, datu-base partekatu bat erabiltzen da aldagai askorekin, eta tresna desberdinek hauen balioak aldatzen dituzte behar den heinean.

2. **PER**. *CAF Signalling* enpresan garatutako eredua da, ikusmen artifiziala

Automatizazio maila	Tren-operazioa	Deskribapena
GoA 1	Eskuzkoa	Treneko makinista batek abiatzea eta gelditzea, ateen funtzionamendua eta batbateko larrialdi edo desbideratzeen erabilera kontrolatzen ditu.
GoA 2	Erdi-automatikoa (STO)	Abiatzea eta gelditzea automatizatuta daude, baina gidari batek ateak maneiatzen ditu; behar izanez gero trena gidatzen du eta larrialdiez arduratzen da. ATO sistema asko GoA 2 dira.
GoA 3	Gidaririk gabe (DTO)	Abiaraztea eta gelditzea automatizatu egiten dira, baina tren-laguntzaile batek ateak operatzen ditu eta trena gidatzen du larrialdietan.
GoA 4	Zaintzarik gabe (UTO)	Abiatzea eta gelditzea, ateen funtzionamendua eta larrialdien kudeaketa guztiz automatizatuta daude trenean langilerik behar izan gabe. Geltoki guztiek nasa ateak izan beharko lituzkete.

1.1 Taula: *ATO*ko automatizazio mailak

erabiliz tren baten inguruan dauden oztopo eta seinaleak detektatzeko gai da, eta behar denean *ATO*ari abisatu diezaioke, semaforo edo seinale baten eraginez trena gelditu behar bada.

3. **Ryder**. *CAF Signalling* enpresan garatutako interfaze grafikorik gabeko tren simulatzailea da. Software tresna honen bitartez tren baten portaera simulatzen da, eta *ATO* tresna probatzeko erabiltzen da, benetako tren batean aplikatu aurretik.

4. **Train Simulator**. *Dovetail Games* enpresak garatutako bideo-jokoa da, eta trenen simulatzaileen munduan erreferentea da duen errealismoagatik, bai grafiko aldetik eta baita mundu errealeko ibilbide ugari eta hainbat tren mota baitu. Urtero bertsio berria argitaratzen da, baina aurreko urteko bideo-jokoaren eguneratze bat da eta guztiz bateragarriak dira denak beraien artean. Beraz, *Train Simulator 2021* eta *Train Simulator* gauza bera dira.

5. **RailDriver**. *P. I. Engineering* enpresak sortutako tren-kabina kontrolatzailea da [3]. Hainbat palanka eta botoi programagarri ditu, eta tren baten gidatzea errealistagoa egiten du. Gaur egun merkatuko tren kontrolatzailerik ezagunena da eta guztiz bateragarria da *Train Simulator 2021* bideo-jokoarekin.

1.4.1 *RailDriver* kabina kontrolatzailea eta *Ryder* simulatzailea integratzea.

Hasteko, lehen zati honetan bi tresna erabiliko dira, *RailDriver* kabina-kontrolatzailea eta *Ryder* simulatzailea. Horrela, zati honetako helburua *RailDriver* kontrolatzailea erabiliz *Ryder* simulatzailea maneiatzea da. Horretarako, kontrolatzailearen hainbat botoi eta palanka konfiguratu beharko dira funtzio desberdinak burutzeko (*Travel Direction*, *Cab Selection*, *ATO Engage*, *TBL*, *Skip*, ...), datu-base partekatu batean aldaketak eginez. Gainera, kontrolatzaileak LED pantaila bat dauka nahi dena bertan azaltzeko.

1.4.2 *Train Simulator 2021 ATO* sistemarekin integratzea

*ATO*az baliatuz eta bideo-jokoak eskaintzen duen errealismoa aprobetxatuz, atal honetan *ATO*ak *Train Simulator 2021*eko trena gidatzea lortu nahi da. Horrela, hemendik aurrera, ez litzateke *Ryder* simulatzailea erabiliko, izan ere kasu honetan trenaren portaera simulatuko lukeena jokoa bera izango litzatekeelako. Haatik, *RailDriver* kontrolatzailea erabili ahal izango da, nahi denean trenaren kontrola eskuzkoa izan dadin.

1.4.3 Aurreko zatiari *PER* sistema integratzea

Behin aurreko zatiak amaituta, *Train Simulator 2021* bideo-jokoak eskaintzen duen errealismoa haratago eraman nahi da, eta horretarako egokia ikusten da ikusmen

artifizialeko sistema bat integratzea trenbidean eta inguruan egon daitezken arazo eta oztopoak ikusi eta hauei aurre egiteko. Eredu hori enpresa garatzen ari den *PER* eredua izango da, eta helburua bideo-jokoa eredu hau entrenatzeko irudiak sortzeko gai izatea da. Beraz, bideo-jokoko irudiekin *streaming* bat egingo da *PER* eredura, oztopoak edo ezbeharrak detektatu ditzan (semaforoak gorrian, trenbideko eragozpenak, ...). Ondoren, denbora errealean, oztopo edo pasa ezin den seinaleren bat baldin badago, *ATO*ari abisatuko zaio trena gelditu dezan, eta larrialdietako balazta aktibatuko da. Software garapenari dagokionez, *PER* ereduaren garapena Gradu Amaierako Lanetik kanpo egongo da, izen bereko taldeak garatuko baitu.

2 Helburuak

Proiektua diseinatzerako orduan, hainbat helburu jarri dira mahai gainean, eta bai enpresaren aldetik eta baita arlo akademiko nahiz pertsonalaren aldetik ere proiektu honek erronka anitz aurkezten dizkigu.

2.1 Enpresaren helburuak

Enpresari dagokionez, helbururik garrantzitsuena proposatutako proiektua ondo burutzea eta baliagarria izatea da.

- *RailDriver* kabina kontrolatzailea martxan jarri eta *Ryder* nahiz *Train Simulator*ekin bateragarria izatea.
- *Train Simulator 2021* jokoa baliatuz, enpresako *ATO* eta *PER* sistemak jokoarekin integratzea.
- *ATO* ingurunea probatzeko ingurune grafiko errealista bat izatea, bideojokoak eskaintzen duen errealismoa aprobetxatuz. Hala, erakustaldiak egiteko agertoki erakargarri bat sortuz.
- *PER* eredua entrenatzeko irudiak sortzeko ahalmena izatea, irudi errealen beharrik gabe.

2.2 Helburu pertsonal eta akademikoak

Bestetik, helburu pertsonalei dagokienez, garrantzitsua da akademikoki egin den bilakaera aztertzea eta burututako gradua amaitzeko aprobetxatzea.

- Graduko lau urtetan zehar barneratutako kontzeptu eta ideiak proiektu baten bidez erakustea.
- Gradua amaitzeko, egindako lan guztia bildu dezakeen proiektu handi bat garatzea.

- Enpresa munduan murgiltzea eta bertako lan egiteko moduak eta metodologiak ezagutzea.

3 Erabilitako teknologiak

Proiektu hau burutzeko hainbat teknologia erabili dira, eta batzuk ezagunak izan arren, beste batzuk kontrolatzea erronka handia izan da.

3.1 C++03

$C++$ munduan gehien erabiltzen den programazio lengoaietako bat da, eta C lengoaian oinarrituta dago. Objektuen manipulazioa ahalbidetzen duten mekanismoak C programazio-lengoaiara zabaltzea izan zen haren sorkuntzaren asmoa. Zentzu horretan, objektuetara bideratutako lengoaien ikuspegitik, $C++$ lengoaia hibridoa da.

2003. urteko $C++$ da enpresan erabiltzen den programazio-lengoaia nagusia. Lengoaiaren bertsioa oso zaharra da, baina orain arte egindako guztia bertan eginda dagoenez eta funtzionatzen duenez, erabiltzen jarraitzen da. Esan beharra dago, graduan zehar ikusi ditugula bai C eta baita $C++$, eta hortaz, ez da lan handia izan lengoaia hau kontrolatzea. Hala ere, kontuan hartu behar da oso goi mailako kodea dela, asko autogeneratua, eta hau dena ongi ulertzeko lan handia egin dela.

3.2 QML lengoaia

*Javascript*ten oinarritutako lengoaia bat da *QML*, eta orokorrean erabiltzailearen interfazeari bideratutako aplikazioak eraikitzeko erabiltzen da. Hala ere, beste edozertarako ere erabili daiteke, eta kasu honetan trenak jarraitu beharreko ibilbide eta arauak definitzeko erabiltzen da (*ATO*a martxan jartzeko pausuak, zein segmentu izango dituen, baimendutako gehienezko abiadurak, baliza eta geltokien posizioak, ...).

3.3 Visual Studio Code

*Visual Studio Code Microsoft*ek *Windows*, *Linux* eta *macOS*-entzat garatutako kode editorea da. Arazketarako euskarria, *Git*-en kontrol integratua, sintaxi-nabarmentzea,

kodearen amaiera adimenduna eta kode-birfaktorizazioa ditu. Pertsonalizatu ere egin daiteke; beraz, erabiltzaileek editorearen gaia, teklatuko lasterbideak eta lehentasunak alda ditzakete. Doakoa eta kode irekikoa da, baina deskarga ofiziala software pribatiboarekin dago, *Microsoft*en lizentziapean.

Kode-editore hau erabiltzeko arrazoi nagusiena erraza, sinplea eta osoa dela da. Luzapen asko daude erabilgarri, lengoaia eta fitxategi-formatu askorako euskarria ematen du eta bertan konpilatu eta arazteko aukera duenez, lanak asko errazten ditu.

3.4 Git

Git programazio munduan gehien erabiltzen den bertsio-kontrolerako softwarea da. Oso sistema boteretsua eta osatua da, eta egokia sortzen den kodea modu eraginkor batean gordetzeko. Enpresaren kasuan, *GitLab*-en oinarritutako web-ostatu bat erabiltzen da, eta bertan gordetzen dira taldeak egindako proiektu guztiak, besteon eskuragarri.

3.5 Redmine

Redmine proiektuak kudeatzeko web-aplikazioa da, *Ruby on Rails framework*a erabiliz idatzia. Web-aplikazio honen bitartez enpresako langile bakoitzak dituen atazak kudeatu daitezke eta hauen jarraipen zehatza egin daiteke. Enpresan oso erabilia da plataforma hau, eta egunero eguneratu behar da ataza bakoitzean eman den denbora eta honen aurrerapena. Horrela, planifikazioa ondo doan ala ez ondorioztatu daiteke era azkarrago eta bisualago batean.

3.6 Train Simulator

Dovetail Games enpresak garatutako tren simulatzaile jokorik arrakastatsu eta ospetsuena da *Train Simulator*. Urtero bertsio berri bat argitaratzen da, eta aurrekoaren eguneraketa moduan har daiteke. Ofizialki, mundu errealeko ibilbide asko barne ditu, baina garatzaileei bideratutako tresna asko eskaintzen dituenez, erabiltzaile adituenek haien inguruko trenak eta ibilbideak sortzen dituzte, eta sarean erabilgarri jartzen dituzte edonork erabili ditzan. Gainera, horrelako edukiak benetako lekuen errepresentazio leialak izaten dira, eta hortaz, oso ibilbide eta tren errealistak erabiltzeko aukera ematen du jokoak.

Hori hala izanik, oso aukera egokia izan daiteke mundu errealean kostu handiegiak dituzten probak egiteko, edo lortzeko zailak diren egoerak simulatzeko (eguraldi edo egoera zehatz bat, trenbideko oztopoak, ...).

3.7 RailDriver kontrolatzailea

Tren simulazioetan gaur egungo merkatuan dagoen kabina kontrolatzailerik ospetsu eta erabiliena da *RailDriver*. Esperientzia errealistago bat emate aldera, kontrolatzaile honen bitartez benetako tren baten palanka eta botoiak imitatu nahi dira, eta tren simulatzaile bat erabiltzen denez, integratua izateko teknologia egokia da.

4 Proiektuaren kudeaketa

Proiektua burutzerako garaian, atal oso garrantzitsua da proiektua bera antolatzea. Alde batetik, egin behar dena planifikatu behar baita, hau da, aurreikusi egin behar da zer egin behar dugun eta zenbat denbora beharko dugun egin beharreko bakoitzean; eta beste alde batetik, sistema ordenatu bat planifikatu behar dugu, bai erabiliko den materiala gordetzeko eta baita bilerak eta elkarrizketak antolatzeko ere.

4.1 Irismena

Proiektu honetako azken garapena bi programek osatuko dute nagusiki, baina baita hauek funtzionatzeko beharrezkoak izango diren konfigurazio fitxategiek ere. Konfigurazio fitxategiei dagokienez, aurrerago azalduko dira zein izango diren eta zertarako beharko diren, eta programei dagokienez, aipatutako ezaugarri guztiak beteko dituzte, eta bi edo tresna gehiagoren arteko zubia kudeatzea izango da hauen helburua. Honetaz gain, azken fasean integrazio bat burutuko da bideo-jokoko irudiak *PER* eredura bidaltzeko denbora errealean, eta probak egiteko ingurune bat egingo da, baina bloke oso baten zati bat besterik ez da izango Gradu Amaierako Lan honetan egin beharrekoa.

Horrela, hauek izango dira garatuko diren bi aplikazioak:

1. **RailController**: *RailDriver* eta *Ryder* simulatzailearen integrazioa burutuko du.

2. **TrainSimATO**: *Train Simulator* bideo-jokoaren eta *ATO* sistemaren arteko integrazioa inplementatuko da.

4.2 Emangarriak

Emangarrien zerrenda zati desberdinetan bana dezakegu beraien arteko erlazioan oinarrituta. Honela, bi zatitan antolatu dira emangarrien zerrenda:

- **Enpresari**. Enpresari eman beharreko emangarriak kodearekin erlazionatutakoak izango dira, hau da, garatu diren programak eta hauei dagokien erabilera-gida labur bat (ikus 10.2 eranskina). Horrela, behin inplementazioa amaitu denean, enpresako *ATO* eta *PER* taldeei egindako lana aurkeztu eta emango zaie.

- **EHUri eta proiektuko zuzendariari**. Unibertsitateari proiektuaren garapena deskribatzen duen memoria hau eta posterra entregatuko zaizkio. Horrez gain, defentsan erabiliko den aurkezpena entregatuko zaio zuzendariari. Aurreikuspenen arabera, memoria eta posterra uztaila amaitzerako entregatuko da, eta aurkezpena proiektuaren defentsa egiten den egunerako, hau da, irailaren hirugarren asterako.

4.3 Proiektuaren antolaketa

Aipatu moduan, proiektu bat egiten hasi aurretik, antolatu egin behar da, eta zati desberdinetan banatu behar da, beharrezkoak izango diren lan-paketeak eta atazak definituz lana hobeto kudeatzeko. Horrela, ataza bakoitzaren dedikazioa aurreikusiko da, hau da, egin behar den lan bakoitzak zenbat denbora beharko duen estimatzen da, eta proiektu honetarako, 4.2 irudiko *Gantt* diagramarekin irudikatu daiteke. Bestalde, ataza garrantzitsuenei mugarriak ipini behar zaizkie (ikusi 4.1 taula), batez ere, egiten den lana egunean doan ala ez ikusteko, eta proiektua ondo eta garaiz amaitzeko.

Laburtuz, esan daiteke proiektua martxoaren hasieran hasi eta uztailaren erdialderako amaitzea espero dela. Horrela, enpresako lan egunak errespetatuko lirateke eta egun bakoitzean 5 orduko lanaldiak egiteko asmoa izango da. Tartean aste eta erdiko geldialdia egingo da aste santuetako oporrak direla eta, eta arazoren bat izango balitz proiektuaren amaiera uztaileko azken asteraino luzatzeko aukera egongo litzateke. Beraz, denborari dagokionez ustekaberen bat gertatzen bada (arriskuen atalean aurreikusitakoak ala beste edozer), bi asteko tartea emango da proiektua ongi bukatzeko.

Ondorengo zerrendan definituko dira beharrezkoak izango diren lan-paketeak eta hauei dagozkien atazak, eta 4.1 irudiko LDE diagramaren bitartez aipatutako ataza eta lan-pakete guztiak laburbiltzen dira.

1. **Proiektuaren plangintza eta jarraipena definitu**
   Atal honetan proiektuaren plangintzaren eta aurreikusten den jarraipenaren inguruko atazak aurkituko ditugu.

   - Proiektuan egin beharrekoa definitu.
   - Jarraipenerako *excel* orriak prestatu eta erabiliko den "Agenda"aplikazioa konfiguratu.

2. **Ingurunea prestatu**
   Lan egiteko ingurunea prestatu behar da, hau da, beharrezkoak izango zaizkigun programa eta tresnak instalatu behar dira, eta honako ataza hauetan bana daiteke lan-pakete hau:

   - Programak instalatu.
   - Erabiliko den kodea eta tresnak eskuratu.
   - Informazio sistema martxan jarri.

3. **Ikasketa-prozesua**
   Aipatu da askotan enpresako teknologiak erabili behar direla proiektu hau burutzeko, eta horretarako beharrezkoa da ikasketa-prozesu bat martxan jartzea. Gainera, zati oso garrantzitsua da *ATO*a, eta beharrezkoa izango da beraz honek nola funtzionatzen duen eta jarraitzen dituen arauak ondo zein diren ikastea. Hortaz, honako ataza hauek burutu beharko dira:

   - *ATO*aren funtzionamendua eta prozesua ikasi.
   - Enpresako tresnak ezagutu eta erabiltzen ikasi.
   - Erabiliko diren teknologien inguruan ikertu.

4. ***RailController*en garapena**
   Lan-pakete honetan, proiektuari dagokion lehen zatiaren garapenarekin hasiko da. Hasteko, aztertu egin beharko da komunikaziorako datu-basea nola erabili beharko den eta zein aldagai beharko diren, eta ondoren kontrolatzailearen aginduekin lotu. Hortaz, ataza hauekin gelditu gara:

   - Datu-baseko beharrezko aldagaiak identifikatu.
   - *RailDriver*aren *API*a ikertu.
   - Programaren garapena burutu.
   - Probak egin eta demoa prestatu.
   - *RailController*en dokumentazioa garatu.

5. ***TrainSimATO*ren garapena**
   Proiektuaren bigarren fasea barne hartzen duen lan-pakete honetan, garrantzitsua izango da beharrezko tresnetatik datuak nola atera identifikatzea, bai komunikaziorako eta baita agertokia edo eszenarioa sortzeko ere. Hori lortzeko, honako ataza hauek aurreikusten dira:

   - *ATO*ak gidatzeko zein aldagai behar dituen identifikatu.
   - *Train Simulator*en *API*a ikertu.
   - Programaren garapena burutu.
   - Erabiliko den ingurune edo agertoki batetik datuak atera.
   - Aukeratutako agertokiaren errepresentazioa *QML*n egin.

- Probak egin eta demoa prestatu.
- *TrainSimATO*ren dokumentazioa garatu.

6. **PER ereduaren integrazioa**
   Garapenaren azken fasea da hau, eta bertan enpresari gaur egungo proiektu batean laguntzeko lan egingo da, bideo-jokoa prestatuz ikusmen artifizialeko eredua probatu eta entrenatzeko, hain zuzen ere. Horretarako, benetan erabiliko den agertoki bat baliatuz. Beraz, honako ataza hauetan laburbiltzen da lan-pakete hau:

   - *Train Simulator* bideo-jokoa aztertu hainbat funtzio egin daitezkeen ikusteko.
   - Ingurunea prestatu ereduari jokoko irudiak bidaltzeko.
   - Larrialdietako balaztaren inplementazioa burutu.
   - *PER*ren dokumentazioa garatu.

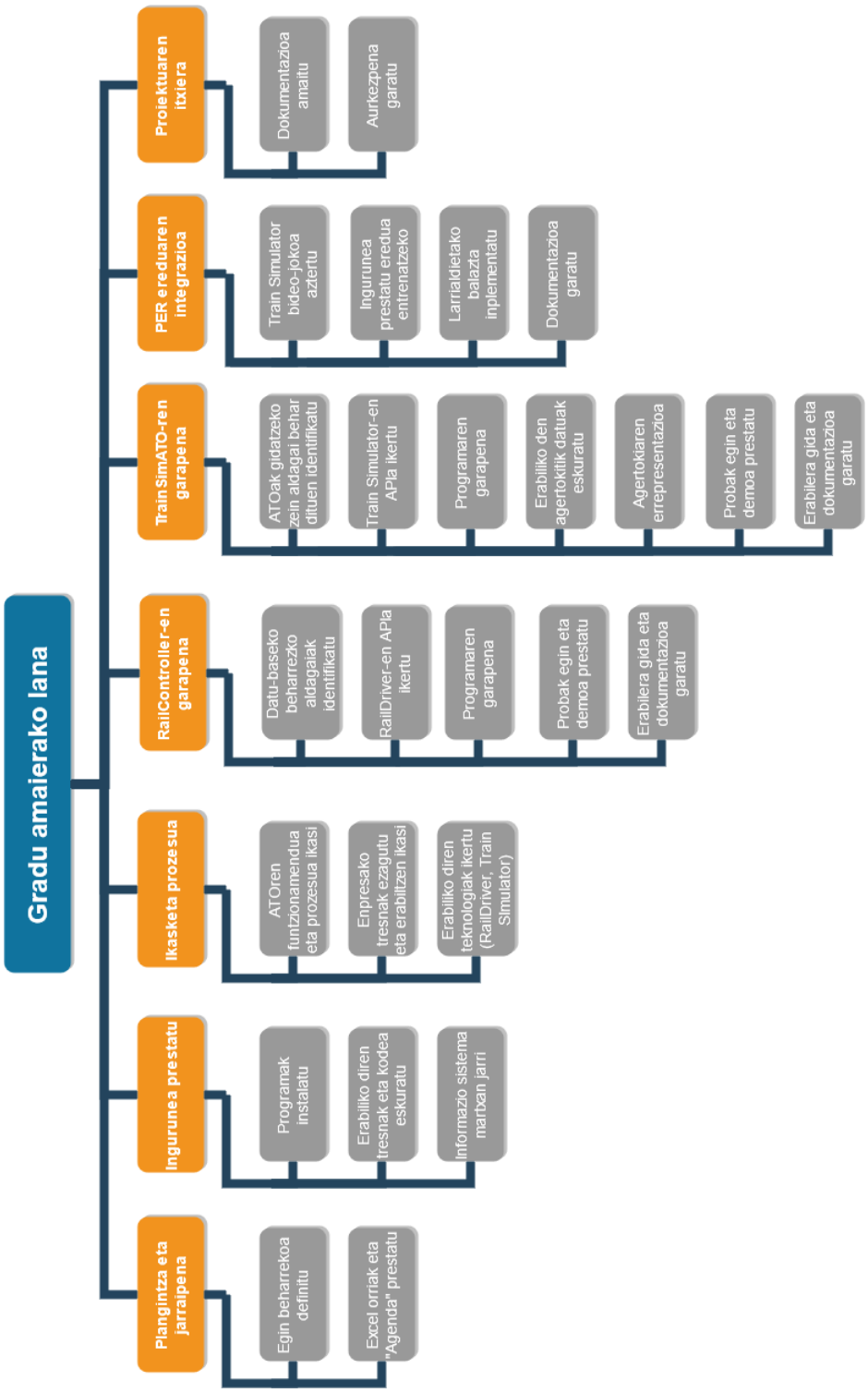
7. **Lanaren itxiera**
   Orain arte egin dena errepasatu eta zuzentzeko erabiliko da lan-pakete hau, batez ere, dokumentazioa amaitzeko eta egingo den aurkezpena prestatzeko, honako ataza hauekin:

   - Dokumentazioa amaitu.
   - Aurkezpena garatu.

Horrela, behin burutu behar diren atazak azaldu ondoren, ataza bakoitzari estimatzen zaion ordu kopurua definitu behar da, eta proiektuak osorik burutzeko beharrezkoa izango den denbora aurreikusi, 4.2 taulan ikusi daitekeen moduan.

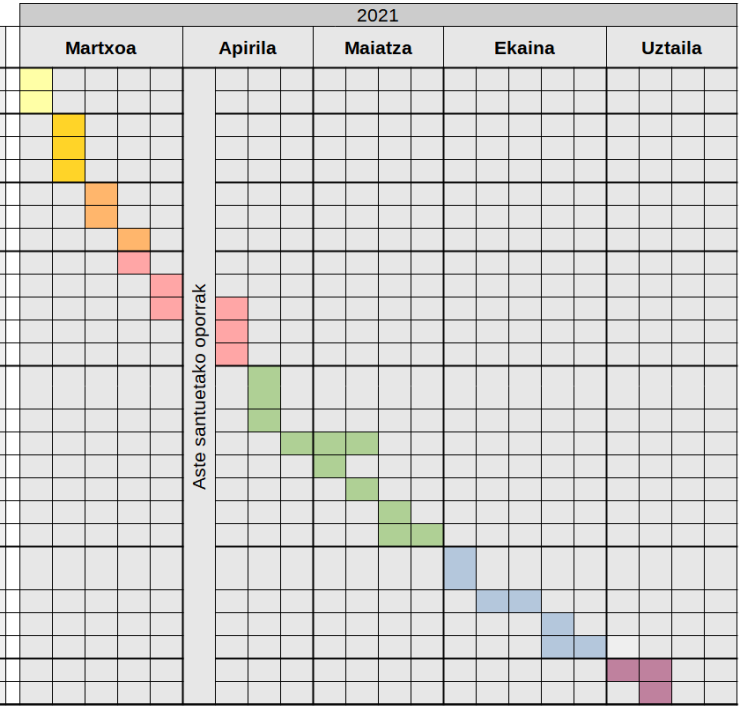
Ataza esanguratsuenak	Mugarria
RailController-en garapena	2021/04/13
TrainSimATO-ren garapena	2021/05/14
PER-en integrazioaren garapena	2021/06/24
Dokumentazioaren behin-behineko bertsioa	2021/07/16
Dokumentazioaren behin-betiko bertsioa	2021/07/30

4.1 Taula: Ataza esanguratsuenen mugarriak



4.1 Irudia: Proiektuaren LDE diagrama

Lan-paketearen izena	Atazaren izena	Hasiera data	Amaiera data
Plangintza eta jarraipena	Egin beharrekoa eta dokumentazioa definitu	2021/03/01	2021/03/05
	Excel orriak prestatu	2021/03/05	2021/03/05
Ingurunea prestatu	Programak instalatu	2021/03/08	2021/03/08
	Erabiliko den kodea eta tresnak eskuratu	2021/03/09	2021/03/11
	Informazio sistema martxan jarri	2021/03/11	2021/03/12
Ikasketa prozesua	ATOaren funtzionamendua eta prozesua ikasi	2021/03/15	2021/03/18
	Enpresako tresnak ezagutu eta erabiltzen ikasi	2021/03/18	2021/03/19
	Erabiliko diren teknologiak ezagutu	2021/03/22	2021/03/23
RailController-en garapena	Datu-baseko beharrezko aldagaiak identifikatu	2021/03/24	2021/03/25
	RailDriveraren APIa ikertu	2021/03/26	2021/03/29
	Programaren garapena	2021/03/30	2021/04/13
	Probak egin eta demoa prestatu	2021/04/14	2021/04/14
	RailDriverRyderaren dokumentazioa garatu	2021/04/15	2021/04/16
TrainSimATO-ren garapena	ATOak gidatzeko zein aldagai behar dituen identifikatu	2021/04/19	2021/04/20
	Train Simulatoren APIa ikertu	2021/04/20	2021/04/23
	Programaren garapena	2021/04/26	2021/05/14
	Erabiliko den eszenariotik datuak atera	2021/05/05	2021/05/07
	Eszenarioaren errepresentazioa QMLn	2021/05/10	2021/05/14
	Probak egin eta demoa prestatu	2021/05/14	2021/05/20
	TrainSimulatorATOren dokumentazioa garatu	2021/05/21	2021/05/28
PER ereduaren integrazioa	Train Simulator aztertu funtzionamenduak identifikatzeko	2021/05/31	2021/06/04
	Ingurunea prestatu eta eredua martxan jarri	2021/06/07	2021/06/16
	Larrialdietako balaztaren inplementazioa	2021/06/17	2021/06/24
	PERen dokumentazioa gehitu	2021/06/25	2021/07/02
Proiektuaren itxiera	Dokumentazioa amaitu	2021/07/05	2021/07/16
	Aurkezpena garatu	2021/07/14	2021/07/16



4.2 Irudia: Proiektuaren Gantt diagrama

4.4 Informazio sistema

Proiektuaren izaera pribatua da, eta hortaz, ez da inongo *cloud* plataforma irekietan gordeko. Zerbitzu hauek, *Google Drive* esaterako, dokumentazioari dagokion babeskopiak egiteko, edota plangintzako *LDE* diagrama eta lan egindako ordu kopuruaren erregistroa gordetzeko erabiliko dira. Lan egun eta ordu kopurua erregistratzeko, *excel* fitxategi bat baliatuko da, non egunero sartuko diren ordu kopurua agertuko den. Horretaz gain, eguneko ordu bakoitza zertan emango den jakiteko, *Agenda* izeneko *Windows*eko aplikazio bat erabiliko da [4], oso modu erraz eta azkarrean idatzi eta kontsultatzeko.

Bestalde, dokumentazioaren garapena LaTeX bidez egingo da, eta lanak errazteko, *Overleaf* [5] izeneko online zerbitzua erabiliko da. Horrela, LaTeX ez da instalatu beharko, eta garatutako dokumentazioa online gordeko da edozein tokitatik atzitzeko aukerarekin.

Hala ere, erabiliko den material guztiaren segurtasun-kopia bat egitea aurreikusten da lokalean. Astean behin, garatuko den dokumentazioaren kopia deskargatu eta lokalean gordeko da *GrAL_Josu* izeneko karpetan, honako egitura honekin:

→ Memoria

→ Irudiak eta taulak

→ Bileren aktak

Kodeari dagokionez, esan moduan, izaera pribatukoa denez, eta hortaz, enpresako beste proiektu guztien antzera, *GitLab* plataforma erabiliko da. Honen bitartez,

Ataza	Ordu-kopurua
Prestaketa eta dokumentazioaren hasiera	20
Formakuntza	25
*RailController*en garapena	70
*TrainSimATO*ren garapena	160
PER ereduaren integrazioa	80
Dokumentazioaren amaiera	15
Aurkezpenaren garapena	5
Guztira	375

4.2 Taula: Proiektuko lan-paketeen denbora aurreikuspena

bertsio-kontrol sendo bat ziurtatuko da, eta gainera dena online gordeko da enpresako zerbitzarietan. Kopiak egiteko irizpideen artean, saiatu beharko da ahal den gehienetan kopia bat egiten, eta egiten diren aldaketak zehazki dokumentatzen, bai garatzailearentzat baita irakasleak edo enpresako zuzendariak ikusi nahiko balute eskuragarri utziz. Horretaz gain, garapena lokalean egingo denez, lokalean ere gordeko da kopia bat gutxienez, atzigarriagoa eta uneko jarduera eraginkorragoa izan dadin.

Jarraipenaren *excel* taulari dagokionez, hilero bat gordeko da, hilabete horretan burututako egun eta ordu kopuruekin. Dokumentu horien izenburua *hilabetea_orduak.xlsx* izango da. Dokumentu hau enpresari ere igorri behar zaio, lanean emandako egun eta ordu kopurua jakin ditzan.

4.5 Komunikazio-sistema

Pandemia dela eta, enpresari dagokion alorretan aurreikusten da komunikazio guztia online egingo dela. Hortaz, enpresako zuzendariarekin nahiz beste lankideekin burutuko diren bilera edo elkarrizketak *Microsoft Teams* erabiliz egingo dira, eta askotan tresna honek berak eskaintzen duen aukerarekin programatuko dira bilerak. Garrantzitsua izango da lankideekin ere komunikatzea, enpresako lan bat denez, enpresako ingurugirora egokitzeko eta batez ere eginda dagoen kodea eta programen funtzionamendua hobeto ulertzeko.

Bestalde, *EHU*ko zuzendariarekin eskolaz kanpoko komunikazioa posta bidez egingo da, eta honela adostuko dira bilera egunak ere. Hala ere, ez da bileren daten inguruko periodikotasun finkorik ezarriko, eta horren ordez, beharraren arabera adostuko dira bilerak. Gainera, bilera horiek printzipioz aurrez-aurrekoak izatea espero da, irakaslearen bulegoan bertan.

4.6 Lan-metodologia

Garatuko den proiektua enpresako *ATO* taldearen barruan burutuko da gehienbat. Taldearen barruan burutuko den arren, garapen pertsonal eta inkrementala egingo da, hau da, software garapenaren bizi-zikloa inkrementala izango da. Eredu honekin, proiektua hainbat zatitan deskonposatzen da, eta horietako bakoitzak funtzionalitatearen zati bat hornitzen du, proiektuaren betekizunei dagokienez. Betekizun hauek lehentasuna izango dute eta dagokien gehikuntzak lehentasun ordenaren arabera ematen dira.

Horrela, garapen inkremental hau hiru zatitan banatuko da, proiektuaren zati bakoitzeko bat (*RailController*en garapena, *TrainSimATO*en garapena eta *PER* ereduaren integrazioa). Hainbat alorretan lankideen beharra izango da, batez ere erabiliko diren enpresako tresnen zalantzak argitzeko. Era horretan, egunero goizeko 8:30etan bilera laburrak egingo dira, bakoitzak aurreko egunean egin duena besteei adierazteko eta elkarbanatzeko.

Aipatutako ataza horiek hobeto kudeatzeko helburuz, *Redmine* tresna erabiliko da. Bertan, langile bakoitzari esleitutako atazak ikusi ahalko dira, eta ataza horiek egin ahala, bakoitzaren egoera, egindakoaren ehunekoa eta dedikatutako ordu kopurua gehitu beharko dira.

Ordutegiari, dagokionez, aipatu den moduan egunero 5 orduko lanaldiak egingo dira orokorrean (goizeko 8:00tatik eguerdiko 13:00ra), eta proiektu guztia tarte honetan burutzea izango da helburua. Batzuetan, bilerak eta beste langileen erabilgarritasuna tarteko, lanaldia luzeagoa izan daiteke, baina ez da ohikoena izango.

4.7 Kalitate-kudeaketa

Proiektua garatu ahala, eta kalitatea bermatze aldera, honako irizpide hauek hartuko dira kontuan:

- **Komunikazioak eraginkorrak eta egokiak izatea**. Oso garrantzitsua da egiten diren komunikazio guztiak egoki eta arin egitea, denbora errealean egiten direlako kalkuluak bai *ATO*an eta baita *Train Simulator* jokoan. Beraz, arreta asko jarriko da honek ondo funtziona dezan.

- **Aukeratutako agertokian gidatzea ongi egitea**. Bi agertoki aukeratuko dira trena automatikoki gidatu dezan *ATO*ak, eta gidatze hori ondo egitea garrantzitsua izango da. Gidatzea egokia izan dadin kontuan hartuko da abiadura ahalik eta linealena izatea eta agertoki horri dagozkion muga guztiak errespetatzea (abiadura, gradienteak, geltokiak., ...).

- **Mugarriak errespetatzea**. Hasieran egindako planifikazioa errespetatzea oso garrantzitsua da, batez ere egin behar den dena egiteko denbora izateko, eta baita zuzentzeko aukera izateko ere.

Beraz, kalitatea neurtzeko, adierazle batzuk definituko dira. Kalitatea bi motatan banatuko da (egokia eta bikaina), eta nahiz eta kalitate onena lortzen saiatuko den, kalitate egokia lortzea onartuko da, betiere kalitate onena ezin bada lortu.

Kalitate egokiari dagokionez, tresnen arteko komunikazioa ongi burutu beharko da eta derrigorrezkoak diren datu guztiak irakurri eta idatziko dira behar direnean. Gainera, trena ondo gidatuko da, errespetatu beharreko muga guztiak errespetatuz, eta amaitzeko, sortuko diren bi programak sendoak izango dira, eta aurreikusitako errorerik gertatzen ez den bitartean hauen exekuzioa ez da etengo, edo beste modu batera esanda, programa hauek ez dira *hilko*.

Bestalde, kalitate bikainari dagokionez, sistema osoa ez da trena gidatzera soilik mugatuko, eta *Train Simulator* bideo-jokoko gauza gehiago automatizatzeko aukera izango du (ateak, argiak eta abar).

- **Kalitate egokia**

  - Komunikazioa arina eta eraginkorra da.
  - Gidatzea ongi burutuko da.
  - Sistema sendoa eta segurua izango da, hau da, ez da inola ere sistema hilko.

- **Kalitate bikaina**
  - *Train Simulator*eko trenarekin egiten den elkarrekintza gidatzetik haratago joango da (ateen kontrola, argiak, ...).
  - *RailDriver* kontrolatzailearen botoiak programatuko dira bideo-jokoan gauza gehigarriak egiteko (argiak, klaxona, ...).

4.8 Arriskuak

Proiektu guztietan bezala, arriskuak egon daitezke, eta arrisku horiek baldintza dezakete proiektuaren egoera nahiz garapena. Hortaz, arrisku horiek identifikatzea eta aurreikuspen bat egitea beharrezkoa da beti.

- **COVID-19 pandemia**
  Badakigu murgilduta gauden pandemia egoerak hankaz gora jar dezakeela dena momentu batetik bestera eta kontuan hartu beharreko arriskurik handiena da hau. Enpresan, bere aldetik, telelana egiteko ez dago inongo arazorik, eta horrela funtzionatzeko aukera eta medio asko ditu, baina gaixotzeak benetan baldintza dezake proiektuaren garapena.

- **Ordu aurreikuspen okerrak**
  Plangintza egiterako garaian aurreikusten diren ordu kopuruak egokia izaten saiatu behar da, nahiz eta ez den erraza, batez ere enpresaren erritmora mugitu behar den proiektuetan. Aurreikuspenak handiegiak baldin badira, ez da arazo handirik suertatzen, baina aurreikuspenak azpitik badira, arrisku handia dago mugarriak ez betetzeko, eta hori arazo bat izan daiteke.

- **Enpresako edo zuzendaritzako pertsonek eragin dezaketen atzerapena**
  Horretaz gain, beste pertsonen beharra nahitaezkoa izango proiektua burutzeko, bai enpresako lankideena zein unibertsitateko zuzendariarena. Horregatik, kontuan hartu beharko dira pertsona hauen erabilgarritasuna eta haien lanen lehentasunak zein diren, proiektu honetan atzerapenak eta denbora galtzeak ekar ditzazkeelako.

- **Erabili behar diren tresnekin arazoak**
  Erabiliko diren hainbat tresna enpresakoak dira, eta oso goi mailako kodea dute askok. Horregatik, arriskutsua izan daiteke tresna hauek erabiltzerakoan denbora gehiegi ematea tresna hauek ulertzeko eta erabiltzeko.

- ***Train Simulator 2021* jokoarekin arazoak**
  Proiektuaren helbururik garrantzitsuenetakoa *Train Simulator 2021* jokoarekin komunikatzea da, eta enpresa batek garatutako joko bat denez, software pribatiboa da, eta posible da hainbat informazio lortzeko eskuragarri dagoen *API*a motz gelditzea, eta beraz, atera ezin izatea.

5 Domeinuaren deskribapena, betekizunak eta soluzioaren diseinua

Behin proiektuaren nondik norakoak azalduta, atal honetan, zati bakoitzak bete behar dituen betekizunak zerrendatuko dira, enpresaren behar eta nahien arabera, eta domeinua zehaztasun handiagoz deskribatuko da. Hala nola, soluzioaren lehen diseinu bat ere egingo da.

5.1 Domeinuaren betekizunak

Hasteko, domeinuaren betekizunak zein izango diren zehaztu behar dira, hau da, proiektuaren arrakasta definitzeko bete behar diren baldintzak edo betebeharrak azaldu behar dira. Betebehar hauek deskribatu diren hiru taldetan banatuko dira, garatuko diren hiru faseetan, zehazki. Gainera, lehentasunaren arabera daude ordenatuta, lehentasun handienetik txikienera, enpresaren beharren eta nahien arabera.

1. **RailController**
    - *A.1* Simulatzailea kontrolatzeko beharrezko botoiak konfiguratu behar dira.
    - *A.2* VarDB datu-basearen gainean aldagai egokien balioak idatzi behar dira.
    - *A.3* VarDB propioa sortu behar da Ryder gabeko inguruneetarako.
    - *A.4* ERTMS+ATO inguru nerako bateragarria egin behar da.
2. **TrainSimATO**
    - *B.1 ATO*ko kontrol komandoak *Train Simulator* jokora iritsiko dira eta honek erantzun beharko du.

- *B.2 Train Simulator* jokoak trenaren inguruko informazioa *ATO*ari helarazi beharko dio.
- *B.3 RailDriver* kontrolatzailea integratu behar da.

3. **PER-en integrazioa**
   - *C.1* Tren simulatzailearen irudiak *PER* ereduari bidali behar zaizkio.
   - *C.2* Ereduak *ATO*ari oztopoak eta seinaleak ondo ohartarazten dizkiola ziurtatu behar da.
   - *C.3 ATO*a trena gelditzera iristen ez den egoeretarako larrialdietako balazta aktibatu beharko da.

5.2 Domeinuaren deskribapena eta soluzioaren diseinua

Garatuko den software zati bakoitzak modulu desberdinei eragiten die, edo modulu berdinei eraginda ere atal desberdinetan erabiltzen dira. Horregatik, egokia eta beharrezkoa izango da zati guztiak beraien artean bateragarriak izatea, horrela, gainera, funtzionalitate gehiago lortuko dira eta aberasgarriagoa izango da. Beraz, proiektu honetan garrantzi handiena datuen kudeaketak eta kontrol unitate desberdinen arteko interoperatibitateak izango du, hau izango baita garapenaren funtsa.

Beraz, aipatu moduan, proiektuaren garapena hiru fase nagusitan banatuko da, eta bakoitzean honako modulu hauen arteko loturak inplementatuko dira:

1. *RailDriver* kabina-kontrolatzailea eta *Ryder* simulatzailearen arteko lotura.
2. *Train Simulator* bideo-jokoa eta *ATO*ren arteko lotura.
3. *Train Simulator* bideo-jokoa eta ikusmen artifizialeko *PER* ereduaren arteko lotura.

Aurretik aipatu bezala, *Ryder* simulatzailea, *ATO* kontrol-unitatea eta *PER* eredua enpresak garatutako sistemak dira, eta hauek beraien artean komunikatzeko datu-base konplexu eta handi bat erabiltzen da, *VarDB*. datu-base honetan trenak behar dituen ehunka aldagai daude gordeta, eta trenaren informazio orokorra gordetzeko (trenaren pisua, luzera, ...), trenaren egoera monitorizatzeko, gertatzen diren arazoak identifikatzeko eta gehiagorako erabiltzen dira. Horregatik, garrantzitsua izango da datu-base hau era eraginkor batean kudeatzea, eta sortuko diren software zatien atazarik handiena datu hauen kontrola izango da. Ideia orokor moduan, orain arteko sistemaren eta proiektuaren fase bakoitza deskribatzen duen sistemaren komunikazioak eta prozesuak irudikatzeko, hurrenez hurren, 5.1, 5.2, 5.3 eta 5.4 irudiak lagungarriak izan daitezke.

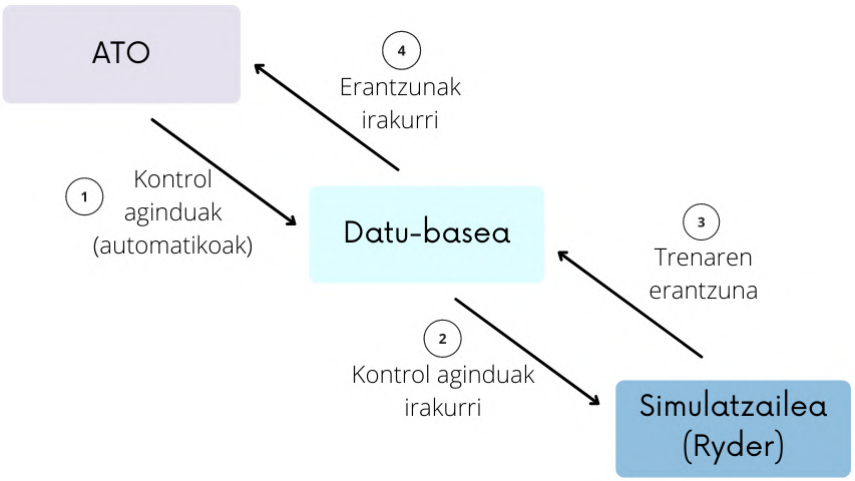
5.2.1 1. garapen fasea

Beraz, hasteko *RailDriver* kabina-kontrolatzailearekin egingo da lan. Fase honen helburua kontrolatzaile hau erabilgarri egitea da, eta datu-base elkarbanatua erabiliz, enpresako beste tresnak kontrolatzea, *Ryder* simulatzailea kasu honetan.

Kontrolatzaile honek benetako trenen kabina simulatzen du eta hainbat palanka eta botoi programagarri ditu; tartean, garrantzitsuenak diren *Reverser* (trenaren norabidea erabakitzeko, aurrera edo atzera martxa), *TBL* (trena azeleratzeko eta balaztatzeko palanka, balio positiboekin azeleratu egiten da trena eta negatiboekin balazta dinamikoa deritzona aktibatzen da, trena balaztatuz) eta balazta palanka (balazta indartsuagoa, trena gelditzeko) daudelarik. Horrela, hasteko, kontrolatzailearen *output*ak irakurri beharko dira eta ondoren datu-basean zehaztutako aldagaiei beharrezko aldaketak egin beharko zaizkie. Hala ere, probak egin nahi badira, posible da datu-base bat sortzea probak egiteko, eta horrela, existitzen diren tresnek datu-basearekin duten aurretikako hartu-emana oztopatzea ekidin daiteke, baina programaren azken bertsioan ez da datu-base berririk sortuko. Gauzak horrela, orain arteko sisteman *RailDriver*ak *ATO*a ordezkatuko du eta honen ordez bidaliko ditu trena gidatzeko aginduak. Kasu honetan, agindu guzti horiek eskuzkoak edo manualak izango dira.

5.2.2 2. garapen fasea

Ondoren, *Train Simulator 2021* bideo-jokoarekin egingo da lan. Fase honen helburu nagusia existitzen den *ATO* sistemaren bitartez bideo-jokoko trena gidatzea izango da, eta horrela, erantzun bisual ikusgarriago eta erakargarriago bat izatea. Horretarako, garrantzitsua da bideo-jokoa kontrolatzeko API bat erabiltzea, eta aipatu den *RailDriver* kontrolatzailea bideo-jokoarekin bateragarria denez, honek erabiltzen duen liburutegiaz baliatuko da proiekturako beharrezkoa izango den liburutegi propioa osatzeko. *ATO*a oso konplexua izateaz gain, hainbat estandar jarraitu behar



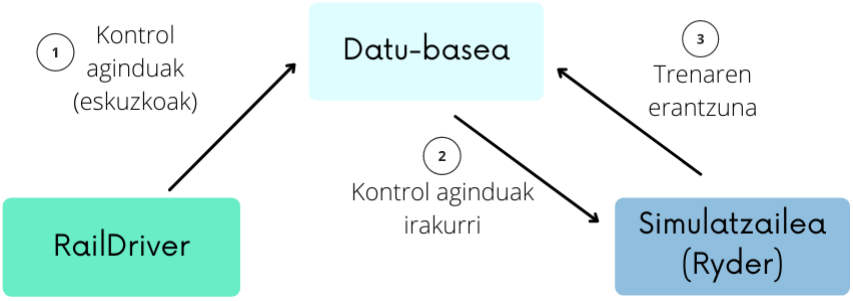
5.1 Irudia: Oinarrizko sistemaren komunikazioa

ditu [6], eta hori horrela, haren kodean ezin da ezer aldatu eta dagoenera moldatu behar da. Beraz, bi tresnen interoperatibitatea burutzeko beharrezkoa izango da hasteko bideo-jokoko trena gidatu ahal izatea, eta ondoren trenaren egoera deskribatzen duten hainbat informazio eskuratzea, abiadura, azelerazioa, posizioa eta orografia, esaterako.

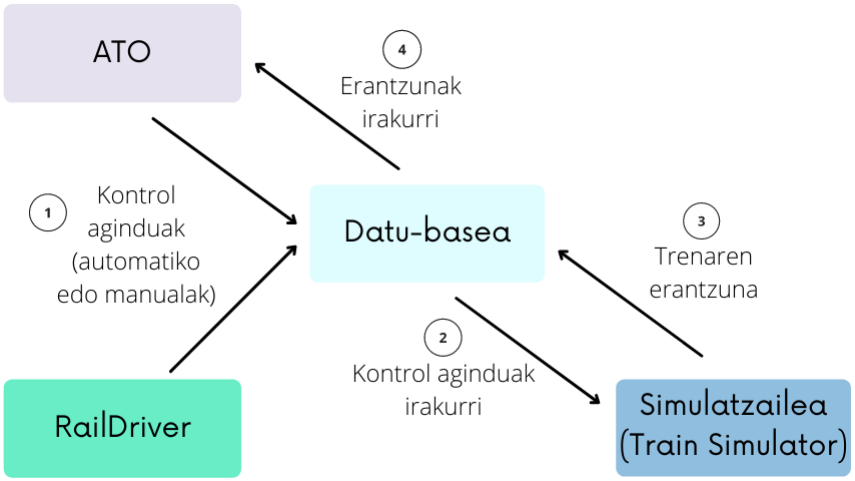
Behin *ATO*a jokoa kontrolatzeko gai denean, gidatu behar duen agertokia espezifikatu behar da, hau da, *ATO*ak jakin behar du zer distantzia egin behar duen, zein geltoki dauden eta zeinetan gelditu behar duen, abiadura mugak edo trenbidearen gradiente eta orientazioa, besteak beste. Horretaz gain, oso garrantzitsuak izango dira balizak, hauek erabiltzen baitira tren bat lokalizatzeko.

5.2.3 3. garapen fasea

Amaitzeko, orain arte sortu den sistema erabili nahi da ikusmen artifizialeko *PER* eredua entrenatu eta proban jartzeko. Horretarako, *Train Simulator 2021* bideo-jokoko irudi errealistak hartuko ditu eta agertoki zehatz batean eguraldi eta ordu desberdinak aplikatuz eredua entrenatuko da. Esan beharra dago eredu honen

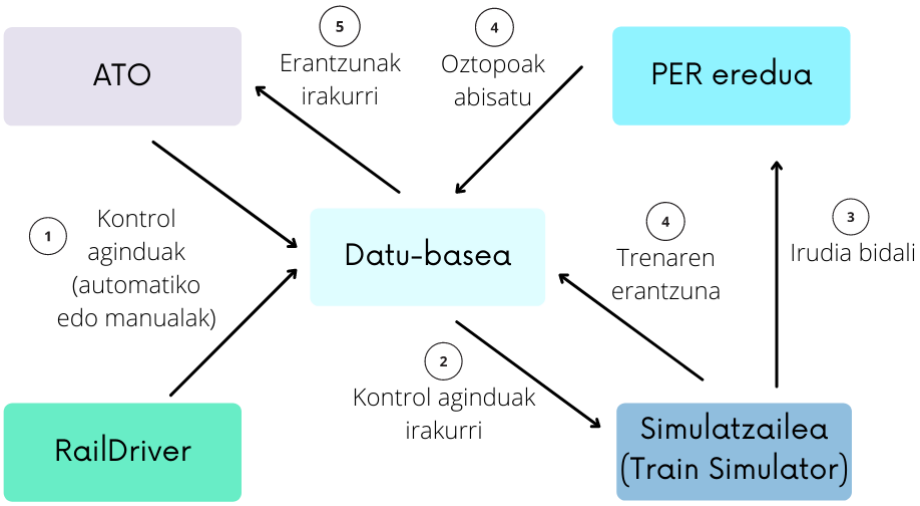


5.2 Irudia: Lehen garapen fasea errepresentatzen duen prozesuaren eskema



5.3 Irudia: Bigarren garapen fasea errepresentatzen duen prozesuaren eskema

inplementazioa proiektu honetatik kanpo dagoela, eta fase honen helburua entrenamendu eta probetarako ingurune egoki bat sortzea dela, horrela, *ATO* eta *PER* tresnak erabiliz tren baten gidatze automatikoa hobetuz. Hortaz, behin eredua martxan dagoela, ereduak trenbideko trabak (jendea, objektuak, ...) eta seinaleak (semaforoak eta geltokietako seinaleak) detektatu eta ulertuko ditu, horrela *ATO*ari abisatuz. Behin *ATO*ari abisatu ondoren, larrialdietako balazta aktibatuko da.



5.4 Irudia: Hirugarren garapen fasea errepresentatzen duen prozesuaren eskema

Amaitzeko, soluzioaren diseinu orokorra azaltze aldera, aipatu egitura bera ez dela oso konplexua izango, izan ere, aurretik azaldu den moduan proiektu honetan garrantzi sistemen arteko interoperatibitateak izango du. Hala ere, klase eta egitura desberdinak erabiliko dira, bi hain zuzen ere. Bi klase nagusi eta bi liburutegi garatzea aurreikusten da. Honela, erabiliko den kanpoko tresna bakoitzeko (*Train Simulator* eta *RailDriver*) liburutegi bat sortuko da sinpleago eta errazago komunikatzeko ahalmena emanez. Ondoren, datu-basearen eta tresnen arteko komunikazioaz arduratuko diren bi klase edo programa nagusiak aurkitu ditzakegu (*RailController* eta *TrainSimATO*), eta ondoren xehetasun handiagoarekin azalduko den arren, balizak errepresentatzeko datu-egitura bat sortuko da, balizei dagokien informazio guztia gordetzeko.

6 Proiektuaren garapena

Behin proiektua planifikatu eta bideratu ondoren, garapenari ekin zaio. Garapena inkrementala izan da, aurretik azaldu den moduan, eta hiru fasetan banatu da. Hala ere, azalpena argiagoa izan dadin, egindako garapena sei atal desberdinetan azaltzea erabaki da; proiektuaren egitura, liburutegien garapena, datuen kudeaketa, komunikazioa, agertokiak eta *PER* ereduaren integrazioa.

6.1 Egitura eta klase-diagrama

Aipatu den moduan, proiektuak ez du egitura konplexua izango, eta bi programa garatuko dira, bi liburutegiz lagunduta. 6.1 irudiko klase diagraman argi azaltzen da nola garatu diren programak, liburutegiekin zein erlazio duten eta zein funtzionamendu gehitu zaizkion bakoitzari.

Klase-diagrama horretan ikus daitekeen moduan, bi liburutegi (*raildriver_lib* eta *trainsimulator_lib*) eta bi klase nagusi (*RailController* eta *TrainSimATO*) daude, eta baita balizen inguruko datuak gordetzeko erabiltzen den egitura bat ere. Bi liburutegiak tresnekin komunikazio zuzenago eta errazago bat lortzeko erabiltzen dira, eta aurretik eskaintzen dituzten funtzionalitateak zabaltzeko asmoz, hainbat funtzio gehigarri ere sortu dira. Bestetik, bi programa nagusiek liburutegi hauek erabiltzen dituzte, baina hauen funtzio nagusia tresnen arteko datuak era eraginkorrean trukatzea da, eta exekuzioa ahal den eraginkorrena izatea.

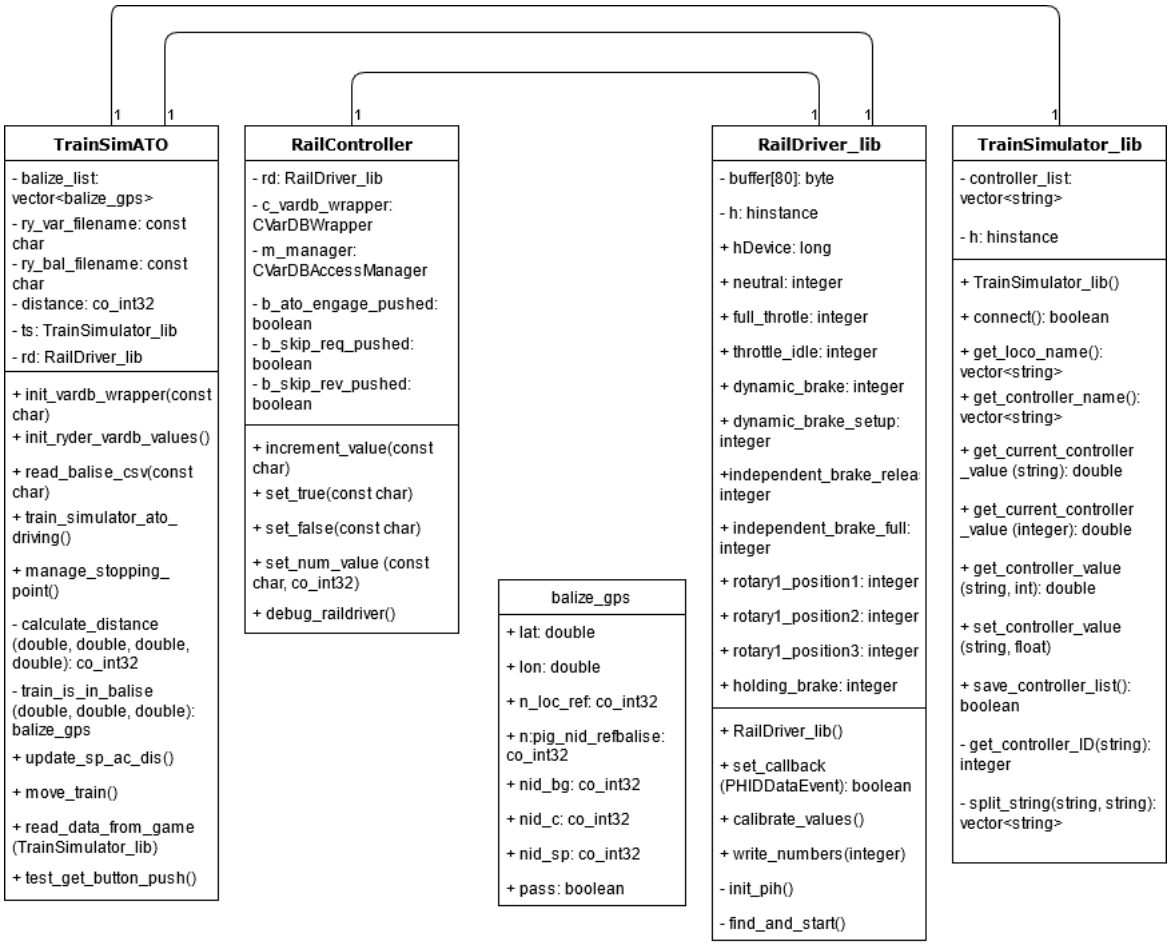
Programei dagokienez, bakoitzak sarrerako parametro batzuk behar ditu ongi funtzionatzeko, eta kasu honetan, parametro horiek konfigurazio fitxategietarako *path*ak dira, horrela, modu erraz batean konfigura daitezke hainbat gauza.

- **RailController**: sarrerako bi parametro behar ditu.
  - VarDB aldagaien hasierako *.ini* fitxategia. Honen bitartez, programari adieraziko diogu zein diren erabiliko diren aldagaiak eta zein den bakoitzaren *alias*a. Horrela, aldagaiaren kokalekua *VarDB* batetik bestera aldatzen bada, fitxategi hau aldatu besterik ez da egin behar.

  - RailDriver kontrolatzailearen kalibrazio fitxategia. Fitxategi hau lortzeko *RDCalibration* izeneko programa bat dago eskuragarri, eta palanka bakoitza kalibratzeko aukera emango digu.

- **TrainSimATO**: sarrerako hiru parametro behar ditu.

  - VarDB aldagaien hasierako *.ini* fitxategia. Aurkakorik ezean, bi programek erabiliko duten fitxategia bera izango da.
  - Balizen informazioa adierazteko *.csv* fitxategia. Fitxategi honen bitartez aukeratutako agertokian dauden balizen informazioa adierazten zaio programari, batez ere bakoitzaren GPS koordenatuak. Informazio honekin, programak zikloro begiratuko du ea trena balizaren baten gainean dagoen eta hala bada *ATO*ari abisatuko dio.
  - *RailDriver.dll* fitxategiaren *path*a. *Train Simulator* bideo-jokoarekin komunikatzeko beharrezkoa da *dll* hau, eta bideo-jokoaren instalazio karpetan dago kokatuta. Hala ere, instalazio karpeta hori aldagarria



6.1 Irudia: Garapenaren klase-diagrama

denez, fitxategi honen bitartez adieraziko da non dagoen liburutegia kokatuta.

6.2 Liburutegien garapena

Aurretik aipatu da *dll* liburutegiak erabilgarri daudela erabili behar ditugun tresnekin komunikatzeko[7]. Hala ere, liburutegi hauen zuzeneko erabilera nahiko nahasia da, eta sinpleagoa izan dadin, hauen gainean beste bi liburutegi sortu dira.

6.2.1 raildriver_lib

Liburutegi honen bitartez, *RailDriver* kontrolatzailera konektatu, hau kalibratu eta botoi eta palankak programatu daitezke. Horretarako, funtzio bat sortu behar da palanka eta botoi bakoitzak burutuko duen ekintzarekin, eta liburutegiari funtzio hau esleituko zaio *callback* moduan; horrela, palanka edo botoiren batean aldaketa dagoen bakoitzean funtzio horri deituko zaio. Gainera, funtzionalitate batzuk ere gehitu dira, tartean, kontrolatzaileak duen LED pantailan zenbakiak idaztea edo konektatzen den *RailDriver*eko palankak kalibratzea, enpresa garatzaileak eskuragarri jartzen duen kalibrazio programa bat erabiliz.

Hortaz, honako funtzionalitate hauek ditu *raildriver_lib* liburutegiak:

- Klasearen edo liburutegiaren eraikitzailean palanken defektuzko balioak esleitzen dira, ondoren, kalibraziorako fitxategi bat baldin badago, aldatu eta erabiliko den kontrolatzailera moldatuko direnak. Gainera, *dll* fitxategia kargatzen da eta honek dituen funtzioak definitzen dira.

- `boolean set_callback(PHIDDataEvent)`: *callback* funtzioa definitzen da, eta deitu egiten zaio kontrolatzailearen *output*ean aldaketaren bat gertatzen den bakoitzean. *Callback* funtzio horren parametro modura karaktere-kate bat bidaltzen da, kontrolatzailearen elementu guztien balioekin.

- `write_number(int)`: *RailDriver*en LED pantailan zenbaki bat idazteko funtzioa. Hiru digitu idazteko aukera duenez, *(-1000, 1000)* balioak idazteko ahalmena izango du, zenbaki negatiboek amaieran "." sinboloa izango dutelarik, 6.2 irudian ikusi daitekeen moduan.

- `calibrate_values()`: kontrolatzailearen palanken balioak kalibratzeko funtzioa da hau. Horretarako, aurretik *RDcalibration* izeneko aplikazio bat erabilita, ordenagailura konektatuta dagoen kontrolatzailearen kalibrazio fitxategi bat sortu behar da, eta ondoren, aipatutako funtzioan fitxategi hori lerroz lerro irakurtzen da eta dagokion aldagai globalaren balioa eguneratzen da.

6.2.2 trainsimulator_lib

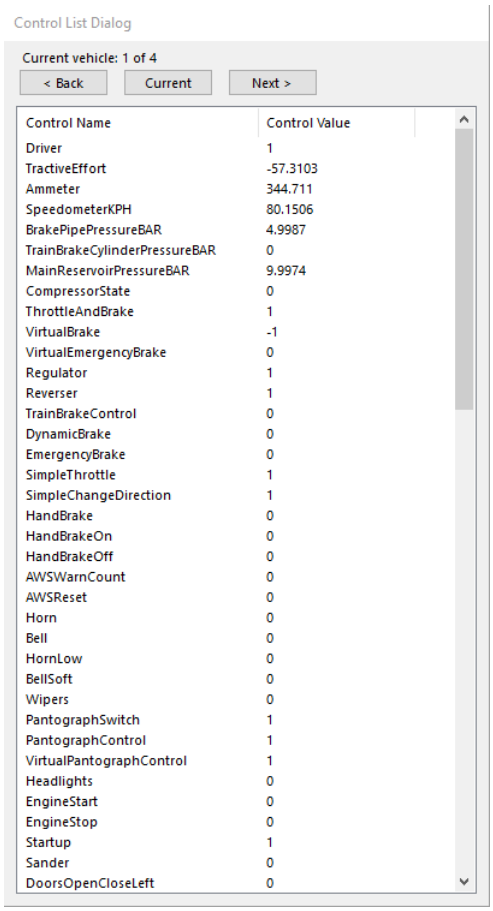
Bigarren liburutegi honekin *Train Simulator* jokora konektatu eta momentuan jokoan martxan dagoen treneko hainbat aldagai irakurri eta aldatzeko aukera izango da. Aipatu bezala, dagoeneko liburutegi bat eskaintzen da honetarako, baina kontrol sinple eta argiago bat burutzeko sortu da hau aurrekoaren gainean. Horretarako, honako funtzio hauek definitu dira:

- Klasearen edo liburutegiaren eraikitzailean jokoaren *path*a aurkitzen da eta bertako *dll* fitxategia inportatzen da; gainera, bertan dauden funtzioak ere definitzen dira.
- `boolean connect()`: funtzio honen bitartez jokoa martxan dagoela eta programaren eta jokoaren artean komunikazioa dagoela egiaztatzen da.
- `vector<string> split_string(string, string)`: karaktere-kate luze bat emanda, eta hau banatzeko karaktere batekin, *string* motako elementuez osatutako bektore bat itzultzen du, non elementu bakoitza sarrerako *string*aren zati bat izango den, mugatzaile baten arabera banatuta. Funtzio hau oso erabilia izango da, izan ere, jokotik itzultzen diren trenaren balioak (trenaren izena eta erabili daitezken aldagaien zerrenda) karaktere-kate luze baten moduan itzultzen ditu, eta beraz, banatu egin behar dira ondoren erabili daitezen.
- `vector<string> get_loco_name()` eta `vector<string> get_controller_list()`: bi funtzio hauen bitartez trenaren izena eta tren horrek erabiltzen dituen aldagaien zerrenda itzultzen du, hurrenez hurren. Aipatu behar da aldagaien izenak trenaren arabera aldatzen direla, tren bakoitzak dituen funtzionalitateengatik. Hala ere, garrantzitsuenak diren oinarrizko aldagaien izenak berdinak dira, *SpeedometerKPH*, *Reverser* eta *ThrottleAndBrake*, esaterako.
- `boolean save_controller_list()`: funtzio honen bitartez trenaren aldagaien zerrenda erabilgarri dagoen ala ez egiaztatzen da, eta hala bada, aldagai global batean gordetzen da, uneoro `vector<string> get_controller_list()` funtzioari deitzea aurrezteko.

6.2 Irudia: 344 eta −818 balioak LED pantailan idatzita, hurrenez hurren

- `double get_current_controller_value(integer)`: izenak aurreratu dezaken moduan, funtzio honen bitartez trenaren aldagai baten balioa kontsulta dezakegu, aldagai horri aurreko funtzioko zerrendan dagokion posizioarekin. Aldagaiaren zerrendako posizio zenbakia tren bakoitzean desberdina da, eta horregatik izena emanda bere indizea itzultzen duen funtzioa inplementatu da, `int get_controller_ID(string)`, eta hortaz, `double get_current_controller_value(string)` funtzioa sortu da, aipatutako biak uztartzen dituena. Aldagai hauen balioak zenbakizkoak dira, eta horietako asko *boolear*rak. Adibide moduan, 6.3 irudian ikusi daiteke exekuzio batean hainbat aldagaik duten balioa.

- `set_controller_value(string, float)`: funtzio honekin aipatutako zerrendako aldagaien balioa alda dezakegu, jokoaren portaera aldatuz. Horretarako, aldagaiaren izena eta balioa behar dira, eta aurrekoan bezala `int get_controller_ID(string)` funtzioa erabiliko da aldagaiaren indizea lortzeko.



6.3 Irudia: bideo-jokoko aldagaien balioen adibidea

6.3 Datuen kudeaketa

Proiektu honen muina sistema guztien arteko datuen interoperatibitatea da, eta hortaz, oso garrantzitsua da kudeaketa egokia egitea. Atal honetan, datu horiek nola kudeatu diren azalduko da, eta horretaz gain, azalpen zabal bat emango da hiru tresna nagusiek (*ATO*, *Train Simulator* bideo-jokoa eta *RailDriver* kontrolatzailea) erabiltzen dituzten datu, aldagai eta informazioaren inguruan.

Hasteko, datu-base partekatu bat erabiliko dugu, *VarDB* hain zuzen. *VarDB* enpresan garatutako aldagaien datu-base sistema bat da, eta *ATO*ak eta honekin batera funtzionatzen duten sistemek behar dituzten aldagaien balioak gordetzeko eta partekatzeko balio du. Liburutegi sinple baten bidez datu-base berriak sor daitezke, edo existitzen direnak kontsultatu eta aldagaien balioak aldatu. Liburutegi hori *VarDBWrapper* da, eta bere helburua *VarDB* baten edo gehiagoren aldagaietarako sarbideak definitzeko modu erraz bat eskaintzea da, konfigurazio-fitxategi baten bidez. Gero, aplikazioak aldagaiaren izen zuzena erabili besterik ez du egin behar balioetara iristeko. Aldagaiaren kokalekua *VarDB* batetik bestera aldatzen bada, konfigurazio-fitxategia editatu besterik ez da egin behar, aplikazioan aldaketarik egin gabe.

*ATO*aren kasuan, azaldu behar da hiru *VarDB* erabiltzen direla, bat simulatzaile bakoitzeko. Simulatzaile horiek estandarretan oinarritutako *subset*en araberakoak dira, 126, 130 eta 139 *subset*ak zehatzagoak izateko. Beraz, labur esanda, estandarretan oinarritutako hiru simulatzaile daude, bakoitza helburu desberdin batekin, eta hauek sortzen dituzten *VarDB*ak erabiltzen dira *ATO*arekin kontrolatzeko eta komunikatzeko. Horregatik, aldagai hauen balioak irakurri eta aldatzea izango da gure proiektuaren eginbehar garrantzitsuenetako bat.

Bestetik, aldagai guztiak kontsultatzeko *VarDBMonitor* izeneko aplikazioa erabiltzen da, non interfaze grafiko sinple baten bitartez, nahi den *VarDB*ko aldagaien balioa ikusi daiteken, eta hala nahi bada, balio hori indarrez aldatu. Aplikazio hau oso baliagarria izan da beharrezkoak izan diren aldagaiak zein diren eta zer balio duten identifikatzeko. Aipatutako aldagai hauek mota desberdinetakoak izan daitezke, eta nahiz eta zenbakizkoak diren denak, aldagaiaren kontrolaren arabera, hiru mota bereizi daitezke gutxienez. Batetik, *ATO*ak kontrolatzen dituen aldagaiak daude, eta bertan, *ATO* sistemak berak eragiten dituen aldaketak islatzeko aldagaiak daude, trenaren balazta edo trakzioa aktibatzea, esaterako. Bestalde, tren gidariak treneko kabinatik egiten dituen aginduak adierazteko aldagaiak daude, *ccu* deituak, eta proiektu honen kasuan aldagai horiek *RailDriver* kontrolatzailearen bitartez aldatzen dira. Amaitzeko, berriz, *ERTMS* segurtasun sisteman beharrezkoak diren aldagaiak aurki daitezke, baina kasu honetan ez dira erabili, ez baita beharrik izan.

6.3.1 Trena gidatzeko beharrezko aginduak

Tren bat gidatzeko erabiltzen diren aginduak orokorrean hiru dira, 6.4 irudian ongi adierazten den moduan:

1. **Reverser**. Palanka honek hiru posizio ditu (*forward*, *coasting* eta *reverse*) eta trenaren norabidea definitzeko erabiltzen da. Trenek bi kabina izan ohi dituztenez, kabina bakoitzeko *reverser*ak kabina horretatik begiratuta trenak izango duen norabidea adieraziko da, aurrera joan nahi bada *forward* posizioan jarrita eta atzera joan nahi bada *reverse* posizioan jarrita. *Coasting* posizioa trena geldik dagoenean erabiltzen da, mugitu ez dadin.

2. **TBL**. Palanka honen bitartez trena azeleratu eta balaztatu daiteke, eta palankaren posizioaren arabera azelerazio eta balazta efektua handiagoa edo txikiagoa izango da. Kasu honetan balazta dinamikoa izango da, eta motorraren trakzio-indarra erabiltzen da trenaren abiadura murrizteko, normalean, jaitsiera batean.

3. **Holding brake**. Trena balaztatzeko beste palanka bat, aurrekoaren aldean honen bitartez balaztatze ahaltsuagoa lortzen da, treneko bagoi guztien balaztak aplikatzen baitira, eta beraz, trena gelditzeko erabiltzen da.

6.3.2 RailDiver kontrolatzailearen kontrolak

Honetaz gain, *RailDriver* kontrolatzailearen bitartez trena eskuz kontrola daiteke edo martxan dagoen *ATO*ari kanpotik aginduak bidaltzeko aukera izango da (larrialdietako balazta aktibatzea, esaterako). Aldatzen diren aldagai hauek bi mota desberdinetakoak dira: *boolear* edo zenbakizkoak izan daitezke, eta zenbakizkoak badira, inkrementalak edo balio zehatz batekoak. Hortaz, mota bakoitzeko funtzio bat sortu da, esleipena zuzenagoa izan dadin. Beraz, ondorengo zerrendan agertzen dira erabilgarri egongo diren kontrolak *RailDriver* kontrolatzailea erabiliz gero, eta 6.5 irudiko eskeman ikusi daiteke kontrol horiei zein palanka eta botoi fisikori dagozkien.

1. **Reverser**, **TBL** eta **holding brake** palankak. Palanka hauekin aurretik azaldu diren hiru oinarrizko funtzionamenduak kontrolatuko dira, hau da, trena gidatzeko balioko dute.

2. **Skip stop** eta **skip revoke**. Bi hauek botoi inkrementalak dira, eta *ATO*ak daukan funtzio garrantzitsu bat aktibatzeko balio dute. Printzipioz, *ATO*ak trena adierazitako geltoki guztietan gelditzuko du, baina hala nahi bada, posible

6.4 Irudia: *Train Simulator* bideo-jokoko trenaren kontrol interfazea

da hurrengo geltokian ez geratzea, eta horretarako erabiltzen da *skip stop* botoia. Bestetik, behin *skip stop* aktibatuta atzera bota nahi bada, *skip revoke* botoia erabili daiteke.

3. **Engage** botoia. Botoi inkremental honen bitartez *ATO*a gidatzen hasi dadin adierazi behar zaion azken agindua aktibatzen da. *GoA 2* maila inplementatzen denez (ikus 1.1 taula), beharrezkoa da eskuz ematea botoi honi *ATO*a gidatzen hasi aurretik. Gainera, behin trena martxan dagoela, posible da *ATO*a desaktibatzea, horretarako **disengage** botoia erabiliz.

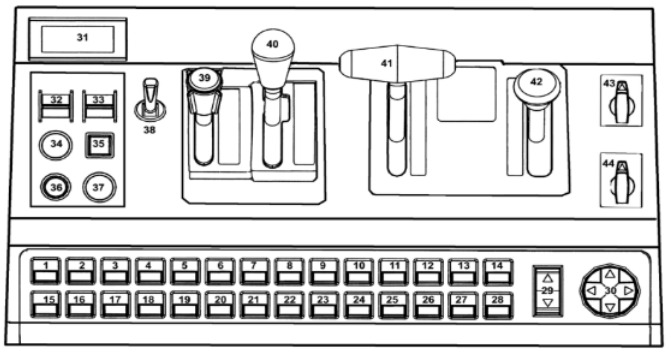
4. **Cab selection**. *Switch* baten bitartez trenaren kabina bat edo beste aukera daiteke (A edo B), alde batera edo bestera gidatzeko aukera emanez. Aldagai hau *boolear* bat da, eta *A* aldeko kabina aktibo badago, *B* aldekoak desaktibatuta egon behar du, eta alderantziz.

5. **Emergency brake** botoia. Botoi honen bitartez, izenak dioen moduan larrialdietako balazta aktibatu/desaktibatuko da, eta beraz, treneko balazta guztiak aktibatuko dira trena ahalik eta azkarren gelditzeko.

6.3.3 Train Simulator bideo-jokotik eskuratutako informazioa

Bestalde, *ATO*ari dagokionez, trenaren informazioa behar du uneoro, horren arabera erabakitzen baitu zein izango den trenak bete beharko duen hurrengo agindua. Zehatzagoak izateko, honako informazio hau behar du *ATO*ak ongi gidatu dezan:

1. **Trenaren uneko abiadura eta azelerazioa**. Trenak une horretan daraman abiadura eta azelerazioa jakin behar dira, horrela jakingo baitu gehiago azeleratu behar duen ala ez, edo geltoki batera iritsi den eta trena geldituko den.



33. Emergency Brake → **emergency_brake** (boolean)
34. Alerter →**engage_button_counter** (Incremental)
35. Sander -> **disengage_button** (Boolean)
36. Pantograph →**skip_stop_req** (Incremental)
37. Bell → **skip_stop_revoke** (Incremental)
→ **travel_direction_fwd/bwd** (Boolean)
40. Combined throttle and dynamic brake (with "gate" center)
→ **tbl_fwd/coasting/bwd** (Boolean)
→ **plc_trc_brk_dem** (-1000, 1000)
41. Automatic brake, (detent before e-brake) -> **holding_brake_applied** (Boolean)
43. Rotary wiper switch → **cabA_active / cabB_active** (Boolean)

6.5 Irudia: *RailDriver* kontrolatzaileko botoi eta palanka fisikoei dagozkien kontrol aginduak

2. **Trenak burututako distantzia**. Aipatu izan da *ATO*ak trena *itsu* gidatzen duela, hau da, aurrez definitutako profilen arabera gidatzen duela, eta benetan ez dakiela bere inguruan zer gertatzen ari den. Horregatik, beharrezkoa da bere posizioa kontrolatzea, egindako distantziaren kontrol bat eramanez.

3. **Pasatako balizen informazioa eta posizioa**. Trenak burutu duen distantziaz gain, komeni da balizak erabiltzea trena lokalizatzeko. Balizen bitartez, hauen ondotik pasatzean, trenari bere uneko posizioa eguneratzen zaio, eta hortaz, aurretik egin daitezkeen distantziaren kalkulu arazoak ekidin daitezke. Hala ere, baliza asko erabiltzea ez da oso errentagarria, garestiak baitira hauen muntaketa eta mantenua.

Ondoren aipatuko den moduan, trena kontrolatzeko beharrezko informazioa zuzenean lortuko da aipatu den datu-basetik. Hala ere, jokoak eskaintzen digun informazio mugatua dela eta, trenaren egoera datu batzuk lortzeko beste bide batzuk erabili behar dira. *Train Simulator* bideo-jokoaren *API*a eta sortu den liburutegia erabilita, trenaren uneko abiadura jakin daiteke, eta baita trenaren aurreko aldearen GPS koordenatuak zein diren. Horrela, azelerazioa eta igarotako distantzia lortzeko, kalkuluak egin behar dira aipatutako bi datuetatik abiatuta. Kalkulu horiek denboraren menpe daude, hau da, denbora tarte batean bi balio hartzen dira eta hauen bidez lortzen da azelerazioa eta egindako distantzia. Beraz, garrantzitsua izango da exekuzio abiadura konstantea izatea edo behintzat beste edozerk ez oztopatzea, horregatik, kalkulu hauek burutzeko, *multithreading*a aplikatzea eta *thread* bat sortzea erabaki da. Zehazki, honako hauek izan dira *thread* horretan burutzen diren kalkuluak:

- Azelerazioa kalkulatzeko, 20 milisegundoro trenaren abiadura irakurtzen da eta bi abiaduren arteko zatiketa bidez azelerazioa adierazteko aldagai global bat eguneratzen da.

- Egindako distantzia kalkulatzeko, berriz, *haversinu*aren formula erabiltzen da [8]. Formula honek, bi punturen arteko distantzia ortodromikoa kalkulatzen du, hau da, lurrazalaren gaineko distantziarik laburrena, mendi eta muinoak kontuan hartu gabe, noski. Aurrekoaren moduan, 20 milisegundoro hartzen dira GPS laginak, eta begizta infinitu batekin exekuzio guztian zehar distantziak etengabe gehitzen dira aldagai global batean. Gure kasuan, gehitzen joango diren distantziak oso laburrak dira, normalean metro bat baino txikiagoak, eta hori dela eta ez du axola distantzia ortodromikoa den ala ez.

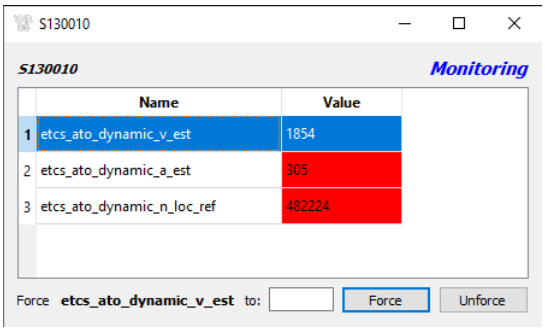
Kudeatu behar den informazioa eta datuak zein diren zehazteko, lagungarria izan da exekuzioak egitea eta *ATO On Board*eko sistemaren interfaze grafikoa (*DMI* deritzona) eta *VarDBManager* behatuz. Hortaz, 6.6 eta 6.7 irudietan agertzen den moduan, ikusi daiteke *VarDB*ko zein aldagaik egiten duen zer, haren efektuak *DMI*an ikustarazten baitira. 6.6 irudian, abiadura, azelerazioa eta egindako distantzia adierazten duten aldagaiak ikusi daitezke, hurrenez hurren; eta hurrengoan (6.7

irudian) hauek interfazean nola errepresentatzen diren. Gainera, kontuan hartu behar dira unitateak zein diren, izan ere, *subset 130* estandarrari kasu egin ezkero, abiadura m/s-tan adierazi behar da, azelerazioa mm/s^2-tan eta distantzia cm-tan.

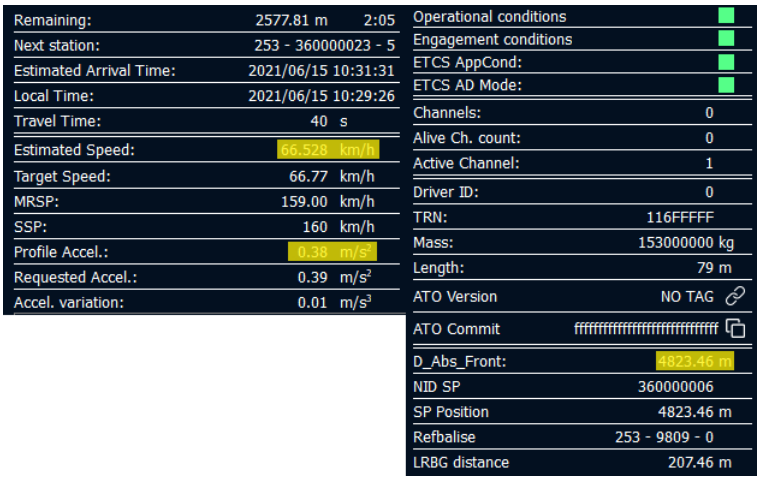
6.4 Komunikazioa

Exekuzio egoki baterako, beharrezkoa da komunikazioa eraginkorra eta azkarra izatea, eta datuak modu ordenatu eta optimizatu batean elkartrukatzea. Horretarako, komunikazioa nola egin behar den definitu behar da, eta beharrezkoak izango diren aldagaiak identifikatu eta bakoitza noiz erabili beharko den erabaki. Proiektuaren zati bakoitzak 5 Domeinuaren deskribapena, betekizunak eta soluzioaren diseinua atalean definitutako egitura jarraituko du, eta beraz, sortu diren programek elementu desberdinen arteko zubia osatzea izango dute helburu.

Hasteko, aipatu da *thread* desberdinak sortu direla exekuzioa orekatuagoa izan dadin eta kalkuluak egoki egin daitezen. Beraz, bi *thread* sortu dira, bakoitzak bere funtzioa izango duelarik eta hortaz, helburu berariazko batekin. Exekuzioaren hasieran, oso garrantzitsua izango da bai jokoa eta baita *ATO*a prest egon arte



6.6 Irudia: *VarDB* datu-baseko aldagaien adibidea



6.7 Irudia: *ATO*ren interfazean aurreko aldagaien errepresentazioa

zain egotea. Biak ala biak software produktu handiak eta konplexuak dira, eta hortaz martxan jartzeko nahikoa denbora behar izaten dute, eta dena prest egon arte garatutako programek zain egon behar dute. *ATO*aren zain egoteko, honen *launcher*ean konfigura daiteke programa noiz exekutatuko den, eta jokoaren zain egoteko, berriz, garatutako liburutegia erabiliz, bideo-jokoko trenaren informazioa lortu arte zain egon gaitezke.

Beraz, behin exekuzioa hasita eta dena prest dagoela, honako bi *thread* hauek sortzen dira eta trenaren gidatzea hasten da:

- `move_train()`: bideo-jokoko trena mugitzeko funtzioa. Funtzio honetan, begizta infinitu baten bitartez *VarDB*tik aipatutako *reverser*, *TBL* eta *holding brake* aldagaien balioak irakurtzen dira eta bakoitzaren arabera bideo-jokoari *trainsimulator_lib* liburutegiaren bidez jokoari komunikatzen zaio. Horretaz gain, aldagai globaletatik abiadura, azelerazioa eta egindako distantzia bidaltzen zaizkio bueltan *ATO*ari. Amaitzeko, trenaren GPS koordenatuen arabera, baliza baten gainean dagoen ala ez egiaztatzen da, horretarako egindako funtzio baten bidez, eta hala baldin bada, *VarDB* bidez baliza pasa dela adierazten zaio *ATO*ari, eta egindako distantzia eguneratzen da.

- `update_sp_ac_dis()`: funtzio honen bitartez abiadura, azelerazioa eta distantzia eguneratzen dira uneoro. Begizta infinitu batekin, 20 milisegundoro kalkulatzen da azelerazioa eta egindako distantzia, GPS koordenatuen bitartez. Ondoren, aldagai globalak eguneratzen dira balio hauekin eta horrela beste hariak inongo arazorik gabe irakurri ditzake.

Bestalde, *RailDriver* kontrolatzailearekin ere komunikazioa garrantzitsua da. Kasu honetan, *RailController* programa arduratzen da kontrolatzailea eta *VarDB*aren arteko loturaz, eta horretarako *callback* funtzio bat sortu da. Funtzio honek sarrerako *string* parametro bat dauka, botoi eta palanka bakoitzaren balioa adierazten duen karaktere-kate batez osatuta, eta beraz, balio bakoitzeko *VarDB*ko dagokion aldagaiaren balioa aldatzen da, ondoren horrek simulatzailean (*Ryder* edo *Train Simulator*, egoeraren arabera) eragina izan dezan.

Komunikazioaren atala amaitze aldera, esan beharra dago ezin dela dena kontrolatu, bideo-jokoak eskaintzen duen *API*a nahiko mugatua baita. Horregatik, gauza gehigarriak egin nahi badira beste modu batzuk topatu behar dira, eta horren adibide dira ateen kontrola eta kabina aldaketa. Simulazioa errealistagoa izan dadin, eta *Train Simulator* bideo-jokoak ahalbidetzen duenez, ateak ireki eta ixteko aukera gehitu nahi izan da trena geltokietara iristen denean (6.8 irudian bezala). Baina hori ezin da *API*aren bitartez egin, eta topatu den modu bakarra teklatuko *t* tekla sakatzea izan da. Gauza bera gertatzen da kabina aldaketa egin nahi bada. Gauza zehatz batzuk egiteko komenigarria da jokoko kabina aldatzea, bai atzeraka joan nahi delako edo beste aldeko zerbait ikusi nahi delako, baina hemen ere teklatua erabili gabe ez da posible hau egitea. Horregatik, funtzio bat sortu da teklatuko

teklak simulatzen dituena, kontuan hartuz bi motatako teklak daudela, *birtualak* eta *hardware* teklak, eta jokoak *hardware* teklak bakarrik irakurtzen dituela.

6.5 Agertokiak

Agertoki edo eszenarioak trenak bete behar duen ibilbide bat deskribatzen du, nondik nora joan behar duen, zein abiadura maximotan, nolako gradientea duen, egin behar dituen geltokiak eta abar. Trena automatikoki gidatzen denez, beharrezkoa da informazio guzti hau helaraztea, eta horretarako erabiltzen dira *QML* fitxategiak.

Proiektu honetarako, bi agertoki sortu dira, biak bideo-jokoak eskaintzen dituen bi ibilbide desberdinetan, Alemanian, eta tren berdina erabiltze aldera, *DB Regio BR 442 'Talent 2'* tren modeloa erabili da bi agertokietan [9]. Lehen agertokia Nurenberg-Regensburg linean kokatutako zati bat da, Undorf eta Regensburg artean, zehatzagoak izateko. Bi geltoki igarotzen dituen ibilbide bat da, eta abiadura handiak hartzen ditu trenak (ia ibilbide osoan 110km/h eta ibilbidearen amaieran 120km/h). Hortaz, ibilbidea nahiko sinple eta azkarra da, normalean 6 minutu behar izaten dira amaitzeko, tarteko geldiadia kontuan hartuta. 6.9 irudian ikusi daiteke zein den benetan ibilbide hau, satelitetik ateratako irudi baten bidez.

Bestalde, bigarren ibilbide bat dago, Stralsund-Sassnitz linean kokatuta, eta zehazki, Lancken eta Lietzow artean, 6.10 satelite irudian ikusi daitekena. Ibilbide hau aurrekoaren aldean konplexuagoa da, abiadura maximoak txikiagoak izateaz gain (40km/h eta 70km/h tartean), abiadura aldaketa asko daude, luzeagoa da eta tartean trenbide-pasagune batzuk ditu. Gainera, ibilbidearen zati handi batean beheranzko aldapa handia dago eta balaztatzen ez bada abiadura asko hartzen da.

Beraz, azaldutako bi agertoki hauek *QML* fitxategietan adierazi behar dira,

6.8 Irudia: *Train Simulator* bideo-jokoko trena geltokian ateak irekita

*ATO*ak ondo gidatu dezan. *QML* javascript-en oinarritutako lengoaia bat da, eta kasu honetan *ATO*ak burutu beharko duen prozesua adierazteko erabiltzen da. Fitxategi hauetan, *125. subset*a jarraituz, agertokiaren exekuzio osoa deskribatu behar da, *ATO*a konfiguratzetik trena azken geltokira iristen den arte, eta nahiko prozesu nahasgarria izan daiteke. Hala ere, orokorrean, bi pakete mota sortu behar dira, *Journey Profile* eta *Segment Profile*, eta *ATO*ak hauek jaso eta irakurri ahala trena gidatzen du.

- **Journey Profile** (JP) batek trenaren ibilbidea definitzen du, trenak zeharkatuko dituen hainbat *Segment Profile* zerrendatuz. Gainera, *Segment Profile* horien inguruko informazio zati bat ere igorri behar du, tartean profil bakoi-

6.9 Irudia: Undorf-Regensburg ibilbideko satelite argazkia.

6.10 Irudia: Lancken-Lietzow ibilbideko satelite argazkia.

tzak izango dituen geldialdi puntuak (*Timing Point* edo TP) adieraziz, proiektu honetan trenak gelditu beharreko geltokiak adieraziko ditu. Kontuan hartu behar da informazio asko igorri behar dela honelako profiletan, baina hemen garrantzitsuenak bakarrik jorratuko ditugu:

  - SP identifikatzailea.
  - SP bertsioa.
  - TP identifikatzailea.
  - TP horren iritsiera data eta ordua.
  - TP horren irteera data eta ordua.
  - TP hori ibilbideko azkena den ala ez.
  - TP horretan zain egon beharreko denbora minimoa.
  - Ateen kudeaketa TP horretan (eskuineko ateak, ezkerrekoak, biak ala aterik ez ireki).

- **Segment Profile** (SP) baten bitartez trenak igaroko duen segmentuaren inguruko informazio xehatuagoa bidaltzen da, ibilbide zati horren inguruko zehaztasunak deskribatuz. Ez dago inongo irizpiderik segmentuak banatzeko, eta hortaz, ingurunearen eta beharren arabera banatzen dira segmentuak ibilbide batean zehar. Kasu honetan ere igortzen den informazioa handia da, horregatik, garrantzitsuenak eta interesgarrienak aipatuko dira:

  - SP identifikatzailea.
  - SP bertsioa.
  - SP egoera.
  - SParen luzera cm-tan.
  - Lurrazalen altitudea.
  - Abiadura maximoen profila.
  - Gradienteen profila.
  - Balizen informazioa (identifikatzailea eta posizioa dagokion SParekiko).
  - TP identifikatzailea.
  - TP posizioa dagokion SParekiko.

Kontuan hartu behar da hainbat arau jarraitu behar dituztela profil hauek, eta ezin dutela inolako inkonsistentziarik izan. Honelakoren bat baldin badute, exekuzioa ez da hasiko eta konsistentzia errorea gertatuko da.

- Konsistentzia erroreak.

  - JPko TP ez dago dagokion SPan.

  - 'invalid' egoerako SP bat jasotzen da.
  - JPko SPetan adierazitako denborak ez daude gorako ordenan.
  - Posizio edo kokapen balio guztiak ez daude gorako ordenan.
  - SPko edozein posizio SParen luzera baino handiagoa da.
  - Posizio daturen bat zero da.
  - SP eta JPko bertsioak ez datoz bat.

Beraz, aipatutako *QML* fitxategiak sortzeko datu guzti hauek behar ditugu, eta datu horiek *Train Simulator*etik lortu behar ditugu, azken finean, bideo-jokoko agertokietan oinarritu behar dugulako. Gainera, *QML* hauen sorrera eskuz egin behar da, ez dago oraingoz datu multzo batetik *QML*ak automatikoki sortzen dituen tresnarik. Lehen zatiari dagokionez, *TrainSimATO* programa moldatu da bideo-jokotik datuak ateratzeko, eta *.csv* bat sortzen du datuekin.

Datuak ateratzeko, garatutako *TrainSimATO* aplikazioari gehigarri bat gehitu zaio, exekutatzeko *ATO*ko *launcher*raren beharrik izan ez dezan. Horrela, funtzionalitate berri honen bitartez, bideo-jokoarekin konektatuko da eta trena prest dagoenean bertako datuak irakurtzen eta gordetzen hasiko da. Horretarako, beharrezkoa izango da eskuz exekuzio bat egitea, trena martxan dagoela hartu behar baitira datuak. Gordetzeko, *.csv* bat sortzen du, eta lerro berri bat gehituko du egindako distantzia aldatzen den heinean, honako balio hauekin batera: latitudea, longitudea, gradientea, abiadura eta abiadura aldaketa.

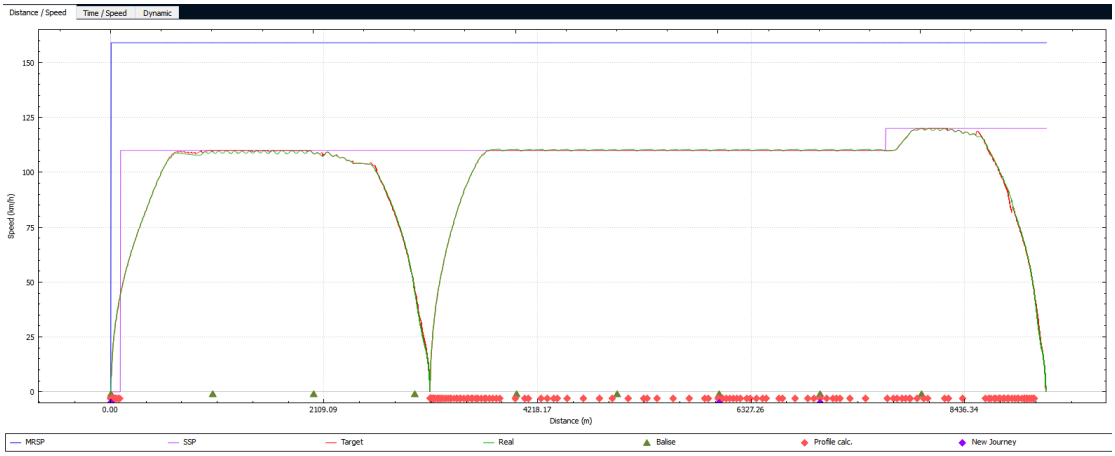
- **Latitudea eta longitudea**: trenaren posizioaren GPS koordenatuak, zuzenean jokotik hartuta.
- **Egindako distantzia**: trenak egin duen distantzia gordetzeko, GPS koordenatuen arteko distantzia kalkulatu eta gehitzen da. Koordenatuek zehaztasun handia dutenez, $50cm$ inguruko tarteak lortzen dira, eta aipatu den moduan, distantzia aldaketa bakoitzeko lerro berri bat gehitzen da.
- **Gradientea**: trenak momentu horretan duen gradientea aprobetxatuz trenbidearen gradientea lortu daiteke, eta zuzenean gehitu *.csv*ra.
- **Abiadura**: abiadura ere zuzenean lortu daiteke jokotik, km/h-tan
- **Abiadura aldaketa**: ez dago modurik *API*aren bidez baimendutako abiadura maximoa zein den lortzeko, eta hortaz, *RailDriver* kontrolatzailea erabilita, aldaketaren bat dagoen bakoitzero botoi bat sakatu daiteke eta horrela *.csv*an adieraziko da abiadura maximoa aldatu dela. Ondoren, eskuz adierazi beharko da zati bakoitzak zein abiadura maximo duen.

Beraz, datu hauekin sor daiteke *QML* bat, baina oraindik balizak faltako lirateke. Balizak gehitzea hautazkoa da, azken finean trena birkokatzeko balio baitute, eta

distantziaren kalkulua egokia baldin bada ez du eragin handirik izaten. Gainera, ezin dugu jokotik atera bertan dauden balizen posizioa, eta horregatik erabaki da irudizko balizak sortzea, hau da, eskuz baliza batzuk gehitzea bidean, sortutako *.csv*a oinarritzat hartuz. Honela, *ATO*aren exekuzioan distantziaren kalkuluan erroreren bat baldin badago, baliza hauetakoren baten koordenatuetatik pasatzean, distantzia eguneratuko da *.csv*aren balioetara, eta amaierako errorea murriztuko da. Hemen ere ez dago inongo irizpiderik balizak kokatzeko, baina erabaki da kilometroro bat kokatzea gutxi gorabehera.

Behin balizak sortu eta *QML*an gehitu direnean, *TrainSimATO* programa arduratuko da jokotik koordenatuak irakurtzeaz eta balizaren bat igarotzen badu *ATO*ari abisatzeaz eta distantzia eguneratzeaz. Baliza baten gainetik pasa den ala ez egiaztatzeko zikloro koordenatuak konparatzen dira, baina beharrezkoa izan da tarte bat definitzea, balio oso zehatzak direnez posible baita koordenatuak zehazki bat ez egitea.

Amaitze aldera, sistema guztia martxan dagoenean, datuak denbora errealean ikusteko *DMI*a erabiltzen da, eta trenak izan duen abiadura eta distantziaren profil edo grafikoa ikusi daiteke. Datu hauek oso baliagarriak izan daitezke sortu diren *QML*ak ondo egin diren ala ez zehazteko, eta adibide moduan, 6.11 irudian ikusi daiteke *Undorf-Regensburg* ibilbidea burutu ondoren gelditu den grafikoa. Grafiko honetan abiaduraren eta distantziaren arteko irudikapen bat egiten da, ardatz horizontalean distantzia eta bertikalean abiadura adieraziz, eta hortaz, azkar antzematen da hiru geltoki daudela (hasierakoa, erdikoa eta azkena), edo behintzat trena hiru aldiz gelditu dela. Horretaz gain, ia ibilbide guztian bai abiadura maximoa (lerro morea) eta baita trenak izan duen abiadura (lerro berdea) 110km/h-koa izan dela ikusi daiteke, azken zatian ezik, bertan 120km/h-koa izan baita. Gainera, beheko aldean hiruki berde batekin trenak igaro dituen baliza guztiak irudikatzen dira, eta baita gradientearen eraginez egin behar izan diren profilaren kalkulu guztiak ere, erronbo gorrien bidez.



6.11 Irudia: *DMI*ak eskaintzen duen abiadura eta distantziaren arteko grafikoa, Undorf-Regensburg ibilbidea amaitu ondoren.

6.6 PER ereduaren integrazioa

Garapenarekin amaitzeko, orain arte egindako lana *PER* ereduarekin integratu nahi da. Integrazio honetan, helburu nagusia *Train Simulator* bideo-jokoa entrenamendurako irudiak sortzeko tresna izatea da, eta horrela, benetako irudiak lortzeko grabazio kostuak aurreztea.

Lanaren zati hau *PER* taldekoekin burutu da, eta talde honek martxan duen proiektu batean laguntzeko asmoz garatu da. Proiektu hau Herbehereetan kokatuta dagoenez, lehen pausua ibilbide horietako agertokiak lortzea da *Train Simulator* bideo-jokorako, eta beraz, agertoki hauetako bat erosi behar da. Ondoren azalduko den moduan, erosketa hau burutzeko denbora behar izan da, izan ere, enpresako erosketa-prozesua konplexua da eta *Steam*[10] moduko bideo-jokoen plataformetatik enpresa baten izenean erostea korapilatsua da.

Behin agertokiak erosi eta instalatu ondoren, bideo-jokoak zein funtzionalitate dituen aztertu behar da, hau da, ereduarentzat sortuko diren irudiak zenbateraino pertsonaliza daitezkeen. Funtzionalitate hauek *PER* taldekoek nahi dutenaren araberakoak izan dira, ondorengo hauek, zehatzagoak izateko:

- **Semaforoen kontrola**. Semaforoak gorriak, horiak edo berdeak egon daitezke, eta ikusmen artifizialeko ereduak hiru koloreak desberdintzen ikasi behar du. *Train Simulator*ek agertokiak editatzeko aukera ematen du, baina semaforoak ezin dira horren erraz kontrolatu. Horregatik, semaforoen koloreak aldatzeko (gorritik berdera eta alderantziz, horiarekin ez), trenbide horretan beste tren bat kokatu behar da, eta bideo-jokoak aurrean tren bat ikusten duenez, semaforoa gorrian jartzen du.

- **Kameraren kontrola**. Bideo-joko honetan kamera nahi den moduan mugi daiteke eta hortaz, trena nahi den angelu eta posiziotik begiratu. Hala ere, bi posizio desberdinetatik lortu nahi izan dira irudiak (bi kamera izango bagenitu bezala), eta 3Dko ikuspegi bat sortu, distantzien kalkulua hobea izan dadin, baina *Train Simulator*ek ez du honelako ezer egiteko aukerarik ematen.

- **Bagoien loturak**. Herbeheretako proiektuaren zati bat *Shunting*a egitea da, hau da, bagoiak elkarren artean lotu eta askatzea eta *Train Simulator*ek ere horretarako aukera ematea komeni da. Kasu honetan, bideo-jokoak ematen du aukera hau egiteko.

- **Pertsona eta objektuak nonahi kokatzea**. Trenbideko seinaleez gain, bertan egon daitezken pertsonak edo objektuak identifikatzeko aukera izan nahi da, segurtasuna bermatzeko. Bideo-jokoak hori egiteko aukera ematen du, eta beraz, hainbat puntutan pertsona edo langileak gehituko dira trenbidean, *shunting*ean esaterako.

Zer egin daitekeen eta zer ez aztertu ondoren, aurreko fasean bezala, agertoki bat hartuz *QML* bat sortu da *ATO*ak automatikoki gidatu dezan. Pauso hau erraza izan da, ibilbide labur bat aukeratu baita, inongo zailtasunik gabea, eta aurretik egindako *QML*ak hartu dira eredu moduan.

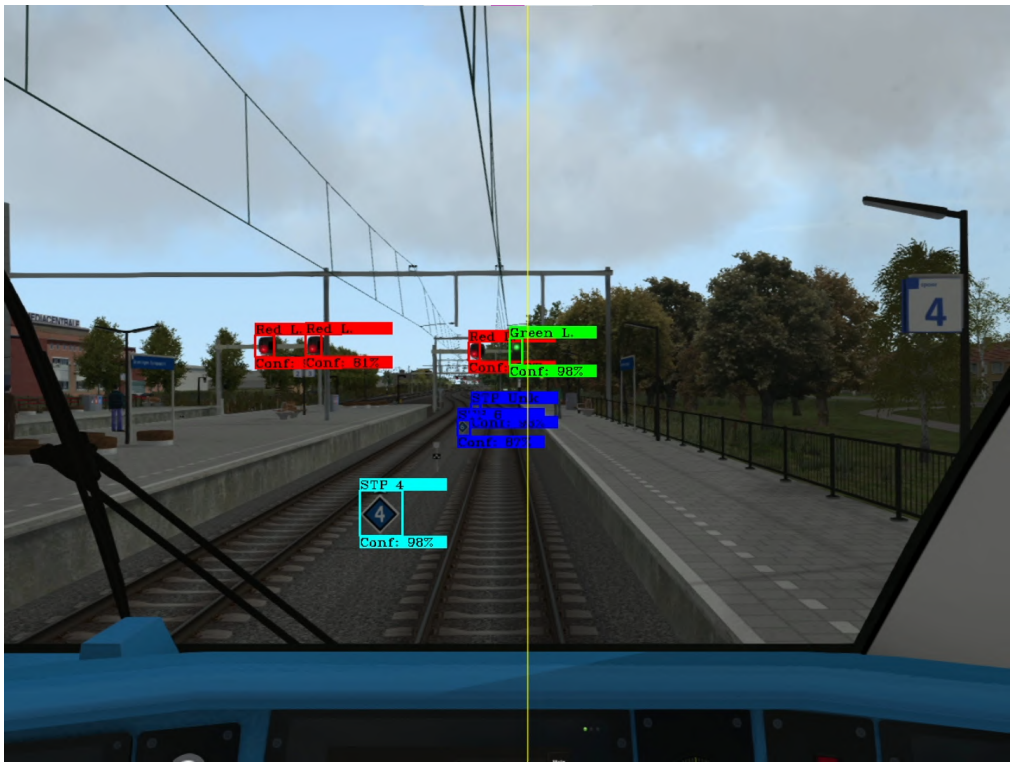
Ondoren, eredua martxan jarri aurretik bideo-jokotik irudi batzuk atera eta etiketatu dira (semaforoak eta seinaleak) eta ereduaren entrenamendu txiki bat egin da irudi hauek hartuta.

Puntu honetara iritsita, komeni da aipatzea fase honetan beharrezkoa izan dela *PER* taldekoen laguntza, beraiek sortutako eredu bat baita, eta haien inplementazioa beharrezkoa zen fase hau aurrera eramateko. Horregatik, lankide hauen erabilgarritasunarekiko oso menpekoa izan da garapena, eta hauek lehentasun handiagoko proiektuak izan dituztenez, hemen aritzeko denbora askorik ez dute izan, eta uste baino denbora gehiago luzatu da garapena alde honetatik.

Behin eredua seinaleak detektatzeko gai denean, hasierako asmoa *streaming* bat eginez bideo-jokoko irudiak denbora errealean zuzenean *PER* ereduari bidali, eta bueltan *ATO*ari oztopoez abisatzea izan da. Hala ere, probak egin ondoren, ikusi da ezin dela *streaming*a egin, zarata asko sortzen baita eta horrela ereduak ez baitu ezer detektatzen. Beraz, alternatiba moduan, eta denbora faltagatik, eredua aurrez grabatutako bideoen bitartez probatu da, eta ongi funtzionatu du. Puntu honetan, eredua gai da *Train Simulator*eko irudiak hartu eta seinale eta semaforoak detektatzeko, asmatze tasa bikainarekin. Adibide moduan ikus daitezke 6.12 eta 6.13 irudiak, non seinale eta semaforoak ondo hautematen dituen ikusten den, %90etik gorako konfiantza-tasarekin. Horrela, hemendik aurrera, bideo-jokotik mota desberdinetako ehundaka irudi desberdin lortu daitezke, eta ereduari entrenatzeko material modura erabili, hasierako helburuak betez.

Amaitzeko, larrialdietako balaztari dagokionez, aipatu behar da ezin izan dela inplementatu, denbora faltagatik ezin baita bideo-jokoaren *streaming*a egoki egitea lortu eta beraz, ezin izan da denbora errealean irudiak prozesatu, oztopoak detektatu eta *ATO*ari abisatu. Arazo hau konpontzea eta inplementazioa egitea etorkizunerako lan moduan gelditu litzateke.

6.12 Irudia: Eredutik igarotako *frame* bat. Semaforo gorrien eta abiadura panelen detekzioa.



6.13 Irudia: Eredutik igarotako *frame* bat. Semaforo eta geltokietako panelen detekzioa.

7 Proba kasuak

Behin garapenarekin amaituta, beharrezkoa izaten da proba kasuak burutzea, benetan egindako lanak espero diren emaitzak ematen dituen ala ez aztertzeko. Kasu honetan, probak egiteko *Testing Environment* deritzon ingurunea erabiliko da. Ingurune honen bitartez *ATO*ak funtzionatzeko beharrezkoak diren tresna eta programa guztiak martxan jartzen dira eta denbora errealean ikusi daiteke gertatzen den guztia (aldagaiak, *ATO*aren egoera, ...). Beraz, garatu diren bi programak ingurune honetan exekutatzea inplementatu da eta hainbat proba egin dira denak ondo funtzionatzen duen ikusteko.

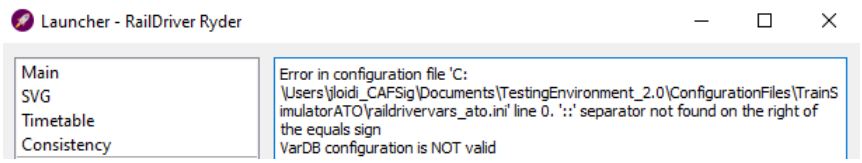
Probak bi zatitan banatuko dira, esan bezala, bi programa garatu baitira, *RailController* eta *TrainSimATO*.

7.1 *RailController* programaren proba kasuak

Programa honetan *RailDriver* kontrolatzailearen funtzionamendua egiaztatzeaz gain, sarrerako parametroen funtzionamendua ere behatu nahi da.

7.1.1 1. kasua

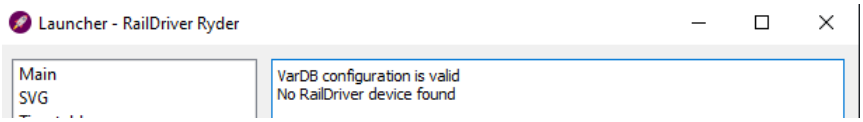
- **Deskribapena**: konfigurazio fitxategien formatu okerra
- **Betekizunak**: *A.2*, *A.3*
- **Sarrera**: *VarDB*ren konfigurazio fitxategiak formatu okerra izango du.
- **Esperotako emaitza**: programak errorea emango du, adieraziz konfigurazio fitxategia ez dela egokia.
- **Emaitza**: espero moduan fitxategiak errorea ematen du, 7.1 irudian ikusi daitekeen moduan.



7.1 Irudia: 1. proba kasuaren emaitza

7.1.2 2. kasua

- **Deskribapena**: sarrerako parametroak zuzenak dira, baina ez dago *RailDriver* kontrolatzailea konektatuta.
- **Betekizunak**: *A.1*
- **Sarrera**: ez da *RailDriver* kontrolatzailea konektatuko eta ingurunea martxan jarriko da.
- **Esperotako emaitza**: errorea emango du esanez ez duela kontrolatzailerik aurkitu, eta programa amaituko da.
- **Emaitza**: esperotako emaitza ematen du, 7.2 irudian bezala.



7.2 Irudia: 2. proba kasuaren emaitza

7.1.3 3. kasua

- **Deskribapena**: konfigurazio fitxategiak zuzenak dira, eta *RailDriver* kontrolatzailea konektatuta dago.
- **Betekizunak**: *A.1*, *A.2*, *A.3* eta *A.4*
- **Sarrera**: konfigurazio fitxategiak egokiak izango dira, eta kontrolatzailea konektatuta egongo da.
- **Esperotako emaitza**: ez da errorerik egongo, denak ondo funtzionatzea espero da.
- **Emaitza**: denak ongi funtzionatzen du, espero moduan.

7.2 *TrainSimATO* programaren proba kasuak

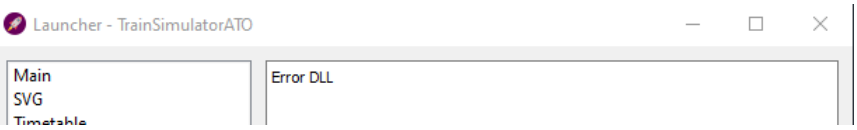
Programa honetan *TrainSimATO* programaren funtzionamendua probatzeaz gain *ATO*ak egiten duen gidatzea ere probatu nahi da, egindako programaren baitan baitago hein handi batean.

7.2.1 1. kasua

- **Deskribapena**: sarrerako balizen fitxategi okerra.
- **Betekizunak**: *B.2*
- **Sarrera**: balizen koordenatuen fitxategi formatu okerra sartzen da programan.
- **Esperotako emaitza**: programak errorea emango du eta exekuzioa amaituko da.
- **Emaitzak**: esperotako emaitzaren berdina.

7.2.2 2. kasua

- **Deskribapena**: sarrerako *dll* fitxategiaren path okerra.
- **Betekizunak**: *B.2*
- **Sarrera**: *dll* liburutegiaren path okerra sartzen da.
- **Esperotako emaitza**: *dll* fitxategia aurkitu ez duenaren errorea emango du eta programa amaituko da.
- **Emaitza**: espero moduko emaitza, 7.3 irudian ikusten den moduan.



7.3 Irudia: 2. proba kasuaren emaitza

7.2.3 3. kasua

- **Deskribapena**: konfigurazio fitxategiak ongi daude, eta ongi konektatzen da bai bideo-joko eta *ATO*rekin.
- **Betekizunak**: *B.1*, *B.2* eta *B.3*

- **Sarrera**: testing ingurunea exekutatzen da konfigurazio fitxategi egokiekin.
- **Esperotako emaitza**: exekuzio eta gidatze egokia espero dira.
- **Emaitza**: programaren exekuzioa egokia izan da eta *Undorf-Regensburg* ibilbidea burutu ondoren gelditu den grafikoak (6.11 irudia) erakusten du gidatzea abiadura mugen barruan egin dela eta oso egonkorra izan dela.

7.2.4 4. kasua

- **Deskribapena**: programaren funtzionamendua *RailDriver* kontrolatzailearekin.
- **Betekizunak**: *B.1*, *B.2*, *B.3* eta *B.4*
- **Sarrera**: testing ingurunea exekutatzen da konfigurazio fitxategi egokiekin, eta behin *ATO*a gidatzen ari denean, *RailDriver* kontrolatzailearekin kontrola hartuko da eta eskuz mugituko da trena.
- **Esperotako emaitza**: exekuzioa ondo hasiko da, aurreko proba kasuan bezala, eta eskuzko kontrola aktibatzean, ongi erantzungo dute *ATO*ak eta bideo-jokoko trenak.
- **Emaitza**: programaren exekuzioa egokia izan da eta kontrolatzailea aktibatzean eskuz ondo gidatu du, espero zen moduan.

8 Garapenaren zailtasun nagusiak

Proiektu guztietan bezala, arazoak ohikoak izaten dira bai planifikazio garaian eta baita garapenean ere. Arazo horiek azaleratzea eta aztertzea oso gomendagarria eta positiboa izan daiteke beste proiektu baterako, edo egindako proiektuarekin jarraitu behar duen beste pertsona batentzako. Horregatik, arazoa identifikatzeaz gain, arazoari nola egin zaion aurre edo zein soluzio proposatu den argitzea garrantzitsua izango da.

8.1 C++03 bertsioa

Aurretik aipatu da C++en 2003 bertsioa erabiltzen dela konpilatzeko, eta bertsio nahiko zaharra denez, hainbat ezaugarri berri ez daude erabilgarri, *thread*ak sortzea edo denbora nanosegundutan neurtzea, esaterako. Horregatik, funtzionalitate hauek lortzeko beste bide batzuk bilatu behar izan dira. Azelerazioa kalkulatzeko beharrezkoa izan da bi abiaduren arteko denbora zenbatzea, baina exekuzio denbora neurtu beharrean, dagokion harian *sleep* bat egin da eta hortaz, azelerazioa kalkulatzeko itxaron den denbora hartu da kontuan. Bestetik, hariak sortzeko enpresak sortutako liburutegi batez baliatu da, *CAF_OS* izenekoa. Liburutegi konplexu bat da hau, eta hainbat funtzionalitate eta datu-mota eskaintzen ditu, tartean modu erraz batean *thread*ak sortzeko aukera.

8.2 *API*ekin lan egitea

Erabili diren *RailDriver* eta *Train Simulator* tresnak ondo menperatzea beharrezkoa izan da proiektua garatzeko, eta ez direnez kode irekikoak, enpresek eskaintzen dituzten *API*ak erabiltzera mugatuta gaude. Horregatik, beharrezkoa da dokumentazioa izatea, eta kasu hauetan ez da aurkitu dokumentazio ofizialik, eta interneten aurkitutako informazioarekin burutu dira *API* hauen integrazioak. Tresna hauen enpresek eskaintzen dituzten liburutegiak *dll* formatua dute, eta hauek erabiltzeko lanak erraztu ditu *Softwarearen Garapenerako Tresna Aurreratuak* irakasgaian landutakoak.

8.3 Datuekin arazoak

Aipatu da *ATO*ak ongi funtzionatzeko aurretik definitu behar dela exekutatu behar duen agertoki edo ingurunea, hauek irudikatzeko *QML*ak erabiliz (abiadura maximoak, balizen posizioak, geltokien distantziak, trenbideko gradienteak, ...). Datu horiek ateratzeko funtzionalitatea ere gehitu da, baina hemendik ateratako datuek eta ondorengo exekuzioko datuek ez zuten bat egiten hasiera batean. Distantziaren kalkuluan nabarmentzen zen gehien aldaketa, eta kasu batzuetan kilometro bateko desberdintasuna izan zezaketen kalkuluen emaitzek bi exekuzio desberdinetan. Azken finean, errore hauen arrazoi nagusia exekuzioan programak duen karga izan da, izan ere, zikloro aldagai asko irakurri eta idatzi behar dira eta kalkuluak ere egin behar dira. Hori ebazteko, 6.4 Komunikazioa atalean azaldu den moduan, *thread* berri bat sortu da kalkulu horiek egiteko bakarrik, eta hortaz, beste exekuzio guztiarekiko independentea da eta exekuzioro emaitza ia berdinak eman ditu.

8.4 Planifikazioarekin ezustekoak

Jarraipena eta kontrolaren atalean sakonago azalduko da, baina aipatzekoa da garapenean ezusteko batzuk izan direla denborari dagokionez. Ezusteko horiek ez dira izan lan-paketeak ongi estimatu ez direlako, hauetatik independenteak diren arrazoiengatik baizik, produktu batzuen erosketan enpresak denbora asko behar izatea eta taldean lan egin behar zen langileekin elkartu ezin izana haien lan-karga handiagatik eta lan hauen lehentasun handiaren eraginez, hain zuzen.

9 Jarraipena eta kontrola

Egindako planifikazioa eta mugarriak errespetatu diren ala ez ikusteko, proiektuan zehar egindako jarraipen eta kontrola hasieran estimatutakoarekin alderatu behar da, eta horren arabera ondorio batzuk atera. Horregatik, atal honetan planifikazioan egindako *Gantt* diagrama eta estimazioak benetakoarekin konparatuko dira, eta egindako aurreikuspen horien zenbateraino bete diren neurtu. Konparatze aldera, egindako ordu kopuruen taula ikus daiteke 9.1 taulan.

Proiektuaren hasieran plangintza bat definitu zen, bertan ataza eta denbora aurreikuspen batzuk egin ziren, eta orokorrean bete badira ere, mugarriak ez dira errespetatu azken fasean, nahiz eta proiektua epe barruan itxi den. 9.1 irudiko *Gantt* diagrama begiratuz, antzeman daitezke non dauden aldaketa nagusiak, batez ere *TrainSimATO*en garapenari uste baino lan-ordu gehiago eskaini baitzaizkio, 40 ordu, zehatzagoak izateko.

Hasteko, aipatzekoa da lehen lan-paketeak burutzeko, hau da, plangintza, ingurunea prestatzea, ikasketa-prozesua eta *RailController*en garapena, aurreikuspen guztiak bete direla eta mugarriak ondo errespetatu direla, 2021eko apirilaren 14rako amaituta baitzegoen lehen zati hau.

Ondoren, *TrainSimATO* programan, estimatu baino denbora gehiago behar izan da, batez ere eszenarioak edo agertokiak irudikatzeko sortu beharreko *QML*ekin. Horretaz gain, aipatu behar da garapenaren hirugarren zatia uste baino beranduago hasi zela, astebete zehatzagoak izateko. Atzerapen honen zergatia enpresari dagokio, izan ere, *Train Simulator* bideo-jokoan probatu nahi zen agertokia erosi beharra dago, eta enpresan zerbait erosteko prozedura nahiko motela, konplexua eta luzea da.

Bestetik, langileentzat garai konplikatuak izan ziren maiatza eta ekaina, hainbat proiektu baitzeuzkaten martxan, eta horrek beste edozein lanetarako oztopoak jartzen zituen, atzerapenak eraginez. Gainera, proiektu honen garatzaileak hirugarren fasea burutzeko *PER*eko taldekideekin egin behar zuen lan, haiek garatutako tresnarekin egin behar baitzen lan, eta haien denbora faltak eta lehentasunek proiektuaren garapenean denbora galtzea ekarri du. Tarte honetan, lan-karga txikiagoa izan da dokumentazioaren garapenean eta *Train Simulator 2021* bideo-jokoaren funtzionamenduak aztertzen eman dira ordu hauek. Horregatik, proiektuaren amaierako

azken zatia ez da guztiz amaitu, proiektuaren irismenean aurreikusi genezaken moduan, eta larrialdietako balaztaren inplementazioa burutu gabe gelditu da, denbora falta eta aurreko zatian azaldu denagatik.

Ataza	Estimazioa	Egindakoa
Prestaketa eta dokumentazioaren hasiera	20	20
Formakuntza	25	25
*RailController*en garapena	70	70
*TrainSimATO*en garapena	160	200
PER ereduaren integrazioa	80	65
Dokumentazioaren amaiera	15	15
Aurkezpenaren garapena	5	5
Guztira	375	400

9.1 Taula: Proiektuko lan-paketeak burutzeko ordu kopurua

Lan-paketearen izena	Atazaren izena	Hasiera data	Amaiera data
Plangintza eta jarraipena	Egin beharrekoa eta dokumentazioa definitu	2021/03/01	2021/03/05
	Excel orriak prestatu	2021/03/05	2021/03/05
Ingurunea prestatu	Programak instalatu	2021/03/08	2021/03/08
	Erabiliko den kodea eta tresnak eskuratu	2021/03/09	2021/03/11
	Informazio sistema martxan jarri	2021/03/11	2021/03/12
Ikasketa prozesua	ATOaren funtzionamendua eta prozesua ikasi	2021/03/15	2021/03/17
	Enpresako tresnak ezagutu eta erabiltzen ikasi	2021/03/17	2021/03/19
	Erabiliko diren teknologiak ezagutu	2021/03/19	2021/03/19
RailController-en garapena	Datu-baseko beharrezko aldagaiak identifikatu	2021/03/22	2021/03/23
	RailDriveraren APIa ikertu	2021/03/23	2021/03/23
	Programaren garapena	2021/03/23	2021/03/29
	Probak egin eta demoa prestatu	2021/03/30	2021/03/31
	RailDriverRyderaren dokumentazioa garatu	2021/04/12	2021/04/14
TrainSimATO-ren garapena	ATOak gidatzeko zein aldagai behar dituen identifikatu	2021/04/15	2021/04/15
	Train Simulatoren APIa ikertu	2021/04/16	2021/04/20
	Programaren garapena	2021/04/20	2021/05/14
	Erabiliko den eszenariotik datuak atera	2021/05/05	2021/05/07
	Eszenarioaren errepresentazioa QMLn	2021/05/05	2021/05/13
	Probak egin eta demoa prestatu	2021/05/14	2021/05/21
	TrainSimulatorATOren dokumentazioa garatu	2021/05/24	2021/06/11
PER ereduaren integrazioa	Train Simulator aztertu funtzionamenduak identifikatzeko	2021/05/31	2021/06/04
	Ingurunea prestatu eta eredua martxan jarri	2021/06/14	2021/06/24
	Larrialdietako balaztaren inplementazioa	-	-
	PERen dokumentazioa gehitu	2021/06/28	2021/07/06
Proiektuaren itxiera	Dokumentazioa amaitu	2021/07/07	2021/07/23
	Aurkezpena garatu	2021/07/19	2021/07/23

9.1 Irudia: Proiektuaren amaierako Gantt diagrama

10 Ondorioak eta etorkizunerako lanak

Behin proiektuaren zatirik garrantzitsuenak amaituta, proiektua itxi aurretik egindako lana errepasatu eta ondoriorik garrantzitsuenak zein diren azpimarratuko dira. Gainera, lan hau oinarritzat hartuta honen gainean egin daitezkeen hobekuntza eta etorkizunerako lanak deskribatuko dira.

10.1 Ondorioak

Hasteko, esan beharra dago proiektuaren zatirik garrantzitsuenak arrakastaz burutu direla. Enpresak dituen helburuak ongi bete dira eta bide eman da garapen bide berrietarako, hala nola, bideo-jokoen errealismoa aprobetxatzea orain arteko tresnak osatzeko eta ikusgarriagoak egiteko. Bestetik, egia da ez dela proiektua hasieran espero moduan burutu, izan ere *PER* eredua ez da gai izan *Train Simulator* bideo-jokoko seinale eta semaforoak detektatzeko denbora errealean, *streaming* arazoengatik, baina software garapenari dagokionez proiektua arrakastatsua izan dela azpimarratu behar da.

Lehen faseari dagokionez, *RailDriver* kontrolatzailearekin *VarDB* datu-basea ongi kontrolatzea lortu da, eta hainbat funtzio inplementatu dira. Ondoren, *ATO* sistemaren bitartez *Train Simulator 2021* bideo-jokoaren automatizazioa arrakastatsua izan da, bertako trena nahi denean automatiko gidatu dadin.

PER ereduaren integrazioari dagokionez, ez da software garapen handirik burutu, azken finean, zati honen helburua *Train Simulator 2021* bideo-jokoa erabiliz eredua entrenatzeko eta proban jartzeko irudiak sortzea baitzen, benetako irudi grabatuak lortzeko behar den dirua aurreztuz. Ondoren, eredua denbora errealean erabiliz, bideo-jokoko irudiak eredura *stream*eatu eta horrela seinale eta semaforoen informazioa zuzenean lortu nahi zen. Lehen helburu hori bete da, hau da, lortu da bideo-jokoko irudiak hartuz eredua entrenatu eta erabiltzea, eta mundu errealeko agertoki berdinak deskargatu daitezkenez, erabilgarria izango da. Hala ere, ez da posible izan *streaming* bidez ereduak ezer detektatzea, zarata asko sortzen baita. Horren ordez, aurrez grabatutako bideoak erabili dira eredua entrenatu eta

probatzeko, eta emaitza oso onak eman ditu. Hori horrela izanik, larrialdietako balaztaren funtzionalitatea ezin inplementatu izana eragin du, ez baita denbora errealean erabiltzen eredua.

Bestetik, nahitaezkoa da aipatzea hasieran izandako trabak. *RailDriver* eta *Train Simulator* tresnen liburutegiak erabiltzeko garaian, hauen dokumentazioa derrigorrezkoa da, *dll* formatuko liburutegi batzuk direnez, ezin delako zuzenean jakin zein funtzio dauden erabilgarri. Baina dokumentazio ofizialik ez dago, ez behintzat webgune ofizialetan, eta beraz, beste erabiltzaileek sortutako dokumentazioak erabiltzea besterik ez da geratu. Behin dokumentazio honekin, garapena erritmo handiagoan joan da eta ez da traba handirik izan.

Horretaz gain, azpimarratu behar da gidatze egokia eta egonkorra lortu dela. Printzipioz, bideo-jokoa ez dago eginda kanpoko software batek automatikoki kontrolatzeko, baina *ATO*arekin egin diren probetan ez da abiadura aldaketa handirik izan, eta abiadura maximoa ere ez da inoiz gainditu, hots, abiadura profil egonkor bat osatu da. Horregatik, ondoriozta dezakegu *Train Simulator* tresna egokia dela tren baten portaera simulatzeko eta *ATO*aren moduko tresnak probatzeko.

Aitzitik, *RailDriver* kontrolatzaileari dagokionez, aipatu beharra dago oso tresna ezaguna eta erabilia dela, baina dituen palanken sentikortasuna ez dela oso ona, eta zaila dela balio zehatzak lortzea. Horregatik, eskuz zehaztasunez gidatzea ez da erraza, batez ere *TBL* palankari dagokionean, baina ez da arazo handia hau, orokorrean ondo gidatu baitaiteke. Funtzionalitateei dagokienez, berriz, kontuan hartu behar da *VarDB* datu-basearen gaineko aldaketak egiten dituela, eta ez dela zuzenean *Train Simulator*ekin komunikatzen. Horregatik, ezin izan dira trenaren argiak eta klaxona kontrolatu *RailDriver* kontrolatzailea erabiliz, hauek bideo-jokoko aldagaiak baitira, eta ez dute zerikusirik *VarDB* datu-basea eta *ATO*arekin.

Laburtuz, *ATO* eta *Train Simulator 2021* oso tresna boteretsuak eta konplexuak dira, enpresa handiek sortutako produktuak dira eta hauek erabiltzea oso esperientzia aberasgarria izan da. Alde batetik, *ATO* sistema (*CAF Signalling* enpresan garatua) tren bat automatikoki gidatzeko gai dela kontuan hartu behar da, horrek dakarren guztiarekin, izan ere, kalkulu asko egiteaz gain, ehundaka aldagai hartu behar dira kontuan, eta trenaren eta bidaiarien segurtasuna bermatu behar da. Bestetik, *Train Simulator 2021* bideo-jokoaren errealismoa aipatu behar da. Oso ezaguna denez, eta hein handi batean pertsonalizatzeko aukera ematen duenez (ibilbideak sortuz, eraikitako trenak jokoan gehituz eta abar), milaka erabiltzailek bideo-jokoa aberasteko ehundaka ibilbide, tren eta seinale gehitzen dituzte, eta beste erabiltzaileen eskura jarri.

Honetaz gain, proiektuaren garapenaren $\%100$ enpresa batean burutzeak aldaketa handia suposatu du proiektuaren garatzailearen lan egiteko moduan. Esperientzia honen bitartez, benetako enpresa batean lan egitea zer den ikasi da, eta baita enpresa mailan talde lanean nola aritu behar den. Gainera, benetako software proiektuekin lehen hartu-emanak izan dira, eta jende aditu eta langilearekin inguratzeak ere lagundu du, eta etorkizunerako asko balioko du.

Amaitzeko, eta proiektuaren kalitateari erreparatuz, osorik bukatu ezin izanak eragina izan du, eta proiektu honetan jarraitzea egokitzen bada, hasierako betekizunak burutzea izango litzateke lehen lana. Hala ere, egindako lanari erreparatuz, hasteko, erakutsi da komunikazioak eraginkorrak eta egokiak izan direla, eta ez dela inongo datu galtzerik ematen. Ondorioz, aukeratutako agertokietan eman den gidatzea ongi burutu da, eta aipatu den moduan, gidatzean izan den abiadura oso egonkorra izan da, eta muga guztiak errespetatu dira (abiadura, geltokiak, ...). Honetaz gain, mugarriak ondo errespetatu dira, lehen bi garapen faseak aurreikusi bezala bete dira, eta nahiz eta azken fasea ezin izan den guztiz amaitu, proiektuari bukaera mugen barruan eman zaio. Horregatik, ondorio nagusi moduan proiektuaren kalitatea egokia dela esan genezake.

10.2 Etorkizunerako lanak

Garapena amaitu ondoren, etorkizunean egin daitezkeen hobekuntza edo eguneraketak aipatu behar dira. Hasteko, aurreko atalean azaldu den moduan *PER* eredura *Train Simulator* bideo-jokoko irudiak denbora errealean *stream*eatzea lortu da, baina irudietan zarata asko sortzen da, eta ez da ezer detektatzea lortzen. Horregatik, lehen lana hau konpontzea izango litzateke eta horretarako, *streaming* zerbitzu edo liburutegiren bat ongi konfiguratu beharko da eta ingurune egoki bat osatu. Horrela, *ATO*arekin komunikatzeko aukera egongo litzateke eta larrialdietako balaztaren inplementazioa egiteko aukera egongo litzateke.

Bestetik, gehiago aprobetxatu nahi da bideo-jokoaren errealismoa, eta *ATO*az gain *ERTMS* sistemarekin ere integra liteke bideo-jokoa. *ERTMS* Europar Batasunaren ekimena da, eta trenbide-sareen elkarreragingarritasuna bermatu nahi du, mundu osoan estandar bakarra sortuz [11]. *ERTMS* maila desberdinak daude, bakoitza bere segurtasun neurri eta muga desberdinekin, eta beraz, bideo-jokoaz baliatu daiteke informazio hau sortzeko. *Train Simulator*etik lortu daitekeen informazioaren arabera lehen edo bigarren mailako *ERTMS* sistema inplementatu daitekeela aurreikusten da, baina proiektuaren garapenean behatu den informazio faltagatik, nahiko konplexua eta zaila ikusten da informazio sortze hori.

Eranskina

A. Programen erabilera-gidak

Programak garatu ondoren, hauek nola erabili behar diren eta zer behar duten azaltzeko erabilera-gidak sortu dira eta enpresari helarazi zaizkio, eta atal honetan erabilera gida horiek azalduko dira. Aipatu ingelesez idatzitako gidak direla, enpresako dokumentazioaren hizkuntza nagusia hau delako.

RailController

Program to change variables values from VarDB using the RailDriver cab controller.

Required stuff

- RailDriver cab controller connected to the PC
- All configuration files (see next section) in ´ConfigurationFiles´ folder

Values

- Reverser
- TBL
- Holding brake
- Cab selection
- Skip request
- Skip revoke
- ATO engage
- ATO disengage
- Emergency brake
- LED display to show TBL value

Launch parameters

This program needs 2 parameters:

- VarDB variables .ini filename
- RailDriver calibration file, created with RDCalibration program

How to launch

This program is launched with TELauncher and loads all the configuration files from ´ConfigurationFiles´ folder.

Return values

This program does not return any value.

TrainSimATO

Program to connect to Train Simulator and drive the train using ATO. It's necessary to create a scenario whitch is available in the game, and create some balises according to the GPS coordinates of that scenario. To get data from the game it's possible to use the program without launching with all the environment, and without any parameters. That way, in one manual launch of the scenario, the program will read game data in loop and it will create a csv with GPS coordinates, distance travelled, train speed, gradient, train orientation and change of max speed. For the change os max speed it's necessary to connect the RailDriver controller and push manually the "Alarm"button everytime the max speed changes.

Features

The program is able to read driving variables from VarDB and send to Train Simulator 2021. These variables include TBL, Reverser, Holding Brake and Emergency Brake. In addition, values are read from the game to give response to those driving commands, speed, acceleration and train position, among others. Also, when the train stops at a stopping point, the train doors will open and close automatically.

Required stuff

- Train Simulator 2021 game ready to drive in a scenario
- All configuration files (see next section) in ´ConfigurationFiles/TrainSimulatorATO´ folder
- The game scenario represented in QML in ´ScenariosSystem´ folder

Launch parameters

This program needs 3 parameters:

- VarDB variables .ini filename
- Balises coordinates .csv filename
- RailDriver.dll file path filename

How to launch

This program is launched with TELauncher and loads all the configuration files from ´ConfigurationFiles´ folder.

Return values

This program does not return any value.

Glosarioa

Atal honetan dokumentu guztian zehar aipatu diren sigla eta terminoen laburpen txiki bat egiten da, bakoitzaren azalpen eta euskarazko itzulpen batekin. Azalpen modura, termino bakoitzaren amaieran dokumentuko zein orrialdetan egiten zaion erreferentzia agertzen da. Ez dira hitzaren agerpen guztiak agertzen, ez baitira beti erreferentziatzen.

ATC Automatic Train Control, edo euskaraz, Trenaren Kontrol Automatikoa. *ATC*ak abidadura kontrolerako segurtasun sistema bat bermatzen du. Ikusi orrialdeak: 2, 69

ATO Automatic Train Operation edo euskaraz, Trenaren Operazio Automatikoa. Trenak automatikoki gidatzeko sortu den tresna edo softwarea, hainbat software zati txikiagoz osatuta. Ez nahastu ATC eta ATPrekin.. Ikusi orrialdea: i

ATP Automatic Train Protection, edo euskaraz, Trenaren Babes Automatikoa. Trenen gidatzea gainbegiratzen duen segurtasun-sistema bat da, larrialdi-balazta aplikatzen duena edo segurtasun-baldintza batzuk betetzen ez direnean beste ekintza batzuk eragozten dituena. Ikusi orrialdeak: 2, 69

DMI *CAF Signalling* enpresan garatutako softwarea, *ATO On-Board* sistema biltzen duen interfaze grafikoa. Ikusi orrialdea: 39

ERTMS European Rail Traffic Management System, edo euskaraz Trenbide-trafikoa Kudeatzeko Sistema Europarra. Europar Batasunaren ekimena da, eta trenbide-sareen elkarreragingarritasuna bermatu nahi da, mundu osoan estandar bakarra sortuz. Ikusi orrialdeak: 25, 63

PER Ingeleseko *perception* hitzetik eratorria, enpresako azpitalde bat da, eta izen bereko ikusmen artifizialean oinarritutako seinale eta oztopo detektatzaile baten garapenaz arduratzen dira. Ikusi orrialdea: i

RailDriver Tren kabina-kontrolatzailea, oso erabilia tren simulatzaileen zaleen artean. Hainbat palanka eta botoi programagarri ditu. Ikusi orrialdea: i

Redmine Kode irekiko proiektuak kudeatzeko web-aplikazioa, langile bakoitzaren atazak kudeatzeko eta aurreikuspenak egiteko oso tresna egokia. Ikusi orrialdea: 21

Ryder *CAF Signalling* enpresan garatutako softwarea, tren baten portaera simulatzeko. Ikusi orrialdea: 4

Shunting Bagoiak ordenatzeko eta tren osoak osatzeko (edo alderantziz) egiten den maniobra da. Ikusi orrialdea: 47

Train Simulator 2021 Dovetail Games-ek garatutako bideo-jokoa, errealista eta arrakastatsua, batez ere mundu errealeko trenbide sareak eta trenak dituelako. Gainera, edonork sortu dezake agertoki edo tren bat eta publiko egin. Ikusi orrialdea: i

Bibliografia

[1] CAF Signalling enpresaren webgunea. https://www.cafsignalling.com/eu/enpresa/, 2021. Ikusi 1 orrialdea.

[2] *ATO*aren inguruko Wikipediako artikulua. https://en.wikipedia.org/wiki/Automatic_train_operation, 2021. Ikusi 2 orrialdea.

[3] RailDriver kontrolagailuaren webgunea. http://raildriver.com/, 2021. Ikusi 4 orrialdea.

[4] Agenda aplikazioa, egunero egin dena deskribatzeko erabili den programa. https://www.microsoft.com/es-es/p/agenda/9wzdncrfhzgk, 2021. Ikusi 19 orrialdea.

[5] Online Latex editorea. https://es.overleaf.com/, 2021. Ikusi 19 orrialdea.

[6] Shift2Rail iniziatibaren webgune ofiziala, entitate hau *ATO*aren eta ingurukoen estandarrak sortzeaz arduratzen da. https://shift2rail.org/, 2021. Ikusi 28 orrialdea.

[7] P.I. Engineering enpresako SDK deskargatzeko webgunea. https://xkeys.com/software/developer/developerwindowssdk.html, 2021. Ikusi 33 orrialdea.

[8] Bi GPS koordenatuen arteko distantzia ortodromikoaren kalkulua haversinuaren bitartez. https://www.movable-type.co.uk/scripts/latlong.html, 2021. Ikusi 39 orrialdea.

[9] Regio BR 442 'Talent 2' trenaren eskuliburua. https://steamcdn-a.akamaihd.net/steam/apps/258656/manuals/DB_BR_442_Talent_2_Manual.pdf, 2021. Ikusi 42 orrialdea.

[10] Steam bideo-joko dendaren webgunea. https://store.steampowered.com/, 2021. Ikusi 47 orrialdea.

[11] European Rail Traffic Management System. https://en.wikipedia.org/wiki/European_Rail_Traffic_Management_System, 2021. Ikusi 63 orrialdea.