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

INDUSTRIA ELEKTRONIKAREN ETA AUTOMATIKAREN INGENIARITZAKO GRADUA

GRADU AMAIERAKO LANA

PLC BATEN INTEGRAZIOA MANUFAKTURA SISTEMA MALGU BATEAN: ODK-JADE PLATAFORMA

Alumno: García, Gutiérrez, Diego

Director: Casquero, Oyarzabal, Oskar

Curso: 2019-2020

Data: 2019-ko Azaroaren 4-a

RESUMEN

A lo largo de este Trabajo de Fin de Grado se desarrolla el diseño de una plataforma de comunicación entre agentes JADE y un PLC. Se comunica un agente programado en lenguaje Java con el programa de un autómata usando un intermediario llamado ODK, programado en lenguaje C++, para que, en un futuro, se pueda integrar dicha plataforma en un sistema de fabricación flexible. En este proyecto se expondrán los recursos que serán necesarios para realizar dicha plataforma, así como el hardware y los programas que se van a utilizar para el desarrollo del software.

Palabras clave: JADE, ODK, PLC, Java, fabricación flexible, plataforma

ABSTRACT

Along this Final Degree Project, the design of a communication platform between a JADE agent and a PLC is developed. A Java programmed agent is communicated with a PLC program using a middleware called ODK, programmed in a C++ application, to, in a near future, be able to have this platform integrated in a flexible manufacturing system. In this project the resources needed to create the platform are explained, as well as the hardware and the programs that are going to be used to develop the software.

Keywords: JADE, ODK, PLC, Java, flexible manufacturing, platform

LABURPENA

Gradu Amaierako Lan honetan zehar, JADE agenteak eta PLC bat komunikatzeko plataformaren diseinua garatzen da. Java-z programatutako agente bat automatarekin komunikatuko da, C++-z programatutako eta ODK izeneko erdikari bat erabiliz, etorkizunean plataforma hau manufaktura malguko sistema batean integratzeko. Proiektu honetan plataforma garatzeko beharrezkoak diren errekurtsoak eta erabiliko diren hardwarea eta softwarea garatzeko beharrezkoak diren programak azalduko dira.

Gako hitzak: JADE, ODK, PLC, Java, manufaktura malgua, plataforma

Edukia

Taulak

Irudiak

1. SARRERA

Edozein erakundetan, produkzio sisteman, moldakortasun eta eraginkortasun maximoa, eta gero eta automatizazio maila handiagoak bilatzen ari dira. Gainera, merkatuaren hazkunde tasa moteldu egin da eta produktuen dibertsifikazioa handitu egin da, bezeroaren behar konkretuei egokitzeko. Fabrika klasikoek serie handietan produktu konkretu baten produkzioa egiteko diseinatuta daude. Hau dela eta, erakundeek produkzio kapazitate handiegiekin eta produktu berriak sortzeko zailtasunekin topatu egin dira. Beraz, erakundeek fabrikazio sistemaren aldaketa bat behar dute, zailtasun hauei aurre egiteko.

Dagoen merkatuan errentagarritasuna lortzeko hurrengo kriterioetan oinarritu behar dira produkzio sistemak:

- Flexibilitatea produktua ekoizteko.
- Produktuaren kalitatea .
- Produktu berriak merkaturatzeko azkartasuna.
- Beharrezkoak ez diren gastuen txikitzea.
- Prozesuen automatizazioa.
- Produktibitatearen handipena.

Manufaktura sistema malguen (MSM) inplementazioa egoera hau konpontzeko sortu egin dira. MSM-ak garraio sistema automatiko batez konektatutako hainbat lan-estazio edo makinek osatzen dute eta produktu familia ezberdinen produkzio sistema automatiko eta adimendua izatea baimentzen du. Honek, fabrikazio sistemak aldatu behar direnean sorturiko kostuak txikitzen ditu eta produktibitate eta kostu unitario erreserbatuak mantentzen ditu produktu bat seriean egitea errentagarriagoa den arte.

MSM-a era bat baino gehiagotan inplementatu daiteke. Eratako bat multi-agente sistema (MAS) bat erabiltzea da, unibertsitateko ikerketa taldeak egin duen bezala. MAS motako plataforma batean, agente anitz aplikazio banatu bat osatu egiten dute eta agente mota bakoitza betekizun bat egiteko programaturik dago. Agenteek haien artean komunikatu, negoziatu eta erabakiak hartu ditzakete, eta ondorioz, inguruko kontestuari sentikorra den aplikazio adimendua osatzen dute. Bi agente mota nagusi daude:

- Sistema agenteak. Gainontzeko agenteak kudeatzen dituzte.
- Domeinu agenteak. Aplikazio banatua osatzen dute.

MAS erabiltzen bada MSM bat egiteko, agente batzuk entitate fisikoak errepresentatu ditzaketen bitartean (makinak, garraio robotak, karga estazioak…), beste batzuk plataformaren kudeaketaz edo logikaz enkargatu daitezke.

2. TESTUINGURUA

Unibertsitateko ikerketa taldeak MSM baten diseinua egiteko asmoa du. Horretarako, proiektu handi hau zatietan banatu egin dute. Alde batetik, doktore tesi baten bitartez MAS plataforma bat diseinatu da aplikazio banatuen kudeaketa egiteko eta, beste aldetik, fabrikazioan MAS plataformak inplementatzeko partikularizazioa definitu da, TFM batean. Fabrikazioan MAS-ak inplementatzeko, logistikarekin lan egiten duten agenteez aparte, entitate fisiko bakoitzari agente bat esleitu behar zaio.

Testuinguru honetan hainbat TFG eta TFM egin dira garraio robot bat MSM batetan integratzeko. Robotak ROS "framework"-aren bitartez programatu dira eta behe mailako kontrola diseinatu da, "LowLevelControl" (robotaren operazioak) eta "TaskControl" (operazioak hasteko eta bukatzeko aginduak, erroreen abisu ematea, aldagaien gordeketa etb.) operazioak diseinatuz eta inplementatuz. Robotaren programazioaz aparte, robotari agente bat esleitzeko ROS-JADE plataforma diseinatu da, robota eta agentearen arteko komunikazioa baimentzeko.

Ikerketa taldearen hurrengo pausua MSM-an makinak integratzea da, izan ere orain arte egin diren proiektuetan makinek dituzten agenteen jokabidea bakarrik simulatu egin da. Makinak lan konkretuak exekutatzeko pentsatuak daude baina MSM baten barnean makinaren produkzioa bere ingurunean dagoen baldintzen arabera izango da. Adibidez, makina bat bertan behera geratuz gero beste makina batek lehen makinaren lan karga hartu beharko du eta horretarako lan karga hartuko duen makinak jakin beharko du beste makina eskuragarri ez dagoela. Hau dela eta, makinei agente bat esleitzea ezinbestekoa da MSM sistema batean integratzeko.

Makinak PLC baten bidez kontrolatuta daude. PLC-ak automata lengoaiarekin programatu behar dira eta lengoaia hau ezin da bateratu zuzenean agentearekin. Hau dela eta, proiektu honen xedea PLC bat agente batekin konektatzeko eta informazioa trukatzeko plataforma bat diseinatzea da. Plataforma hau egiteko "MiddleWare" izeneko erdikari baten laguntza erabili behar da, garraio robotan diseinatutako ROS-JADE plataforma bezala.

3. HELBURUAK

Proiektu honen helburu nagusia PLC bat JADE estrukturaz programatutako agente batekin komunikatzea da, MAS batean PLC erreal bat izateko aukera lortzeko.

Helburu nagusi hau lortzeko hurrengo bigarren mailako helburuak bete behar dira:

- Open Controller bat programatzen ikastea KOP lengoaia erabiliz proiektuan inplementatzea.
- ODK software paketearen funtzionalitatea definitzea eta C++ lengoaiarekin PLC baten funtzioak sortzea Open Controller baterako.
- Java lengoaiari buruz ezagutza lortzea.
- C++ lengoaian programatutako aplikazio bat Javan programatutako beste batekin komunikatzen lortzea.
- IntelliJ IDEA programa erabiltzen ikastea agenteen programazioa egiteko. Agenteen kasu praktiko bat sortu proiektuan integratzeko.
- Aurreko guztiarekin MAS plataforma baten funtzionalitatea erakustea aplikazio praktiko batekin.

4. ONURAK

Ingurugiro komertzial batean produkzio makinak MSM batean integratzean hainbat onura sortu egiten ditu. 3 ataletan banatu dira: onura sozialak, onura teknikoak eta onura ekonomikoak.

4.1. Onura sozialak

Industria iraultza sortu zenetik makinak gizakiaren esku lana ordezkatu egin dute. Duela urte batzuk arte, makinak lan errepikakorrak egitera mugatuta zeuden eta gizakiaren adimenarekin konbinatu beharra zegoen produkzio eraginkorra lortu nahi bazen. Makina bat MSM batean integratzean, makinari adimena ematen zaio eta erabakiak hartzeko kapazitatea izango du. Adimen hau gizakiaren parte hartzea txikitu egingo du produkzioan, eta ondorioz, giza erroreak ere ezabatuz joango dira. Giza erroreak ezabatzean langileen segurtasuna bermatuko da fabriken barneko istripuak murriztuz.

4.2. Onura teknikoak

Hiru onura tekniko nagusi antzeman daitezke makinak MSM batean integratu egiten direnean.

Alde batetik, aurretik esan bezala, giza erroreak murriztuko dira. Arlo teknikoan, produkzioan erroreak ezabatzean produktuaren kalitatea handituko da.

Bestetik, MSM batean integratutako makinaren bat bertan behera geratuz gero, beste makinek makina horren lana hartu dezakete automatikoki. Honela produkzioaren efizientzia hobetzen da.

Azkenik, MSM batean makinek ez dute zergatik beti programa bera izan behar. Makinak sare batera konektatuta daude eta sare horretan zerbitzari lana egiten duen ordenagailu batek makina guztien programak izan ditzake. Honekin makinak programa bat edo bestea izateko aukera du egoeraren arabera. Azken batean, malgutasuna asko handitzen da sistema honekin.

4.3. Onura ekonomikoak

Aztertuko den azken onura aspektu ekonomikoan dituenak dira. Egia esan aurreko onura guztiak aspektu honetan eragina dute zuzenean edo zeharka. Malgutasuna adibidez, produkzioa merkatuari moldatzeko erraztasuna izatea baimentzen du. Honen ondorioz, produktuak merkaturatzeko abiadura handitzen da, irabaziak handituz.

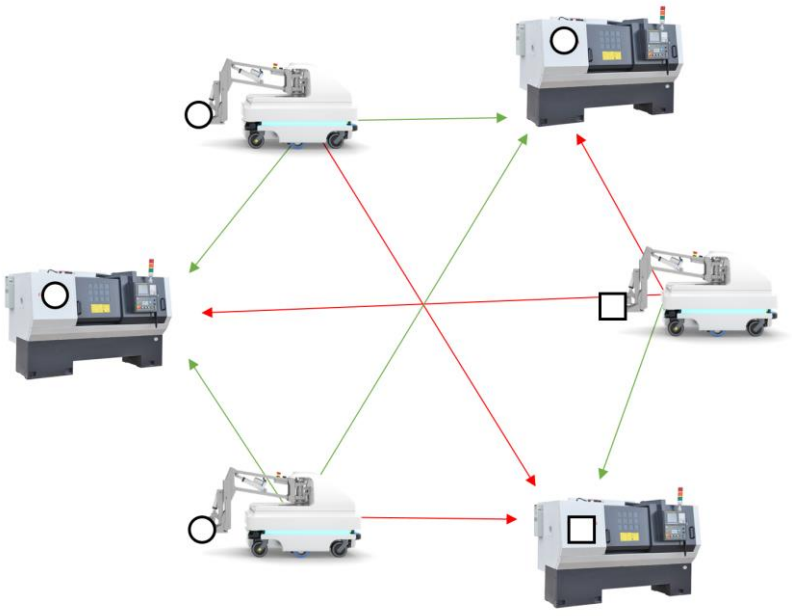
Beste aspektu ekonomiko bat gastuen txikitzea da. Produkzio klasikoan inbertsio oso handiak egin behar dira produktu berri bat merkaturatzeko. MSM batean integratutako makinak produktu berriak kantitate txikietan ekoizteko kapazitatea dute, seriean produzitzeko errentagarriak diren arte.

5. ESPEZIFIKAZIO TEKNIKOAK

Garatu egin den proiektua erabakiak hartu ahal duen robot-makina plataforma moldakor bat da, multi-agente plataforma batez baliatuz, manufaktura sistema malgu batean inplementatzeko. CNC makina anitzek pieza mota konkretu bat mekanizatu dezaketela suposatu da eta simulatutako hainbat robotek piezak makinei eraman behar diete, 1. irudian adierazten den sarea osatuz. Bi parametroen arabera banatuko dituzte piezak:

- Robotari heldu zaion pieza motaren arabera.
- Makinek duten lan kargaren arabera.

Alde batetik, pieza mota makinaren gaitasunaren arabera izango da. Makina bakoitza pieza mota konkretu bat mekanizatzeko programatuta dago eta ondorioz, robotak jakin beharko du daukan pieza zein makinek onar dezakete. Beste aldetik, pieza hori onartzen duten makinetatik, zein izango den hoberena erabaki beharko du. Kasu honetan, makinetan, biltegi automatiko batean, piezak gordetzen dira, mekanizatzen dagoen pieza bukatzean, gainontzeko piezak mekanizatzeko. Beraz, robota pieza konkretu hori onartu ahal duten makinetatik biltegian pieza kantitate txikiena duen makina aukeratuko du.



1. Irudia: Robot-makina plataforma

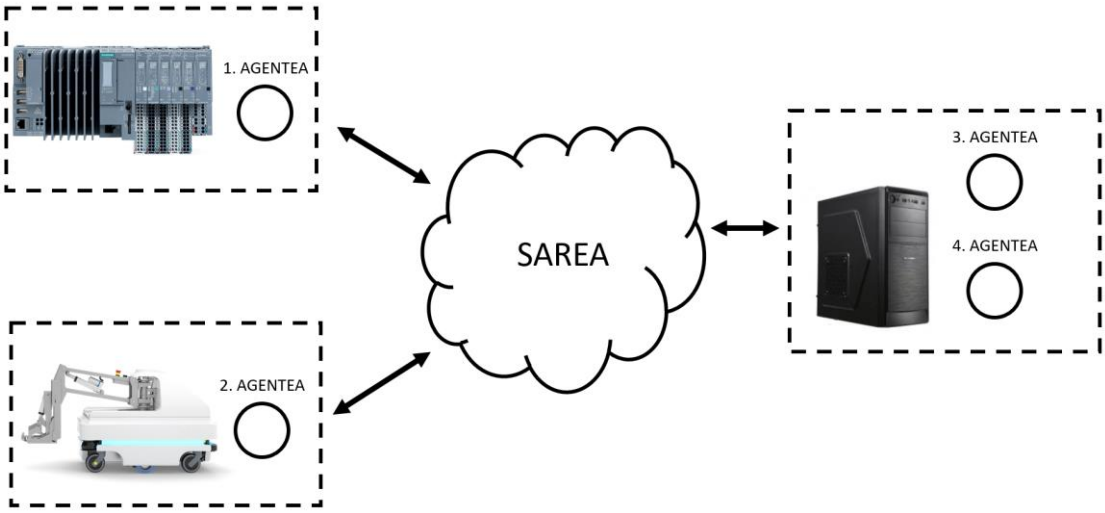
Sistema hau funtzionala izateko hurrengo baldintzak betetzeko diseinatu da:

- Komunikazioa roboten eta makinen artean bidirekzionala izan behar da. Robotek makinaren informazioa jaso behar dute eta azken honek pieza jasotzeko prestatu behar da.
- Makinaren bat matxuratuta egon daitekela kontuan hartu behar da, eta sistema funtzionatzen segitu behar du.
- Robotak ezin du pieza bat eraman makina bati honek onartu arte.
- Makinek bat lan karga maximo bat dute, ezin dute piezarik onartu hori gertatzekotan. Lan karga unitatetan ematen da, makinaren biltegi automatizatuan dauden pieza unitateak, hots.
- Robot bat baino gehiagok aldi berean ezin ditzakete makina berari haien pieza bidali.
- Makina eta robot batek informazioa trukatzeko bitartekari batzuen beharra dute, izan ere, bakoitza bere programazio lengoaia partikularra du. Proiektu honetan makinak erabiliko dituen bitartekariak bakarrik diseinatuko dira, robotaren funtzioa simulatu egingo da eta.

Baldintza hauek direla eta, JADE Java estruktura, Siemens Open Data Kit (ODK) erreminta eta ET200SP Open Controller-a erabili dira.

5.1. JADE estruktura

Hasteko, robot-makina plataforma diseinatzeko makinak robotarekin komunikatzeko sistema bat izan behar du. Eredu industrial batetan oinarritzen bagara, robot eta makina bat baino gehiago egongo direla suposatuko da, eta ondorioz, komunikazioa diseinatu behar da robot anitz makina berara joan nahiko balira bezala. Gainera negoziazio sistema baten beharra dago, izan ere piezak kriterio batzuen arabera makina batera edo bestera bidaltzea nahi da. Honetarako JADE izeneko Java-n oinarritutako programazio estruktura erabiliko da, 2. irudian dagoen adibidean adierazten den bezala.



2. Irudia: Jade estruktura

JADE-k multi-agente bidezko programazio moldakorra baimentzen du aplikazio banatuak programatzeko. Zenbait agentek, kasu honetan robot eta makina agenteak, "jokabide" batzuk izango dituzte eta hauen arabera agente bakoitza zer eginkizun dituen definituko ditu. Agenteak sare batera konektatu eta haien artean komunikatu daitezke, eta komunikazio honen bidez, erabakiak hartu ditzakete. Agenteren bat faltako balitz, adibidez makina bat, beste agenteek ez dute kontuan hartuko agente hori eta eskuragarri duten agenteekin lanean segituko litzateke.

JADE-ren bidez estrukturatutako aplikazioak exekutatzean sistema agenteak eta domeinu agenteak agertuko dira.

Sistema agenteek aplikazio banatua osatzen duten agenteen kudeaketaz arduratzen dira. 3 daude:

- AMS (Agent Management System). Agente honek paltaforma kontrolatzen du eta agenteak sortu eta itzali dezakeen entitate bakarra da.
- RMA agentea AMS agentearekin batera lan egiten du eta interfaze grafiko bat eskaintzen du agenteen kudeaketa egiteko.
- DF (Directory Facilitator) plataforman zein agente eskuragarri dagoen idazten du direktorio batean.

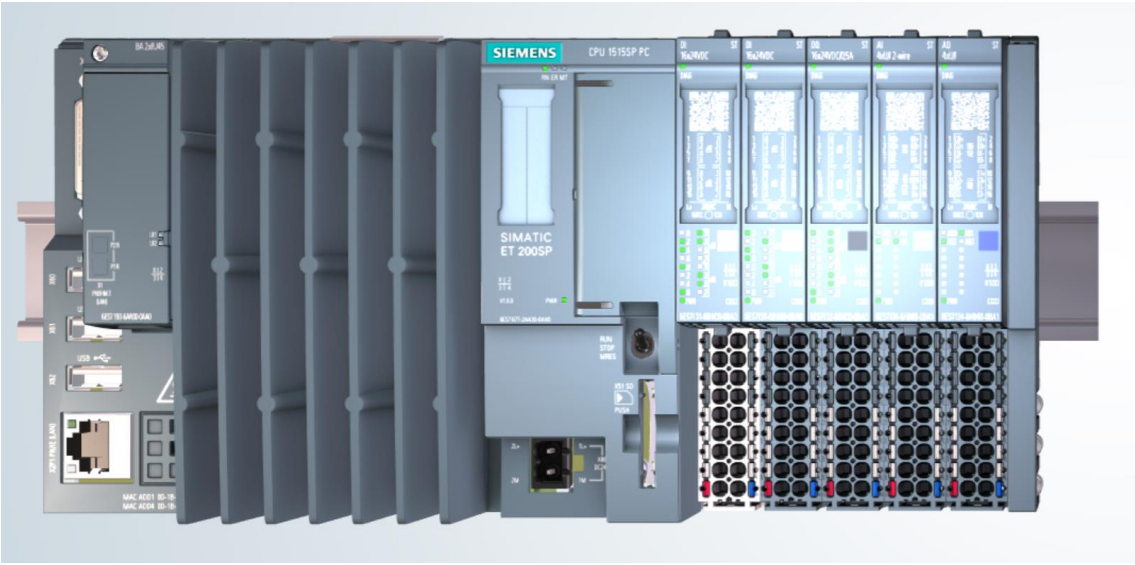
Domeinu agenteak aplikazio banatua osatzen dute. Proiektu honetan bi agente egongo dira bakarrik, robot agentea eta makina agentea.

Bi agente hauen arteko informazio trukaketa baimentzeko ACL izeneko mezu motak erabiliko dira. Hauek jasotzaile anitzei bidali daitezke, eta mezuaren semantika zehaztu daiteke. Hainbat semantikaren arabera banatu daitezke mezu motak:

- “Call For Propolsal” edo CFP. Mezu hauek proposizioak eskatzeko erabiltzen dira, “PROPOSE” edo “REFUSE” motako mezuak erantzun bezala espero ditu.
- “PROPOSE” motakoak CFP mezuei positiboki erantzuteko erabiltzen dira. CFP bidaltzaileari balore bat proposatzeko erabiltzen dira.
- “REFUSE” motakoak CFP mezuei negatiboki erantzuteko erabiltzen dira.
- “INFORM” motakoak informazioa bidaltzeko erabiltzen dira
- “ACCEPT_PROPOSAL” motakoak “PROPOSE” motako mezuen proposizioa onartzeko balio dute.
- “FAILURE” arazo bat gertatu dela negoziazioan adierazteko balio du.

5.2. ET200SP Open Controller

Makinak automata baten beharra du baina automata konbentzionalak HMI bat izatera eta makinaren programa exekutatzera mugatuta daude. Honen ondorioz ezinezkoa da edo oso zaila da automata Java-z programatutako agente batekin konektatzea. Hala ere, merkatuan badago automata bat hau baimentzen duena, “SIMATIC ET200SP” produktu sortaren barruan, “Open Controller” izenekoa, 3. irudia bezalakoa.

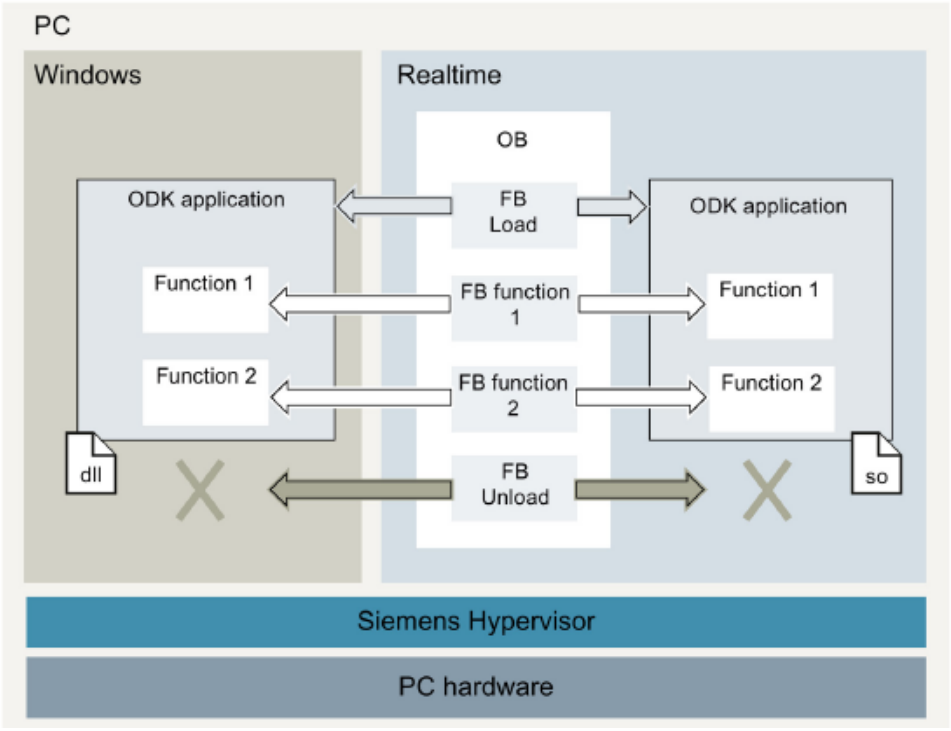


3. Irudia: ET200SP Open Controller

Automata hau, edozein etxeko ordenagailu bezala, sistema eraginkor bat du. Proiektu honetan erabiliko dena Windows 7 sistema eragilea du. Honetaz aparte, automata CPU bat du, automata tipiko baten lana beteko duena, irteera eta sarrera batzuen bitartez eta TIA Portal programaren bidez programatu daitekeena. Sistema eragilea eta CPU-a elkarrekin erlazionatuta daude, hau da, ez dira bi sistema independente. Hau guztia dela eta, Java aplikazio baten exekuzioa eta automatarekin komunikazioa emateko baldintzak betetzen dira.

5.3. Open Data Kit

Azkenik, aurreko automata eta bere sistema eragilea komunikatzeko “Open Data Kit”, edo oraindik aurrera “ODK”, izeneko software baliabidea erabili beharra dago. ET200SP-a C++ edo C lengoaian programatutako aplikazio batera konekta daiteke software honen bidez. Software pakete honek Visual Studio edo Eclipse garapen programei, TIA Portal-ekin bateragarriak izateko liburutegiak eta baliabideak eskaintzen ditu eta funtzio pertsonalizatuak programatzeko aukera ematen du, gero funtzio horiek TIA Portal programan kargatzeko. Bi posibilitate baimentzen ditu: alde batetik, denbora errealeko programak sortzea Eclipse programaren bitartez, hau da, automata programa exekutatzen den bitartean, funtzio (FB) konkretu bat exekutatzean, Eclipserekin programatutako funtzio bati deitzeko aukera izatea, adibidez funtzio matematiko bat; eta beste aldetik, Windows-en Visual Studio-ren bitartez garatutako programa bat exekutatzea, Windows-en abantailak eta liburutegiak aprobetxatuz, adibidez zerbitzari bat egiteko. ODK-ren dokumentaziotik (1) ateratako 4. irudian bi aukera hauek ikusi daitezke era grafiko batean.



4. Irudia: ODK-ren funtzionamendua

Proiektu hau garatzeko, Visual Studiorekin C++ motako aplikazio bat programatuko da, Java aplikazio batekin komunikatzeko gaitasuna duena, eta JADE agentera informazio bidalketa bidirekzionala baimentzen duena.

6. ALTERNATIBEN ANALISIA

Proiektu honen helburua lortzeko software eta hardware alternatiba ezberdinak aztertu egin dira. Horretarako, “espezifikazio teknikoak” atalean aipaturiko baldintza teknikoak bildu egin dira eta horiekin, proiekturako egokiena aukeratu da, jarraiko alternatiben analisian azaltzen den moduan.

Alternatiben analisia egiteko 2 aspektu hartuko dira kontuan: Alde batetik ET200SP-aren modeloa eta bestetik IPC komunikazio sistema. Azterketa egiteko, “Batuketa Haztatua” erabiliko da.

Batuketa Haztatua: faktore bakoitzari, kalifikazio bana emango zaio. Proiektuen baldintzak gehien betetzen dituen aukerari kalifikazio altuena emango zaio. Gero kalifikazioak dagokion inportantzia mailarekin biderkatuko dira.

$$BH = \sum_{1}^{n} p_i \cdot x_i \qquad [1]$$

6.1. ET200SP Open Controller modeloa

ET200SP Open Controller sistemak, esan bezala, seinaleen prozesatzea eta informazioaren bidalketa egiteko kapaza izan behar da. Baldintza horiek betetzen dituzten hainbat modelo existitzen dira merkatuan SIMATIC ET200SP Open Controller modeloen barruan.

4.1.1. Alternatiben deskribapena

3 modelo bereizten dira:

4.1.1.1. SIMATIC ET200SP Open Controller CPU 1515SP 32bit.

Modelorik basikoena, prozesadore nahiko kaskarrarekin baina prezio baxuarekin. (2)

EZAUGARRIAK:

- 32bit-eko prozesadorearekin.
- 4gb RAM
- PROFINET eta Ethernet-ekin baterakorra.

- Prezioa: 2119,08€

6.1.1.2. SIMATIC ET200SP Open Controller CPU 1515SP 64bit.

Erdibideko modeloa, prozesadore onarekin. (3)

EZAUGARRIAK:

- 64bit-eko prozesadorearekin
- 4gb RAM
- PROFINET eta Ethernet-ekin baterakorra
- Prezioa: 2837,15€

6.1.1.3. SIMATIC ET200SP Open Controller CPU 1515SP 64bit + HMI

Modelorik garatuena, prozesadore onarekin eta WinCC-rekin bateragarria. (4)

EZAUGARRIAK:

- 64bit-eko prozesadorearekin.
- 4gb RAM
- PROFINET eta Ethernet-ekin baterakorra.
- "WinCC Advanced" HMI sistema dakar.
- Prezioa: 3903,28€

6.1.2. Alternatiben analisia

Automata guztietatik modelo egokiena zein den aukeratzeko 3 faktore kontuan hartu dira: prozesadorearen kapazitatea, software paketea eta prezioa.

Hurrengo lerroetan, I. taula osatzeko erabili diren kriterioak azalduko dira; hau da, pisu bakoitzari eman zaion balioa eta zergatia azalduko da:

- Prozesadorearen kapazitateari % 45ko pisua eman zaio, JADE estrukturarekin eginiko Java programaz aparte C++ programa exekutatu behar da eta. Horretaz aparte, automatan bertan programazioa egiten bada denbora aurreztu egingo da eta ondorioz, kostuak txikitu, eta horretarako prozesadore on bat egotea nahitaezkoa da.

- Software paketeari % 35eko pisua eman zaio. Izan ere, modelo aurreratuena HMI dakar, eta ezinbestekoa ez bada ere, sarrerak eta irteerak simulatzeko erabili ahal da, proiektuaren garapena erraztuz.
- Prezioari % 20ko pisua eman zaio. Proiektu honetan faktorerik garrantzitsuena ez bada ere, edozein proiektutan kontuan hartzekoa da.

Prozesadorerik garatuena 64bit-ekoa da, baina 2 daudenez biei kalifikazio bera emango zaie, 8. Prozesadorerik kaskarrena 32bit-ekoa denez 5 bat emango zaio.

Software paketerik handiena 64bit+HMI duen modeloa da. Beraz, honi 9 bat emango zaio. beste modeloei, WinCC Advanced ez dakartenez, 6 bat emango zaie.

Azkenik, preziorik altuena 64bit+HMI duena du, eta beraz 5 bat emango zaio. 64bit-eko modeloa erdibidekoa denez 7 bat emango zaio eta 32bit-eko modeloari merkeena denez 9 bat.

I. Taula: Modeloaren selekzioa

	32 bit	64bi	64bit+HMI
Prozesadorea % 45	5	8	8
Software paketea % 35	6	6	9
Prezioa % 20	9	7	5
	6,15	7,1	7,75

Erabiliko den modeloa SIMATIC ET200SP Open Controller CPU 1515SP 64bit + HMI izango dela aukeratu da, notarik altuena daukalako.

4.2. IPC komunikazio sistema

ET200SP Open Controller-aren barruan, esan bezala, bi aplikazioren exekuzioa egin behar da. Alde batetik, Java programa eta bestetik, Visual Studiorekin programatuko den C++ aplikazioa. Bi aplikazio hauen artean komunikazio bidirekzional bat ezarri behar da eta horretarako komunikazio metodo bat aukeratu da.

4.2.1. Alternatiben deskribapena

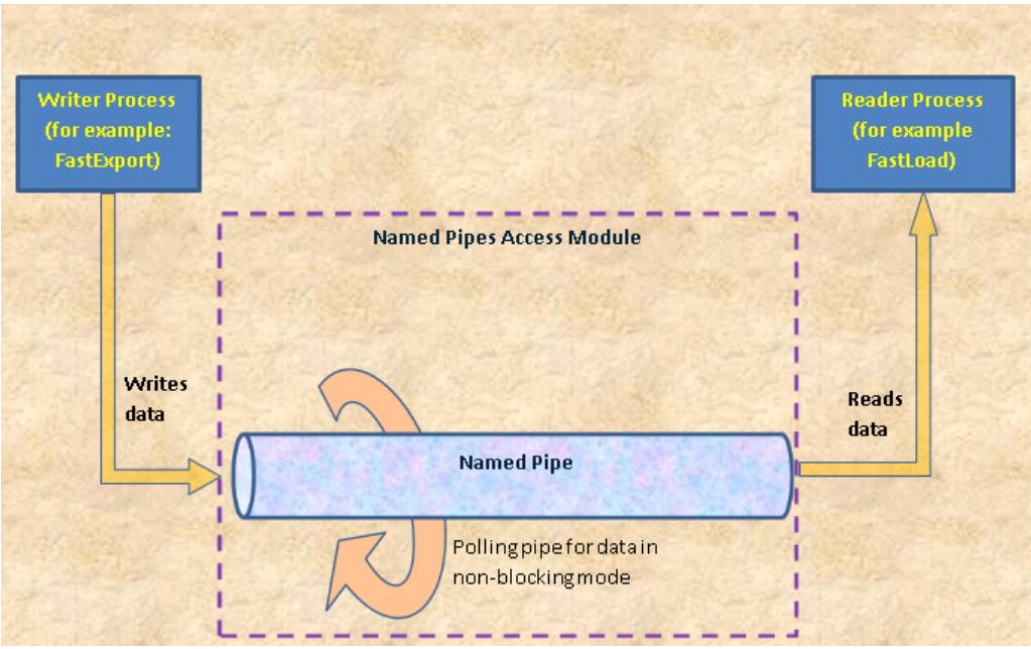
4 metodo nagusi kontuan hartu dira.

4.2.1.1. Pipe izendatuak

5. Irudian ikusten den bezala, FIFO (First In First Out) motako kanal baten sorkuntzan oinarritzen da sistema hau. Pipe-a erregistro bat bezala erabili daiteke, non, prozesu batek irakurle bezala zabal dezake Pipe-a eta beste batek idazle bezala. Honek simplex metodoa baimentzen du bakarrik, eta beraz bigarren Pipe baten sorkuntza beharrezkoa da Full duplex komunikazioa erabiltzeko.

EZAUGARRIAK:

- Bi Pipe-n beharra dago komunikazioa bidirekzionala izateko
- Kontzeptualki erraza, programatzeko zailagoa
- Fidagarritasun makala
- FIFO estruktura



5. Irudia Pipe izendatuen adibidea

4.2.1.2. Memoria partekatua

Sistema beran dauden bi aplikaziok irakurketa eta idazketa prozesuak egin ditzakete memoria zati bat partekatuz. Idazketa eta irakurketa prozesuak independenteak dira, ondorioz, programazioan idazketa-irakurketa sinkronismo sistema bat izatera behartzen du.

EZAUGARRIAK:

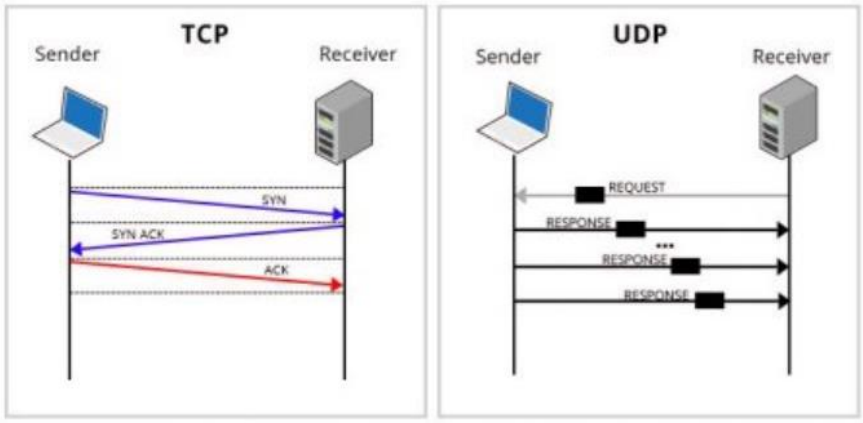
- Idazketa eta irakurketa sinkronismo barik. Aplikazio batek irakurketa agindua jasotzean ez du itxarongo beste aplikazioak informazioa idatzi arte
- Sistema guztietatik azkarrena
- Datuak memoria zati partekatuan geratzen dira berriro idatzi arte

4.2.1.3. UDP Socket-a

Socket-a bezero-zerbitzari motako komunikazioa baimentzen du. Lehenik zerbitzaria hasi eta bezeroen bila hasiko da. Behin bezero bat zerbitzarira konektatzean 2 bide hartu daitezke: edo UDP motako protokoloa ahalbidetzea edo TCP motakoa (6. irudia). UDP motako protokoloan informazioa bidali egiten da eta jasotzaileari ondo heldu zaion ez da konprobatzen. Honek, komunikazio abiadura handiagoak ahalbidetzen du.

EZAUGARRIAK:

- Bezero-zerbitzari sistema bat denez, konexioa baimendu behar da bidalketa bakoitzean
- Programatzeko sinplea
- Aplikazioak sare batera konektatuta egon daitezke, ez dute ekipo beran egon beharrik
- Ez da oso fidagarria
- Nahiko azkarra



6. Irudia: TCP vs UDP

4.2.1.4. TCP Socket-a

TCP protokoloa erabiltzean, jasotzaileak mezu bat jasotzean, mezua ondo heldu zaion erantzungo du. UDP baino motelagoa da.

EZAUGARRIAK:

- Bezero-zerbitzari sistema bat denez, konexioa baimendu behar da bidalketa bakoitzean
- Programatzeko sinplea
- Aplikazioak sare batera konektatuta egon daitezke, ez dute ekipo beran egon beharrik
- UDP baino fidagarriagoa

4.2.2. Alternatiben analisia

Komunikazio sistema guztietatik egokiena zein den aukeratzeko 3 faktore kontuan hartu egin dira: fidagarritasuna, programazio erraztasuna eta abiadura

Hurrengo lerroetan, II. taula osatzeko erabili diren kriterioak azalduko dira; hau da, pisu bakoitzari eman zaion balioa eta zergatia azalduko da:

- Fidagarritasunari %45ko pisua eman zaio, izan ere, informazio bidalketan erroreak egongo balira, kasu hoberenean, berriro bidaltzeko eskaria programatu beharko litzateke, eta kasu txarrenean sistema bertan behera gera liteke.
- Programazio erraztasunari %35ko pisua eman zaio, proiektuaren garapenaren azkartasunarekin erlazionatuta dagoelako, eta azken hau kostuarekin.
- Komunikazioaren azkartasunari %20ko pisua eman zaio, izan ere proiektu honetan komunikazio abiadura oso garrantzitsua ez bada ere, kontuan hartzekoa da.

Komunikazioa fidagarriena TCP motako socket-a da eta 9 bat esleitu zaio. Erdibidean memoria partekatuak eta pipe izendatuak egongo dira, 7 eta 6 batekin, hurrenez hurren. Azkenik, UDP Socket-a egongo da, 5 batekin, komunikazio mota fidagaitzena da eta.

Programazio erraztasunean aldiz, bi socket motako komunikazioak egongo dira, 8 batekin, biak erraztasun beraz programatu egin daitezkeelako. Memoria partekatuak erdibideko postua izango du, ez delako ez erraza ez zaila, 7 batekin. Pipe izendatuak 4 batekin ponderatu dira, izan ere, guztietatik programatzeko zailena dela uste izan da.

Azkenik, komunikazio metodorik azkarrena memoria partekatua denez 9 bat eman zaio. UDP socket-a eta pipe izendatuak hurrengo postua jasoko dute, komunikazio abiadura nahiko azkarra dute eta, 8 batekin. Azkenik, TCP motako komunikazioari, erlatiboki motelena denez, 7 bat eman zaio.

II. Taula: Komunikazio sistemaren selekzioa

	Pipe izendatuak	Memoria partekatua	UDP Socket-a	TCP Socket-a
Fidagarritasuna % 45	6	7	5	9
Programazio erraztasuna % 35	4	7	8	8
Komunikazio azkartasuna % 20	8	9	9	7
	5,7	7,4	6,85	8,25

TCP Socket-a notarik altuena duena da, beraz, komunikazioa egiteko erabiliko dena izango da.

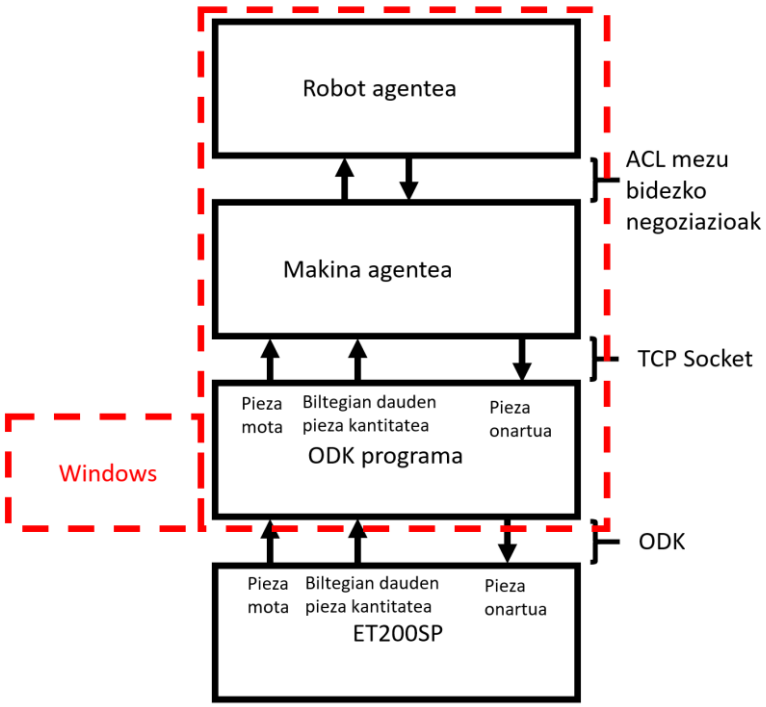
5. DISEINU ETA INPLEMENTAZIOA

5.1. Goi mailako diseinua

Diseinatuko den programa hainbat zatitan banatuko da, 7. irudian adierazita agertzen den bezala. Hasteko, automataren CPU-aren programa izango dugu. Hemen TIA Portalekin automataren eginkizunak programatu daitezke. Hurrengo urratsean Windows-en, ODK-ren eta Visual Studio-ren bitartez, C++ motako aplikazio bat egingo da. Aplikazio honen eginkizuna automatak dituen datuak hurrengo urratsean dagoen makina agenteari bidaltzea da.

Eskeman gora eginez, JADE motako makina agentea egongo da. Agente honen eginkizuna ODK aplikaziotik informazioa jasotzea eta robot agentearekin negoziatzea da. Makina agenteak pieza bat jaso egingo denean makinari abisu bat bidaliko dio honek pieza hartzeko prestakuntza egiteko eta bitartean beste piezarik ez onartzeko.

Azkenik, robotaren agentea egongo da. Proiektu honetan ET200SP-aren sistema eragile beran sartu egin da, robotaren eginkizuna “simulatuz”. Programa honen betekizuna makinen informazioa kontsultatzea eta datu horien arabera erabaki bat hartzea da.



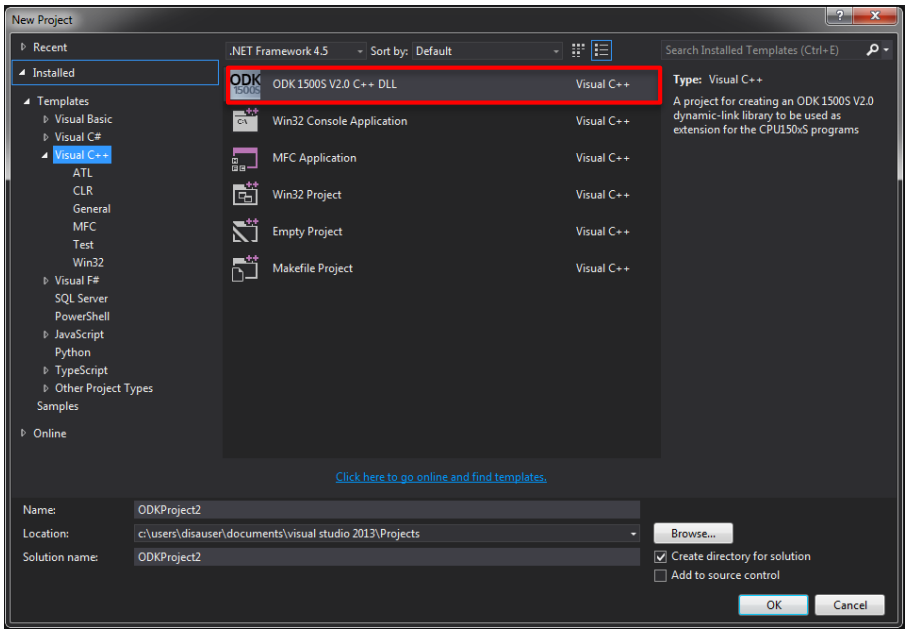
7. Irudia: Proiektuaren eskema

5.2. Programen konfigurazioa eta kontzeptuen azalpena

Programazio kodigoak azaldu aurretik, erabiliko diren programak nola konfiguratu behar diren eta hainbat kontzeptu azaldu beharra dago.

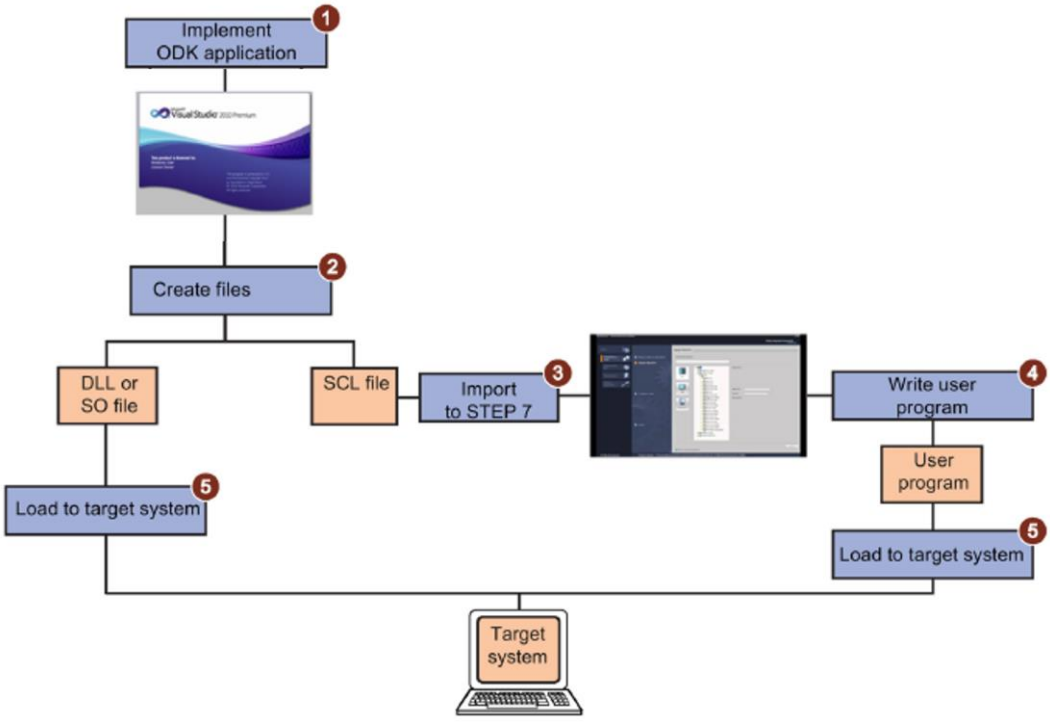
5.2.1. Visual Studio eta ODK

Visual Studio erabiltzeko eta automatarekin bateragarria izateko, lehenik eta behin ODK software paketea instalatu behar da Visual Studio instalatuta dagoen ordenagailuan. V2.0 bertsioa erabiliko da. Behin instalatuta Visual Studio zabaldu eta proiektu berri bat sortu behar da ODK V2.0 proiektu mota aukeratuz 8. irudian adierazten den bezala. Hori egin eta gero, proiektu berri bat zabaldu egingo da. Zabalduko den proiektua adibidezko proiektu bat da baina nahi den proiektua egiteko editatu egin daiteke.



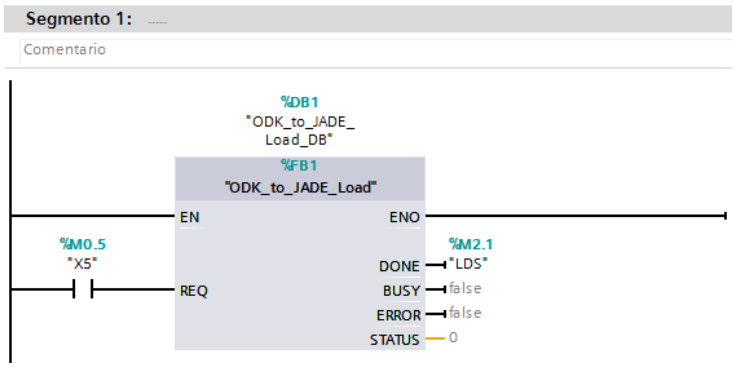
8. Irudia: ODK proiektu baten sorkuntza

ODK programa bat konpilatzean 2 artxibo sortuko dira: bat .scl luzapena du eta bestea .dll luzapena.

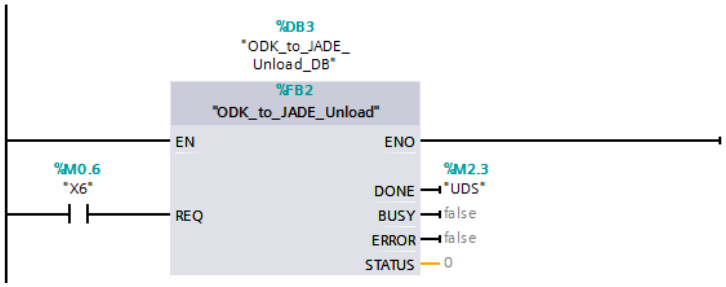


9. Irudia: ODK garapen pausuak.

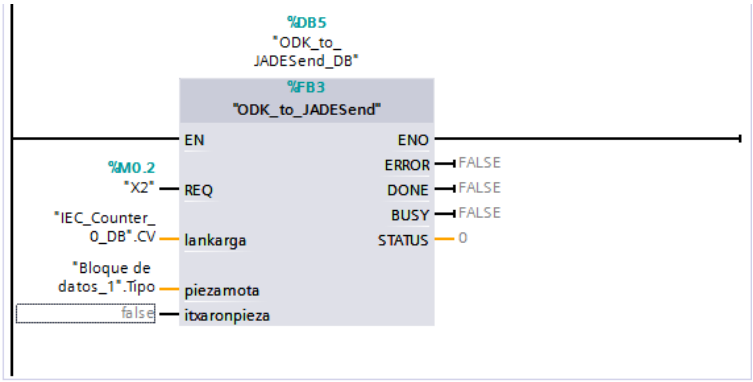
9. irudian ikusten den bezala, lehen artxiboa TIA Portalen kargatzeko pentsatuta dago. Artxibo honek ODK programan definitu egingo diren funtzio pertsonalizatuak TIA Portalen kargatzea baimentzen du. Programatutako funtzio bakoitza FB bat bezala agertuko da TIA Portal-ean eta FB bakoitzari DB bat esleituko zaio automatikoki FB-ak programan integratzean. Funtzio pertsonalizatuez aparte, funtzio generiko batzuk sortuko dira baita ere, Load eta UnLoad izenekoak. 10, 11 eta 12. irudietan Load, UnLoad eta funtzio pertsonalizatu bat ageri dira.



10. Irudia: Load funtzioa



11. Irudia: Unload funtzioa



12. Irudia: Funtzio pertsonalizatua

12. irudian dagoen FB3 funtzio pertsonalizatua exekutatzeko, aldez aurretik, Load (FB1 irudian) funtzioaren exekuzioa egin behar da. Funtzio honek automatan aplikazioa kargatuko du. Funtzioak exekutatzeko REQ sarrera bita erabiltzen da, non, 1 logiko bat detektatuz gero dagokion funtzioa exekutatuko da. Behin Load funtzioa exekutatuta, FB3 funtzioa exekutatu daiteke segidan. 12. irudian ikusten den bezala FB3 funtzioa 3 datu mota ditu, ODK aplikazioa eta automataren artean komunikatuko diren datuak, hots. Datu hauek ODK aplikazioan definitu behar dira .odk artxiboan. Behin aplikazioaren exekuzioa bukatu nahi dugunean UnLoad (FB2 irudian) funtzioa exekutatu behar da. Funtzio honek aplikazioaren deskarga egingo du.

Funtzioren bat txarto exekutatuko balitz ERROR irteeran 1 logiko bat agertuko da. STATUS irteeraren arabera errorea zergatik gertatu den jakin daiteke. 13 , 14 eta 15. irudietan ODK-ren dokumentaziotik ateratako diagnostiko taula hiru ageri dira.

DONE	BUSY	ERROR	STATUS	Meaning
0	0	0	0x7000 =28672	No active unloading
0	1	0	0x7001 =28673	Unloading in progress, the first call
0	1	0	0x7002 =28674	Unloading in progress, ongoing call
1	0	0	0x0000 =0	Unloading was carried out successfully
0	0	1	0x80A4 =-32604	CPU function library could not be unloaded for the following reasons: • Windows is not available Start the ODK service manually or restart Windows.
			0x80C2 =-32574	CPU function library could not be unloaded. There are currently not enough resources available from Windows. Reload the CPU function library after a few seconds.
			0x80C3 =-32573	CPU function library could not be unloaded. The CPU currently does not have enough resources. Reload the CPU function library after a few seconds.
			0x8090 =-32624	An exception occurred during the unloading of the CPU function library. The CPU function library has been unloaded nevertheless.
			0x8096 =-32618	CPU function library could not be unloaded because the CPU function library was not loaded or unloading is not yet finished.
			0x809B =-32613	CPU 1500 V2.0 and later: The CPU function library could be unloaded and returns an invalid value (the values 0x0000 and 0xF000 - 0xFFFF are allowed)
			0xF000 – 0xFFFF =-4096 – -1	CPU 1500 V2.0 and later: CPU function library could be unloaded. An error occurred in the CPU function library during the execution of the "OnUnload()" function.

13. Irudia: Unload funtzioaren diagnostiko taula.

DONE	BUSY	ERROR	STATUS	Meaning
0	0	0	0x7000 =28672	No active loading
0	1	0	0x7001 =28673	Loading in progress, first call
0	1	0	0x7002 =28674	Loading in progress, ongoing call
1	0	0	0x7100 =28928	CPU 1500 V2.0 and later: ODK application has already been loaded.
1	0	0	0x0000 =0	Loading was performed successfully.
0	0	1	0x80A4 =-32604	ODK application could not be loaded. Start the ODK service manually or restart Windows.
			0x80C2 =-32574	ODK application could not be loaded. There is currently not enough memory available at the Windows end. Load the ODK application again after a few seconds.
			0x80C3 =-32573	ODK application could not be loaded. The CPU currently does not have enough memory. Load the ODK application again after a few seconds.
			0x8090 =-32624	ODK application could not be loaded. An exception occurred during execution of the "OnLoad()" function.
			0x8092 =-32622	ODK application could not be loaded because the library name is invalid.
			0x8093 =-32621	ODK application could not be loaded because the ODK application could not be found. Check the file name and path of the file.
			0x8094 =-32620	ODK application could not be loaded. The ODK application was created for the Windows user context, but no user is logged on.
			0x8095 =-32619	ODK application could not be loaded for the following reasons: • The DLL file is not an ODK application • An attempt has been made to load a 64-bit application into a 32-bit system • Dependencies on other Windows DLL files could not be resolved. – Check whether the release build of the ODK application is being used. – Check whether the "Visual C++ Redistributables" are installed for the Visual Studio version you are using. • The CPU does not support the utilized ODK version.
			0x8096 =-32618	The ODK application could not be loaded because the internal identification is already being used by another loaded ODK application.
			0x8097 =-32617	CPU 1500 V1.8 and earlier: ODK application has already been loaded.
			0x8098 =-32616	ODK application could not be loaded because the ODK application is currently being unloaded.
			0x809B =-32613	CPU 1500 V2.0 and later: The ODK application could not be loaded and returns an invalid value (the values 0x0000 and 0xF000 - 0xFFFF are permitted)
			0xF000 – 0xFFFF =-4096 – -1	CPU 1500 V2.0 and later: ODK application could not be loaded. An error occurred during execution of the "OnLoad()" function.

14. Irudia: Load funtzioen diagnostiko taula

DONE	BUSY	ERROR	STATUS	Meaning
0	0	0	0x7000 =28672	No active process
0	1	0	0x7001 =28673	First call (asynchronous)
0	1	0	0x7002 =28674	Continuous call (asynchronous)
1	0	0	0x0000 – 0x6FFF =0 – 28671	Function has been executed and returns a value between 0x0000 and 0x6FFF. (ODK_SUCCESS = 0x0000)
0	0	1	0x80A4 =-32604	CPU function library could not be executed for the following reasons: • The "*<STEP7Prefix>*_Unload" instruction was executed during a function execution. The function execution was aborted at the CPU end. Windows terminates the execution of the function normally. No return value is sent to the CPU. Wait until the "*<STEP7Prefix>*_Unload" instruction has ended. Then load the CPU function library again. • Windows is not available • ODK service is not running Start the ODK service manually or restart Windows.
			0x80C2 =-32574	CPU function library could not be executed. There are currently not enough resources available from Windows. Execute the CPU function library again after a few seconds.
			0x80C3 =-32573	CPU function library could not be executed. The CPU currently does not have enough resources. Execute the CPU function library again after a few seconds.
			0x8090 =-32624	CPU function library could not be executed. An error occurred during execution.
			0x8091 =-32623	CPU function library could not be executed. A "STOP" occurred during the function call.
			0x8096 =-32618	CPU function library could not be executed because the CPU function library was not loaded or unloading is not yet finished.
			0x8098 =-32616	CPU function library could not be executed because the function is not supported.
			0x8099 =-32615	CPU function library could not be executed because the maximum amount of input data (1 MB) was exceeded (declarations with "In" and "InOut")
			0x809A =-32614	CPU function library could not be executed because the maximum amount of output data (1 MB) was exceeded (declarations with "Out" and "InOut")
			0x809B =-32613	The function returns an invalid value (a value between 0x0000 and 0x6FFF; 0xF000 and 0xFFFF is permitted)
			0x809C =-32612	Function uses an invalid data type: • IN_DATA • INOUT_DATA • OUT_DATA
			0xF000 – 0xFFFF =-4096 – -1	CPU 1500 V2.0 and later: The function could not be executed and returns a value between 0xF000 and 0xFFFF. (ODK_USER_ERROR_BASE = 0xF000)

15. Irudia: ODK funtzioaren diagnostiko taula

Bigarren artxiboa, .dll luzapena duena, Open Controller-aren disko gogorraren "%ProgramData%\Siemens\Automation\ODK1500S\" helbidean kopiatu egin behar da.

Artxibo hau Visual Studio-n programatutako aplikazioa darama eta automataren programatik exekutatzen da.

Visual Studio-n ODK proiektu pertsonalizatu bat egiten hasteko bi artxibo kontuan hartzeko dira:

- <zure_proiektuaren_izena>.odk artxiboan funtzio pertsonalizatu guztien definizioa egin behar da. Adibidez, 12. irudian adierazitako FB-aren datuak definitzeko "*ODK_RESULT komunikatu([IN] ODK_INT16 lankarga, [IN] ODK_S7STRING piezamota, [INOUT] ODK_BOOL itxaronpieza*);" funtzioa idatzi behar izan da artxibo honetan. Datuek bi dimentsio dituztela esan daiteke:
    - Datu mota tamainaren arabera (ODK_INT16 adibidez). .odk artxiboan zerrendatuta daude.
    - Datuaren direkzioaren arabera. Sarrerakoak (IN), irteerakoak (OUT) edo bidirekzionalak (INOUT) izan daitezke. Ez da beharrezkoa funtzioan mota bakoitzeko datu bat definitzea derrigorrez, hau da, bi OUT motako datuak definitu daiteke edo IN bat eta INOUT bat baina datuak definitzerakoan IN -> OUT -> INOUT ordenean egin behar da derrigorrez.
- <zure_proiektuaren_izena>.cpp artxiboan aplikazioaren programa nagusia dago. Bere barnean hainbat funtzio ditu:
    - OnLoad(). Automataren programan Load funtzioa egitean exekutatu nahi den kode zatia.
    - OnUnload(). Automataren programan UnLoad funtzioa egitean exekutatu nahi den kode zatia.
    - OnRun(). Automata run egoerara pasatzean exekutatu nahi den kode zatia, aurretik aplikazioa kargatu egin bada Load funtzioaren bitartez.
    - OnStop(). Automata stop egoerara pasatzean exekutatu nahi den kode zatia, aurretik aplikazioa kargatu egin bada Load funtzioaren bitartez.
    - Programatu diren gainontzeko funtzio pertsonalizatu guztiak .odk artxiboan deklaratu egin denarekin kointziditu behar dute. Gainera IN motakoetan "const" gehitu behar zaio datu mota aurretik (const *ODK_INT16& lankarga*), izan ere, IN motakoak automatatik irakurri

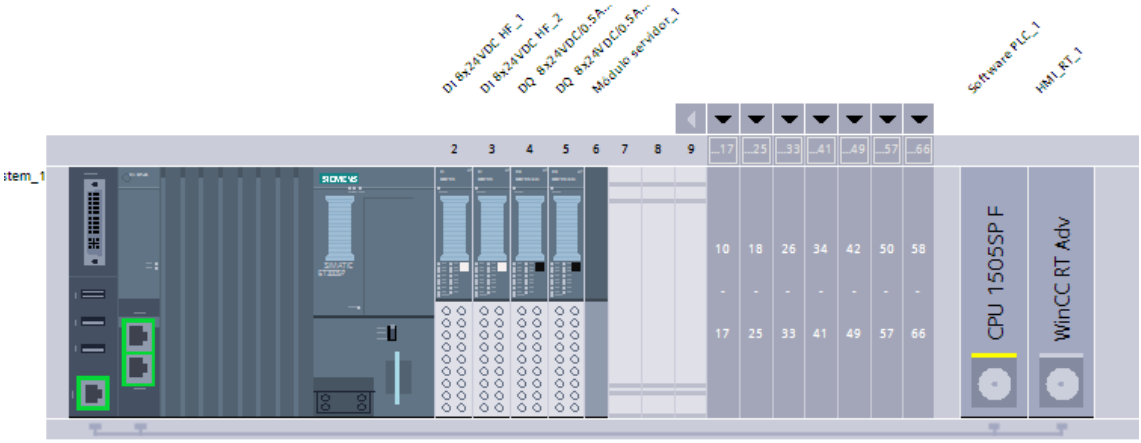
daitezke bakarrik eta bere gainean ez idazteko babesturik egon behar dira.

Programatzen hasi aurretik soluzio plataforman “Win32” jarrita dagoela baieztatu behar da.

5.2.2. TIA Portal

ET200SP-a duen X2 sarreran Ethernet kable baten konektatuko da eta beste muturrean TIA Portal instalatuta dagoen ordenagailua egongo da. Proiektu honetan erabili den sare txartela Intel ® PRO 100 MT izan da. Automataren sisteman “Software PLC_1” aplikazioa exekutatu egingo da eta interfaze bat zabalduko da, automataren informazioa eta bere egoera adieraziz (Run, Stop…).

Ordenagailuan TIA Portal zabaldu eta gero dispositibo berri bat gehituko dugu: “Sistemas PC / SIMATIC S7 Open Controller / ET 200SP Open Controller / CPU 1515SP PC + HMI / 6ES7 677-2AAxx-0xx0”. Dispositibo honen bertsioa V2.1 izan behar da. Gehitu egingo den dispositiboaren barruan dagoen CPU-a CPU 1505SP F dela baieztatu behar da. Honen bertsioa V2.0 izan behar da eta daukan erreferentzia 6ES7 672-5SC01-0YA0 da. Proiektu honen kasuan 2 sarrera eta 2 irteera periferiko erabiliko dira. Hardware konfigurazioan libre dauden moduluetan Input moduluak (“DI / DI 8X24 VDC HF / 6ES7 131-6BF00-0CA0” V1.0 bertsioan) eta output moduluak (“DQ / DQ 8x24VDC/0.5A HF / 6ES7 132-6BF00-0CA0” V2.0 bertsioan) gehituko dira eta ““General/Grupo de potencial” atalean potentzial talde berri bat baimendu behar da (“BaseUnit clara”) sarrera/irteera modulu guztietan. Serbitzari modulua gehituko da baita ere, 6ES7 193-6PA00- 0AA0 erreferentzia zenbakiarekin eta V1.1 bertsioarekin. Erabili nahi ez izatekotan WinCC-a ezabatu daiteke konfiguraziotik. Azkenengo konfigurazioa 16. irudian dagoena bezalakoa izan behar da.

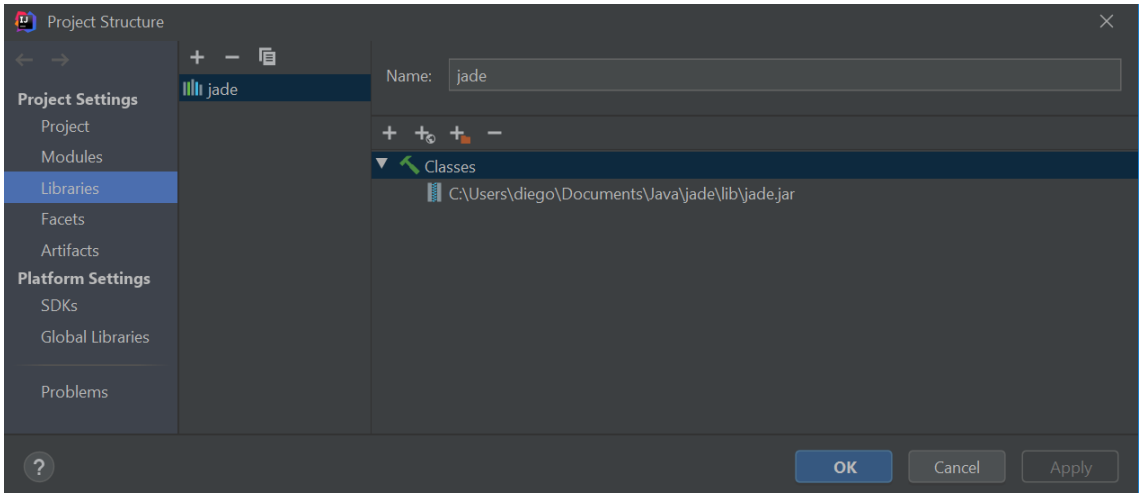


16. Irudia: ET200SP Open Controller-aren TIA Portal konfigurazioa

PC-System_1 hautatu eta “Protección” atalean “Acceso completo incl. Seguridad positiva (Sin protección)” aukera hautatu behar da. X2 Ethernet konektorea aukeratu hardware konfigurazioan eta sartu automatari dagokion TCP/IPv4 helbidea.

5.2.3. IteliJ IDEA

JADE motako Java agenteak programatzen hasteko, JADE liburutegiak deskargatu egin dira (4) eta “File/Project Structure/Libraries” helbidean “jade.jar” liburutegia gehitu da, 17. irudian adierazten den bezala. Ondoren, nahi diren agente guztiak “.java” motako artxibo anitzetan bana daitezke “src” karpetaren barnean. Konpilazioa egitean guztiak batera konpilatuko dira. Proiektu honen kasuan bi agente bakarrik egongo dira, robota eta makina.



17. Irudia: JADE liburutegiak gehitzen.

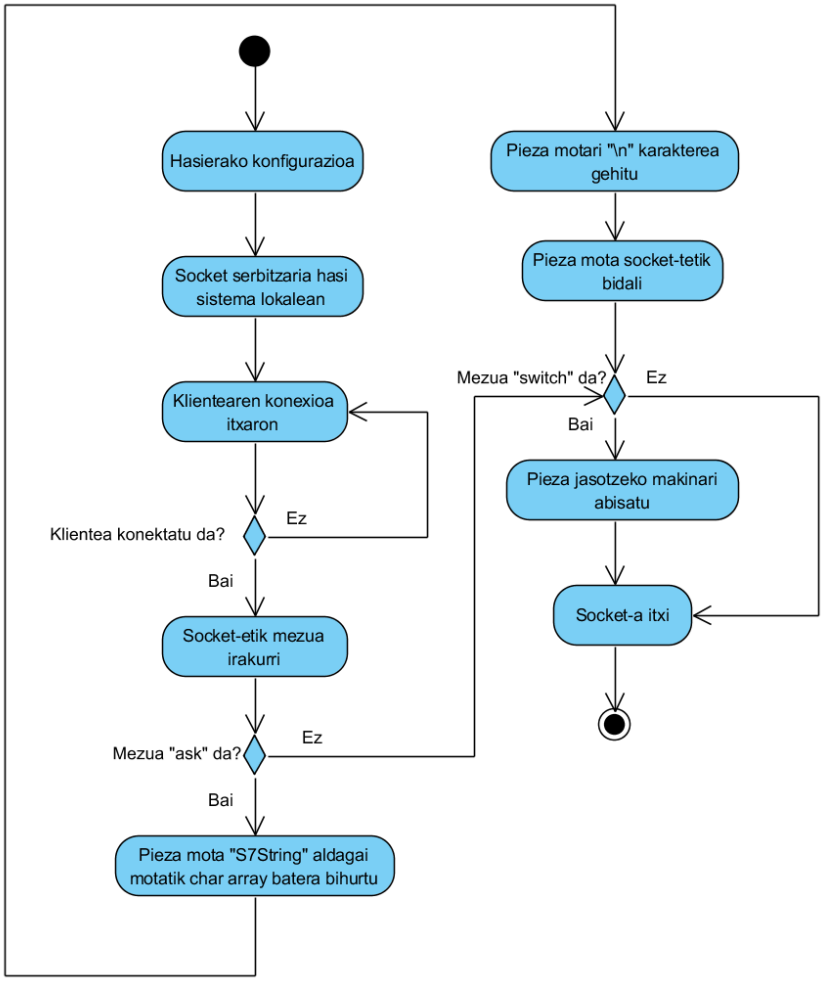
5.3. Behe mailako diseinua

Behe mailako diseinuan programazioa eta exekuzioa egiteko egin behar dena azalduko da atal honetan.

5.3.1. Visual Studio

Egindako programa GitHub web orrialdean estekatu da, hurrengo web helbidean: “https://github.com/DiegoGarciaGutierrez/ODK.git”. 18. irudian programaren aktibitate diagrama gauzatu da, ahalik eta argien kodearen exekuzioa azaltzeko.

Behin zerbitzaria hasita, kliente bat konektatu arte itxaroten egongo da. Konexio bat lortzean, socket-etik irakurri egingo da mezu bat. Mezu hori “ask” bada, pieza motaren izena eta lan kargaren balioa bidali egingo ditu socket-etik. Kontuan hartzekoa da bakarrik bidali daitezkeela “char array” motako aldagaiak programatu den metodoarekin, eta ondorioz, bai lan karga (“int” motakoa) eta pieza mota (Step 7 String motakoa) bihurtu egin behar dira bateragarriak izateko bidalketa metodoarekin. “Switch” mezua jasotzekotan, makinak pieza bat jasotzeko prest egon behar dela esan nahi du. Ondorioz, pieza itxaroteko aldagai binarioari “TRUE” balorea esleituko zaio. Azkenik, Socket-a itxi egingo da, konexioa amaituz.



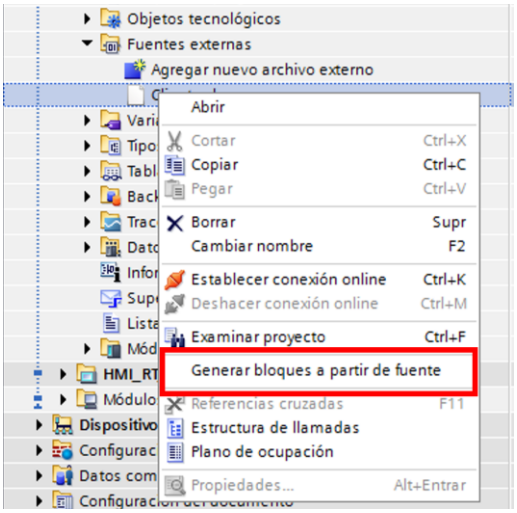
18. Irudia: ODK programaren aktibitate diagrama

Programa hau konpilatzean 2 artxibo sortuko dira:

- .dll motako bat, ET200SP-aren "<$Program data>\Siemens\Automation\ ODK1500S\" helbidean kopiatu behar dena.
- .scl motako bat, TIA Portal programan kargatu behar dena.

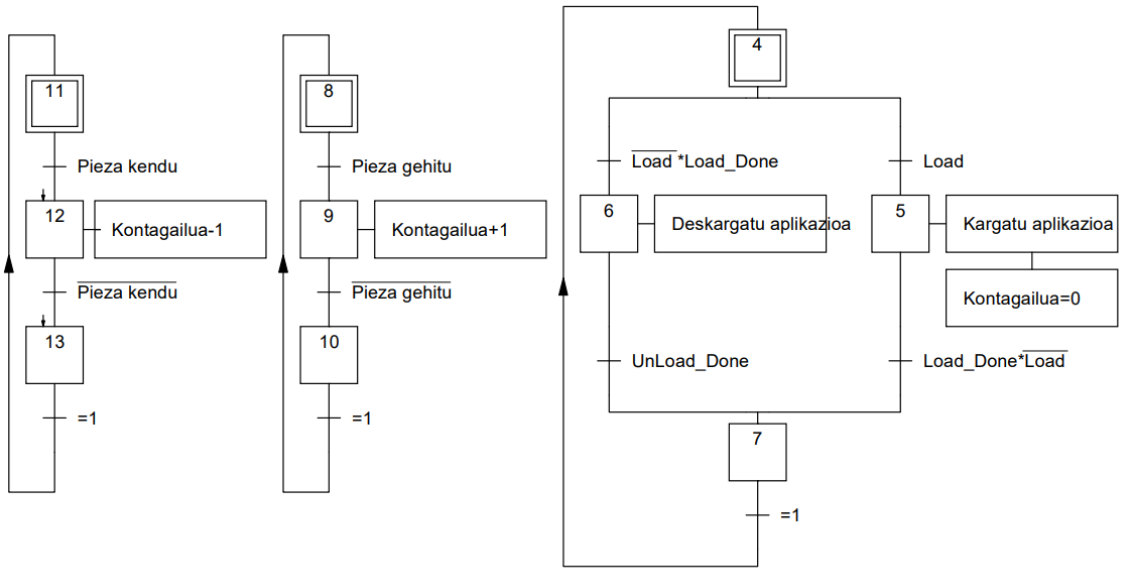
5.3.2. TIA Portal

TIA Portal programak aurrerago eraiki den programa exekutatzeko sortu egin den .scl artxiboa aukeratu da "Fuentes externas" aukeran, eta behin artxiboa gehituta dagoenean, iturritik funtzio blokeak sortu egin dira, 19. irudian ikusten den bezala.

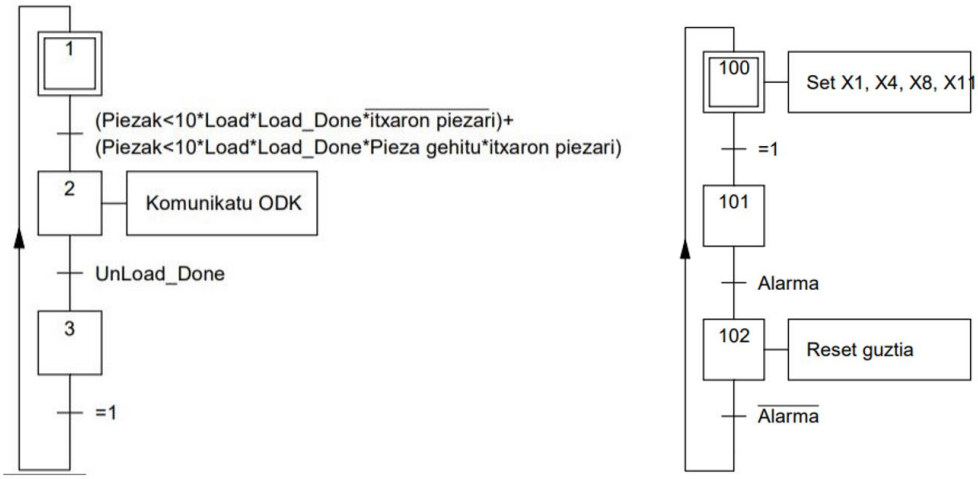


19. Irudia: .scl artxiboa gehitzen proiektuari

Behin dagokion funtzioak programari gehituta, funtzio bakoitza noiz exekutatu nahi diren eta programa nagusia definitu dira. 20 eta 21. irudietan irudian erabili den programazioa adierazten da grafcet batzuen bidez.



20. Irudia: TIA Portal-en programaren grafcet-a. Pieza kontagailua eta aplikazioaren karga/deskarga



21. Irudia: TIA Portal-en programaren grafcet-a. ODK funtzioaren exekuzioa eta alarma.

Alde batetik, makinaren pieza kontagailua dago. Independenteki bi sentsorek pieza bat sartzean pieza bat gehituko dute eta pieza bat irtetzean pieza bat kenduko dute. Biltegiaren kapazitate maximoa 10 unitate dela aukeratu da. Beste aldetik, ODK erabiliz programatutako funtzio pertsonalizatuak daude. “Komunikatu ODK” funtzioa exekutatzeko beharrezkoa da lehenik eta behin aplikazioa kargatzea, horretarako “Load” seinalea erabili da. Seinalea jasotzean ODK aplikazioaren karga egingo da eta pieza kantitatea 10 baino gutxiago izanez gero funtzioaren exekuzioa hasiko da. Makina agenteak pieza bat onartzen duenean “itxaron pieza” aldagaiari 1 logiko bat esleituko dio eta PLC-ak pieza jaso arte itxarongo du berriro “Komunikatu ODK” funtzioaren exekuzioa baimentzeko. “Load” seinalea 0-ra pasako balitz aplikazioaren deskarga hasiko litzateke. Gerta daiteke funtzioaren exekuzioaren bitartean deskarga eskatzea. Hau gertatzekotan funtzioaren exekuzioa bukatu arte itxarongo da.

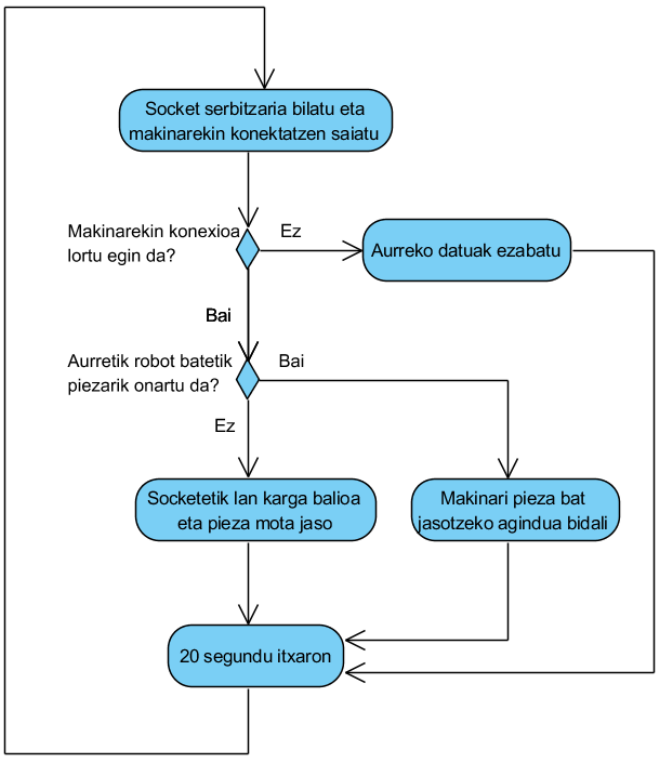
KOP programazioa “Eranskinak” atalean utziko da. Zati sekuentzialean eta konbinazionalean banatu egin da.

5.3.3. IntelliJ IDEA

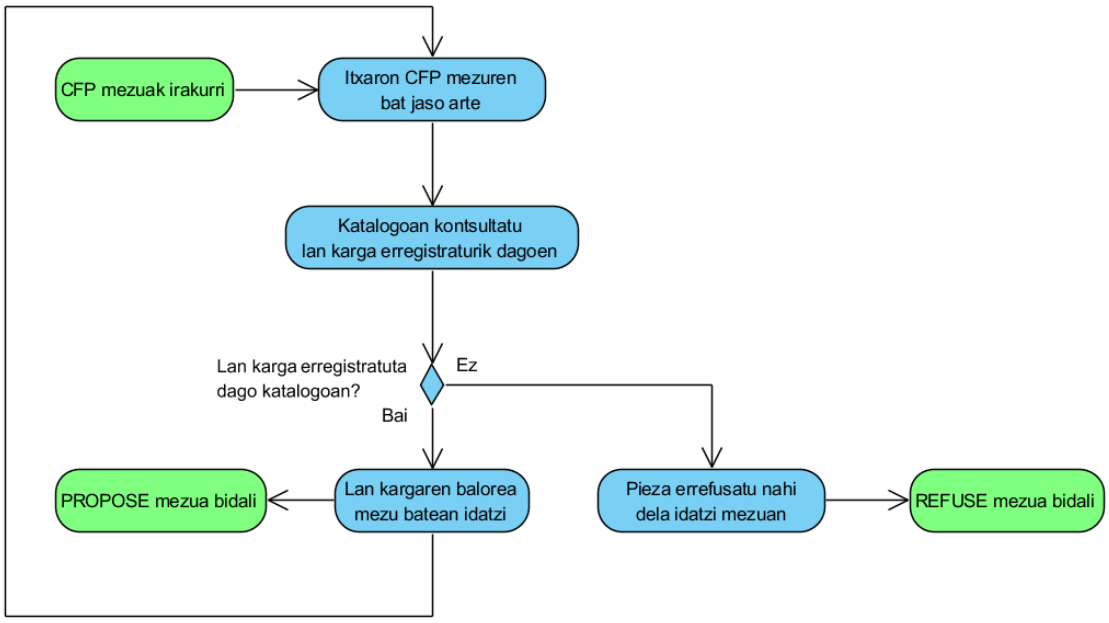
Makinaren eta robotaren agenteen programa bitan banatu behar da. Egindako programa GitHub web orrialdean estekatu da, hurrengo web helbidean: “https://github.com/DiegoGarciaGutierrez/JADE.git”. 22, 23 eta 24. irudietan programaren aktibitate diagrama gauzatu da, ahalik eta argien kodearen exekuzioa azaltzeko.

Makina agenteak 3 hari nagusi ditu:

- Lehen hariak, makinak dituen datuak eguneratzen egongo da, 20 segundu oro. Lehenik eta behin katalogoan dauden datuak ezabatuko dira. Makinarekin konexioa ondo ezartzen bada, lan kargaren balioa eta pieza mota jasoko dira makinatik, baina aurretik robot baten pieza onartu egin bada makinari pieza jasotzeko agindua emango zaio. Konexioa ezin izan bada ezarri ez da daturik idatziko katalogoan, hau da, robotek ez dute makina kontuan hartuko piezak banatzeko.
- Bigarren hariak "Call For Proposals" (CFP) motako mezuak jaso egin diren konprobatzen ibiliko da. Jasotzekotan, katalogoan konprobatuko da lan karga oraindik erregistratuta dagoela, izan ere, gerta daiteke beste robot batetik dagoeneko pieza bat onartu izatea edo egunerapen zikloarekin bat egitea. Katalogoan informazioa oraindik egotekotan robot agenteari "PROPOSE" motako mezu bat bidaliko zaio informazioarekin, eta ez egotekotan, "REFUSE" motako bat bidaliko zaio pieza ukatzeko.

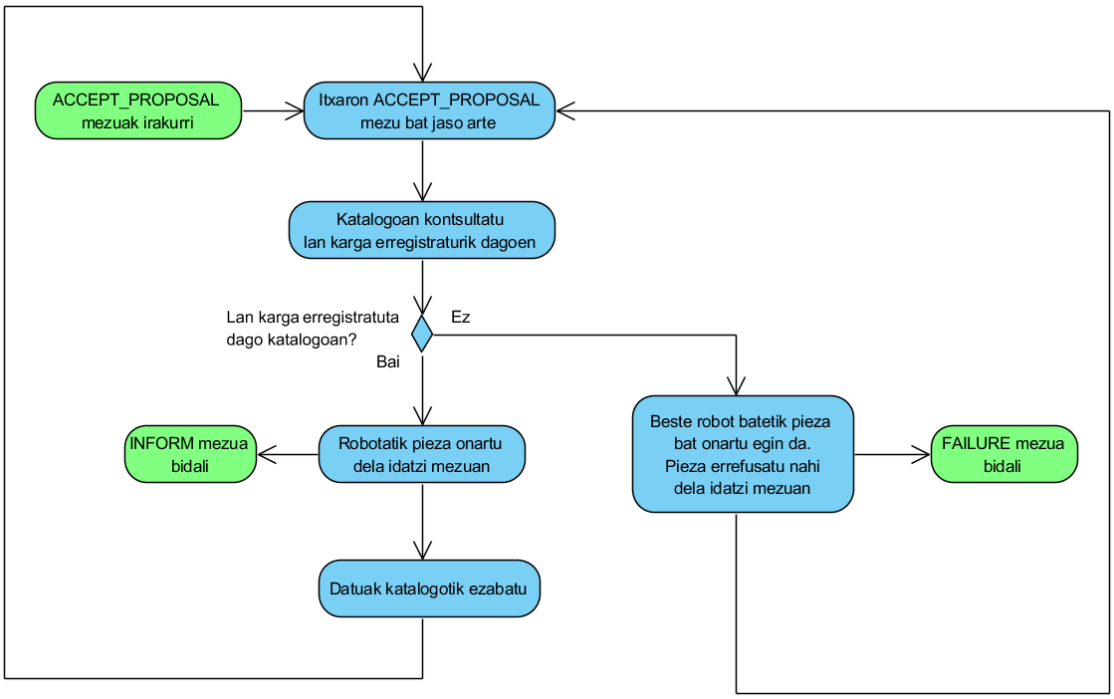


22. Irudia: Makina agentearen lehen hariaren aktibitate diagrama



23. Irudia: Makina agentearen bigarren hariaren aktibitate diagrama

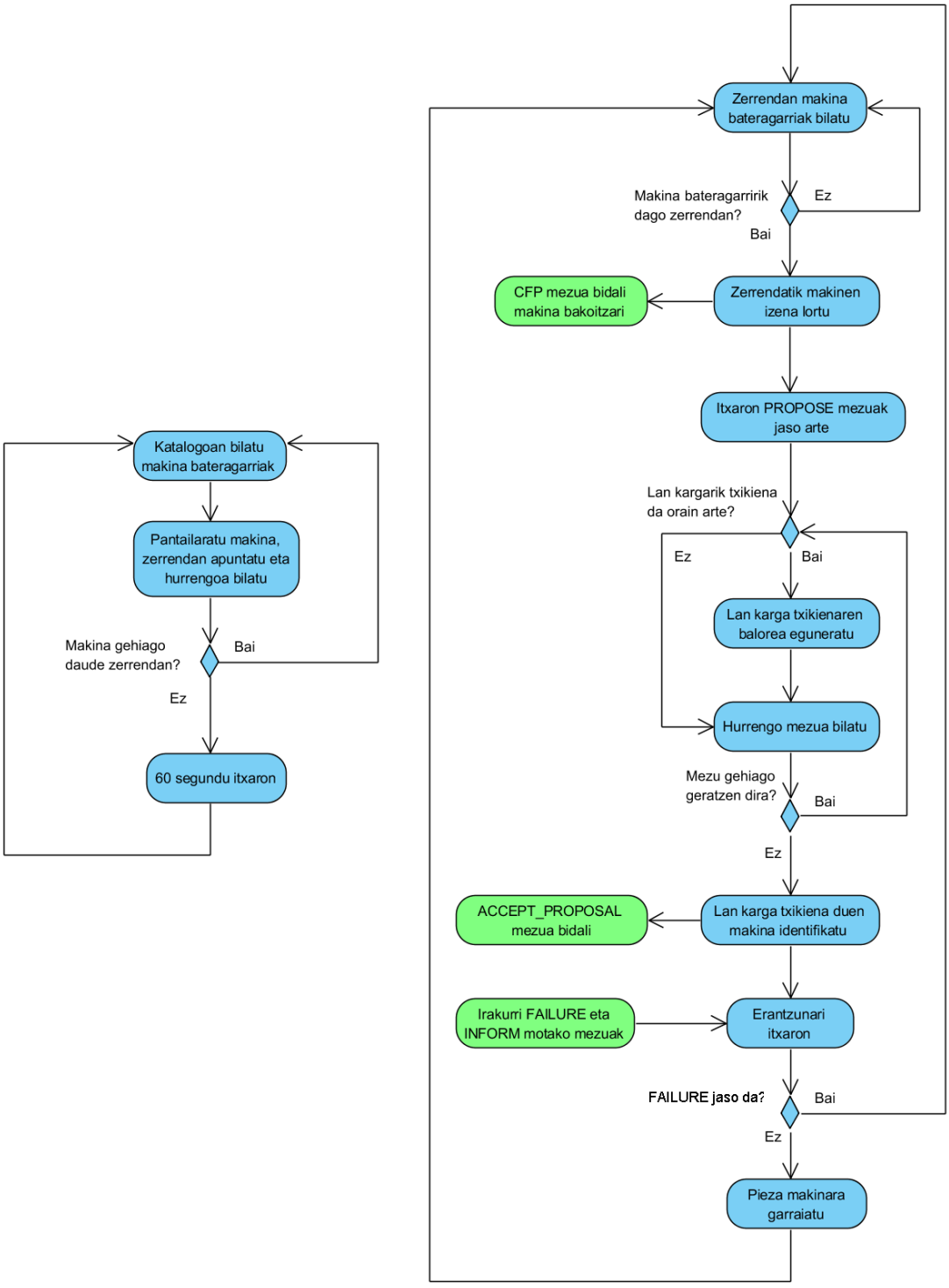
- Hirugarren hariak “ACCEPT_PROPOSAL” motako mezuak jaso egin diren konprobatzen ibiliko da. Jasotzekotan, katalogoan konprobatuko da lan karga oraindik erregistratuta dagoela. Katalogoan informazioa oraindik egotekotan robot agenteari “INFORM” motako mezu bat bidaliko zaio pieza ondo onartu dela esanez, eta ez egotekotan, “FAILURE” motako bat bidaliko zaio pieza ukatzeko eta errorea komunikatzeko.



24. Irudia: Makina agentearen hirugarren hariaren aktibitate diagrama

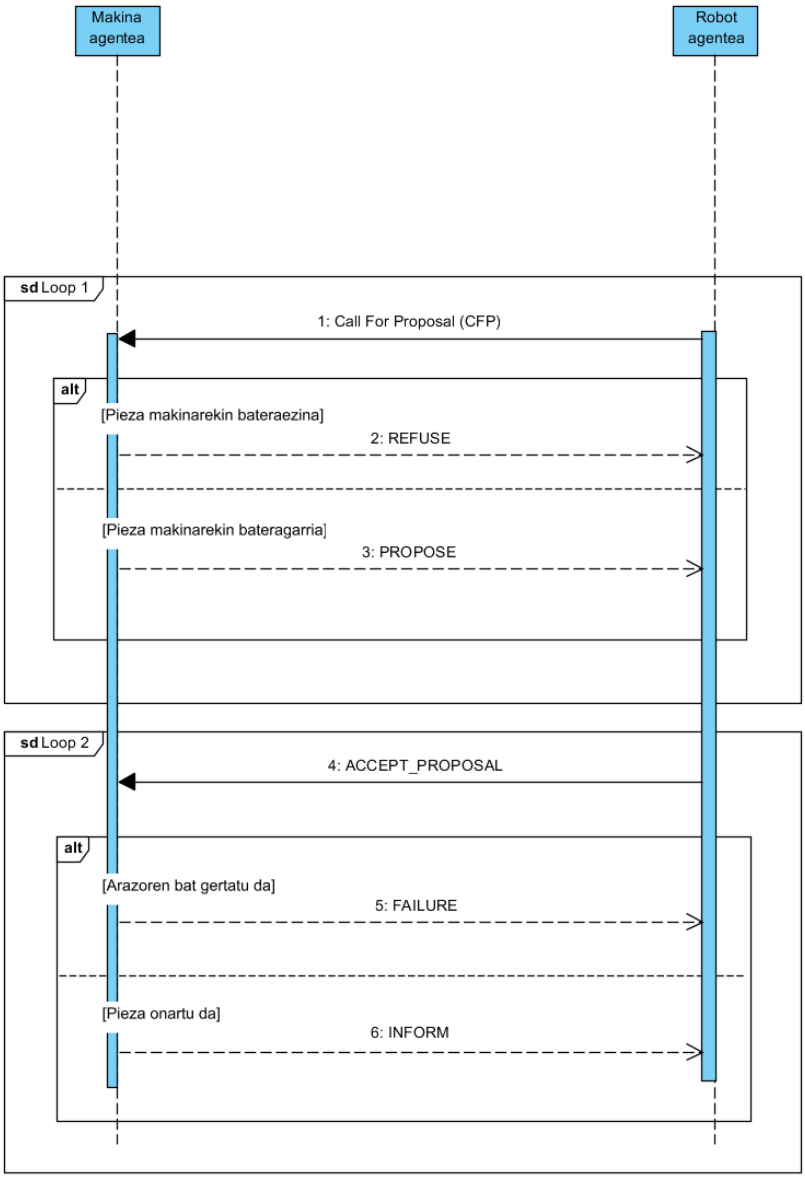
Robot agentea, simulatuko denez, programaren argumentuetan definituko da zer pieza mota garraiatuko duen. Agente honek 2 hari nagusi izango ditu:

- Lehen hariak 60 segundu oro pieza mekanizatu ditzaketen makinak zerrendatu egingo ditu.
- Bigarren hariak zerrendan makinaren bat egon arte itxaroten egongo da. Behin zerrendan piezarekin bateragarriak diren makinak izanda, banaka CFP motako mezua bat bidaliko zaie. PROPOSE motako mezuak banaka jaso egingo dira orduan, non mezu hauek lan karga balioa daramatzate. Mezu bakoitza jasotzean lan karga aurrekoa baino txikiagoa den konprobatuko da. Txikiagoa izatekotan, momentu hori arte txikiena zena eguneratuko da. Behin lan karga balorerik txikiena izanda, lan karga hori duen makina identifikatzen da eta makina hori aukeratzen da pieza eramateko.



25. Irudia: Robot agentearen lehen eta bigarren harien aktibitate diagrama

Agenteen arteko komunikazioaren ulerpena errazteko, makina agentearen eta robot agentearen komunikazioaren sekuentzia diagrama bat diseinatu da 26. irudian ikusten den bezalakoa.



26. Irudia: Makina eta robot agenteen sekuentzia diagrama

Bi prozesu zikliko daude. “Loop 1” prozesuan robotak CFP motako mezu bat bidaltzen dio makina agenteei lehendabizi. Mezu honi erantzuteko, makina agenteek adierazitako pieza mekanizatu dezaketen arabera PROPOSE (lan karga bidaltzeko) edo REFUSE (pieza ukatzeko) mezuak bidaliko dituzte. “Loop 2” prozesuan, robotak, lan karga txikiena duen agenteari ACCEPT_PROPOSAL mezua bidaliko dio. Mezu honi

erantzuteko, makina agenteak beste robot baten pieza aurretik onartu ez badu edo erroreren bat gertatu ez bada, pieza onartuko du eta INFORM mezu bat itzuliko du, bestela, FAILURE mezu batekin erantzungo du eta ez da piezarik bidaliko. Sekuentzia diagramako “Loop 1” prozesua makina agentearen aktibitate diagramaren bigarren hariarekin kointziditzen du. Aldiz, “Loop 2” prozesua makina agentearen aktibitate diagramaren hirugarren hariarekin kointziditzen du.

6. PLANGINTZA

Proiektuaren garapena hainbat fase edo zereginetan banatu da.

6.1. Fase eta zereginen deskribapena

Proiektua aurrera eramateko fase eta zereginetan banatu egin da garapena. 4 fasetan garatu egin da proiektua:

6.1.1. ODK formazioa

Proiektua hasteko, lehen fasean, ODK-ri buruzko ezagutzak finkatu behar izan dira. Fase honetan erabiliko diren teknologiak ezagutu egin dira eta adibide batez frogatu egin dira bere funtzionamendua bermatzeko hurrengo faseetan. III. taulan adierazitako zereginetan banatu egin da fase hau:

- Z.1.1 zereginean ET200SP Open Controller-a TIA Portalekin programatzeko konfigurazioa egin da.
- Z.1.2 zereginean ODK zer eta zertarako den eta nola erabiltzen den ikertu egin da.
- Z.1.3 zereginean ODK programatzeko, Visual Studio programa nola konfiguratu eta ODK-rekin bateratu ikertu da.
- Z.1.4 zereginean aurretik ikerketetan jasotako informazio guztia era praktiko batean probatu egin da.

III. Taula: ODK formazioa

Kodigoa	Deskribapena	Luzapena (egunetan)	Lan karga (orduetan)
F.1	ODK formazioa	55	170
Z.1.1	TIA Portal ET200SP-rako konfigurazioa egin	5	20
Z.1.2	ODK-ri buruz ikertu	15	30
Z.1.3	Visual Studio-ri buruz ikertu	10	20
Z.1.4	Adibide batekin PLC-aren programa eta ODK komunikatu	25	100

6.1.2. Java formazioa

Bigarren fasean Java aplikazio bat PLC-arekin komunikatu nahi izan da. Horretarako, aurretik ikertutako ODK programazioa erabiltzea beharrezkoa izan da. Hau lortzeko, IV. taulan adierazitako zereginetan banatu egin da fase hau:

- Z.2.1 zereginean Java nola programatzen den ikasi egin da.
- Z.2.2 zereginean Java eta C++ aplikazio bat komunikatzeko sistema bat bilatu behar izan da. IPC sistema batzuen abantailak eta desabantailak zerrendatu eta gero, socket bidezko konexioa aukeratu da, alternatiben analisian azalduta dauden arrazoiengatik.
- Z.2.3 zereginean aurreko zeregineko IPC sistema era praktiko batean probatu egin da, Java eta C++ aplikazio bi socket bat erabiliz komunikatuz.
- Z.2.4 zereginean PLC-a ODK erabiliz Java aplikazio batera konektatzea lortu da.

IV. Taula: Java formazioa

Kodigoa	Deskribapena	Luzapena (egunetan)	Lan karga (orduetan)
F.2	Java formazioa	50	150
Z.2.1	Javari buruz ikertu	15	30
Z.2.2	IPC sistemei buruz ikertu eta bat aukeratu	5	20
Z.2.3	C++ aplikazio bat eta Java aplikazio bat komunikatu adibide baten bidez	20	80
Z.2.4	ODK aplikazio bat eta Java aplikazio bat komunikatzea adibide baten bidez	10	20

6.1.3. JADE formazioa

Agenteak proiektuan integratzeko, hirugarren fasean, JADE-ri eta agenteei buruz ikertu egin da. V. taulan adierazita dago bete egin diren zereginak:

- Z.3.1 zereginean multi-agente plataforma bat zer eta zertarako den ikertu egin da.
- Z.3.2 zereginean multi-agente plataforma bat garatzeko JADE Java estrukturari buruz ikertu egin da. JADE programatzeko programa baten aukeraketa egin da, baita ere.
- Z.3.3 zereginean IntelliJ IDEA programa nola erabili ikasi egin da.
- Z.3.4 zereginean agente bidezko plataforma baten adibidea eraiki da eta agenteen arteko komunikazioa probatu egin da.

V. Taula: JADE formazioa

Kodigoa	Deskribapena	Luzapena (egunetan)	Lan karga (orduetan)
F.3	JADE formazioa	35	120
Z.3.1	Multi-agente plataformei buruz ikertu	5	10
Z.3.2	JADE-ri buruz ikertu	10	40
Z.3.3	IntelliJ IDEA-ri buruz ikasi	5	10
Z.3.4	JADE adibide bat eraiki agente anitzekin	15	60

6.1.4. Plataformaren diseinua

Laugarren eta azken fasean aurretik ikasi den guztia proiektu batean inplementatzen da. VI. taulan plataforma eraikitzeko bete egin diren zereginak zerrendatu egin dira:

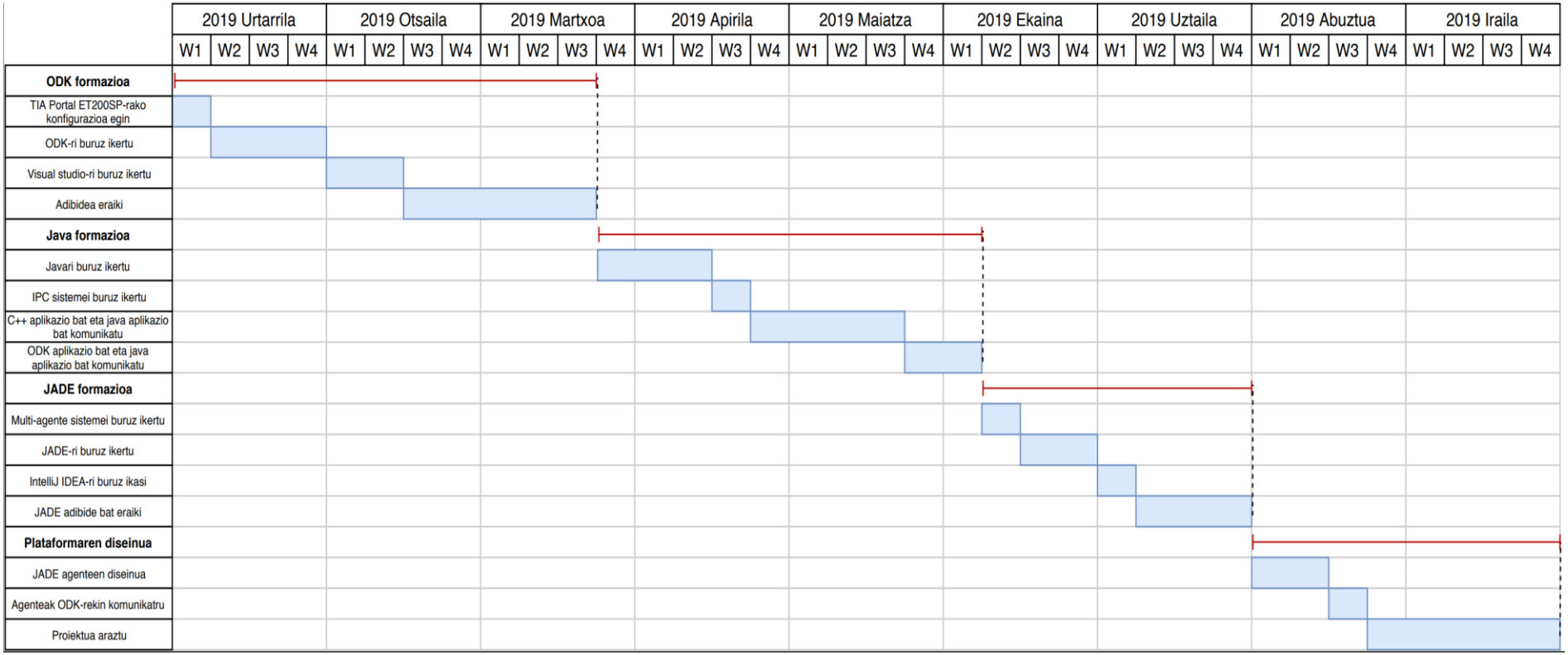
- Z.4.1 zereginean JADE erabiliz espezifikazio teknikoetan adierazitakoarekin bat egiten duen bi agente egin dira, robot agentea eta makina agentea, alegia.
- Z.4.2 zereginean makina agenteari informazioa pasatzeko ODK aplikazioarekin komunikatu egin da, socket-ak erabiliz.
- Z.4.3 zereginean proiektua kasu erreal batean sor zitezkeen arazoak aurreikusi eta zuzendu egin dira.

VI. Taula: Plataformaren diseinua

Kodigoa	Deskribapena	Luzapena (egunetan)	Lan karga (orduetan)
F.4	Plataformaren diseinua	40	110
Z.4.1	JADE robot eta makina agenteen diseinua	10	40
Z.4.2	Makina agentea ODK-rekin komunikatu	5	20
Z.4.3	Proiektua araztu	25	50

6.2. Gantt diagrama

Aurretik zerrendatutako faseak kronologikoki irudikatzeko 27. irudian adierazitako Gantt diagrama diseinatu da. Proiektuaren garapena urtarriletik irailera luzatu egin da.



27. irudia: Proiektuaren plangintzaren Gantt diagrama

7. AURREKONTUA

Proiektuak garatzeko faseak behin zehazturik, aspektu ekonomikoak baloratuko dira. Egingo den aurrekontua proiektua garatzeko hiru aurrekontuetan banatuko da. Alde batetik, VII. taulan, erabili diren osagai komertzialak zerrendatuko dira, hardware-a eta lizentziak besteak beste. Gero, VIII. taulan, eskulanaren aurrekontua egingo da, non zuzendari bat eta ingeniari elektroniko bat proiektua garatuko dute.

VII. Taula: Osagai komertzialak

Osagaia	**Prezioa (€)**	**Kopurua**	**Totala (€)**
SIMATIC ET200SP Open Controller CPU 1515SP 64bit + HMI	3903,28	1	3903,28
ODK V2.0 software paketea eta lizentzia	2530,39	1	2530,39
IntelliJ IDEA lizentzia komertziala	499	1	499
Visual Studio lizentzia komertziala	1199	1	1199
Ordenagailuaren amortizazioa (kostua orduko)	0,045	550	24,75
Sarrera digital modulua	43,17	2	86,54
Irteera digital modulua	55,78	2	111,56
		GUZTIRA	**8354,52**

VII. taulako prezioetan BEZ-a dagoeneko gehituta dago. Ordenagailuaren amortizazioaren kostua orduko 4 urteko bizitza erabilgarria duela eta 1599 euroko prezioa izan duela kontuan harturik kalkulatu da.

VIII. Taula: Eskulana

Postua	**Prezioa/egun (€/ordu)**	**Ordu kopurua**	**Kopurua**	**Totala (€)**
Proiektu zuzendaria	25	100	1	2500
Ingeniari Elektronikoa	15	550	1	8250
			GUZTIRA	**10750€**

AURREKONTU OSOA:……. 19.104,52€

8. ONDORIOAK

Gradu amaierako lan hau garatu eta fase guztien betetzea lortu eta gero hurrengo ondorioak atera egin dira:

Hasteko, automata bat sistema eragile batekin komunikatzearen abantailak erakutsi egin dira. Komunikazio sistema honek makinak manufaktura malguko sistema batean integratzea baimendu egingo du etorkizunean, agente eta PLC-aren arteko komunikazio bidirekzionala lortu egin delako.

Erabili den teknologia ingurune akademiko batean proiektuak garatzeko nahikoa bada ere, fabrika erreal batean inplementatzeko oraindik errore batzuen soluzioan lan egin behar du Siemens-ek. Open Controller-a memoria txikia duenez ezin ditzake ODK-ren aplikazioak abiadura handiz kargatu eta deskargatu, 13, 14 eta 15. irudietan ageri den 32573 errorea ematen du eta.

Beste alde batetik, garatu diren JADE agenteek bi aldagai baino ez dute kontuan hartzen, pieza mota eta lan karga, eta beraz, ez dira nahikoak fabrika erreal batean inplementatzeko, ingurune honetan aldagai gehiago ageri dira eta. Hau dela eta, sistema komertzial bat sortu aurretik, kontuan hartu egingo diren aldagai guztien zerrendaketa eta programazioa egin beharko da, geroago dagoeneko diseinatuta dagoen plataforma honetan integratzeko. Hala ere, proiektu honen xedea ez da hau, eta funtzionamendu kontzeptua adierazteko balio egin da, adibide gisa.

Azkenik, etorkizunari begira, PLC-ari makina baten programa erreal bat esleitzeko beharra dago. Ikerketa taldearen hurrengo pausua hauxe da beraz, makina baten biki birtualaren diseinuaren eta programazioaren bitartez.

9. BIBLIOGRAFIA

1.(d.g.).https://support.industry.siemens.com/cs/attachments/109741218/s71500_odk1500s_manual_en-US_en-US.pdf?download=true helbidetik eskuratua

2. (d.g.). http://www.tpautomation.de/Automatisierungssysteme/SIMATIC-ET-200/ET-200SP/CPUs-und-Controller/6ES7677-2AA31-0EB0-ET-200SP-Open-Controller-CPU-1515SP-PC-32-Bit::32074.html?MODsid=a744e9e5dd79bdf4ab3973a4881e36c5 helbidetik eskuratua

3. (d.g.). http://www.tpautomation.de/Automation-systems/SIMATIC-ET-200/ET-200SP/CPUs-and-Controller/6ES7677-2FA41-0FB0-ET-200SP-Open-Controller-CPU-1515SP-PC-F-64-Bit::34543.html?language=en helbidetik eskuratua

4. (d.g.). http://www.tpautomation.de/Automatisierungssysteme/SIMATIC-ET-200/ET-200SP/CPUs-und-Controller/6ES7677-2AA41-0FM0-ET-200SP-Open-Controller-CPU-1515SP-PC-HMI-2048-PT-64-Bit::32078.html?MODsid=a744e9e5dd79bdf4ab3973a4881e36c5 helbidetik eskuratua

5. (d.g.). https://jade.tilab.com/download/jade/ helbidetik eskuratua

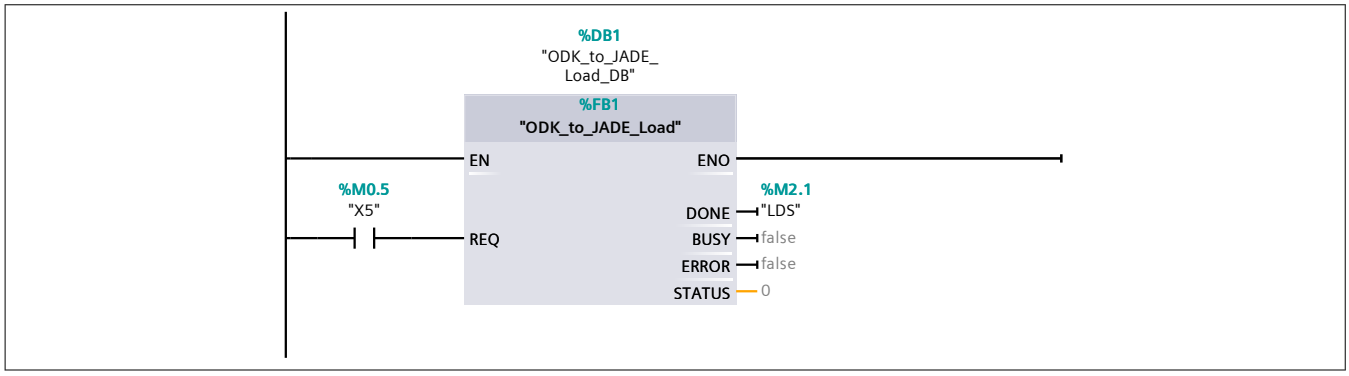
10. ERANSKINAK

Konbinazionala [OB123]

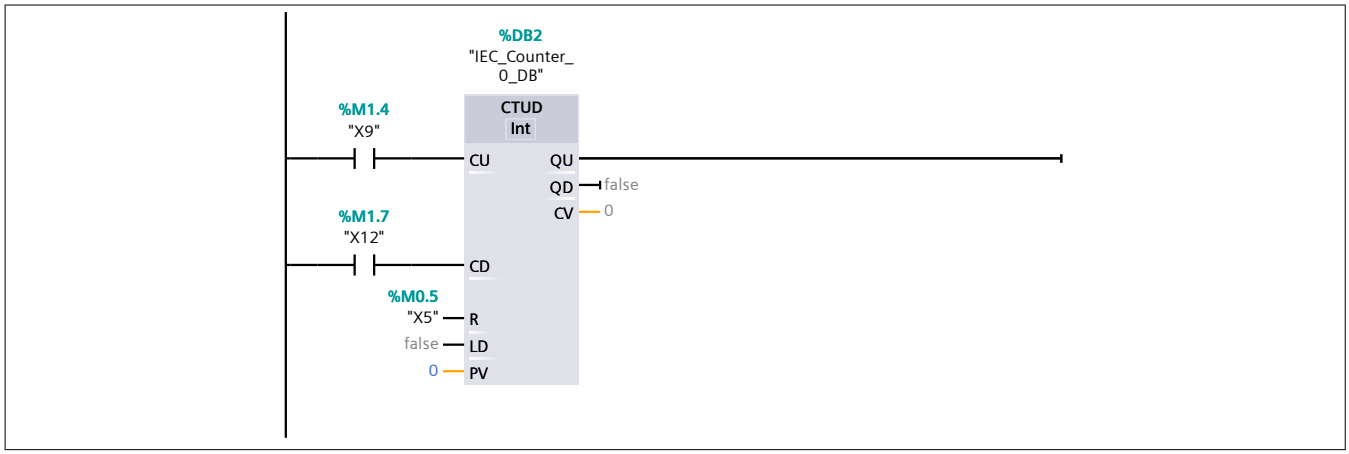
Konbinazionala Propiedades					
General					
Nombre	Konbinazionala	**Número**	123	**Tipo**	OB
Idioma	KOP	**Numeración**	Automático		
Información					
Título	"Main Program Sweep (Cycle)"	**Autor**		**Comentario**	
Familia		**Versión**	0.1	**ID personalizado**	

Nombre	Tipo de datos	Valor predet.	Comentario
▼ Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

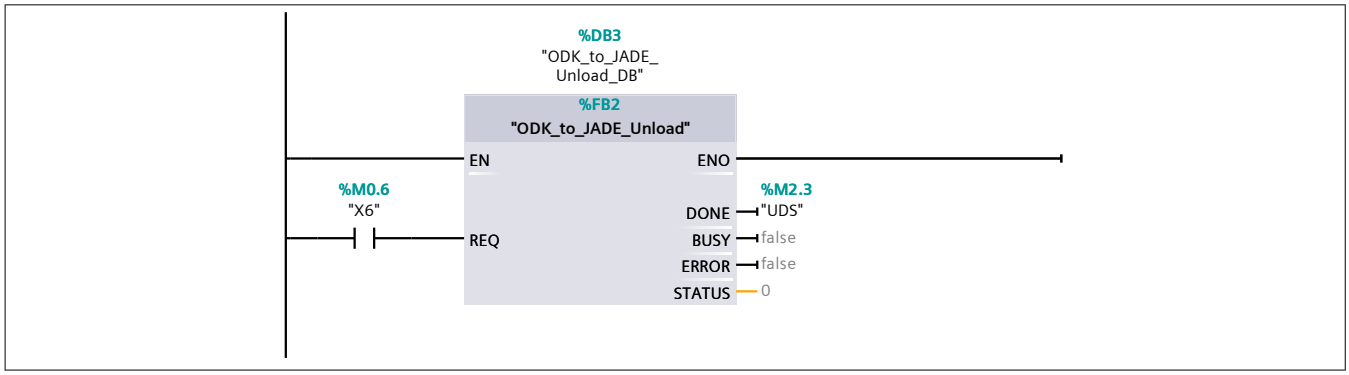
Segmento 1:



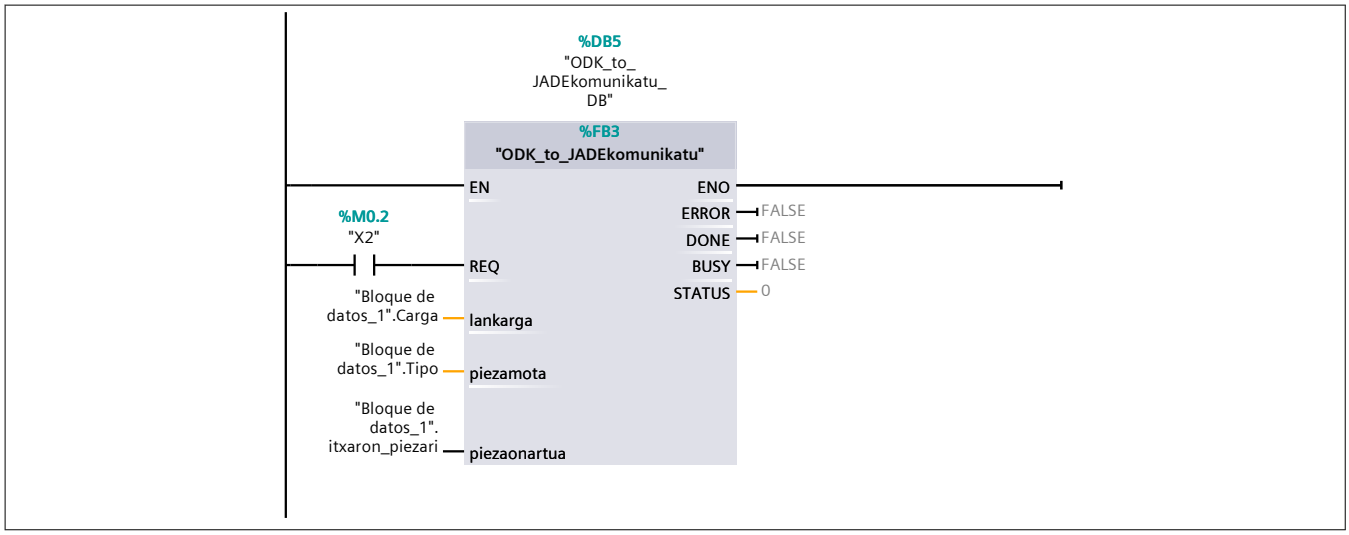
Segmento 2:



Segmento 3:



Segmento 4:



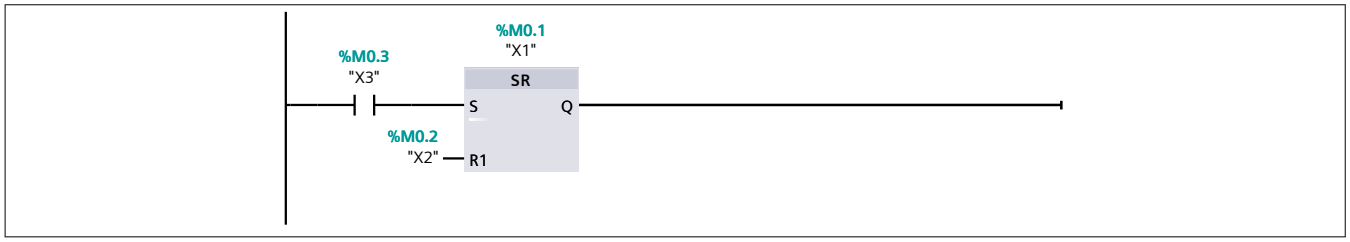
Segmento 5:

Sekuentziala [OB1]

Sekuentziala Propiedades					
General					
Nombre	Sekuentziala	**Número**	1	**Tipo**	OB
Idioma	KOP	**Numeración**	Automático		
Información					
Título	"Main Program Sweep (Cycle)"	**Autor**		**Comentario**	
Familia		**Versión**	0.1	**ID personalizado**	

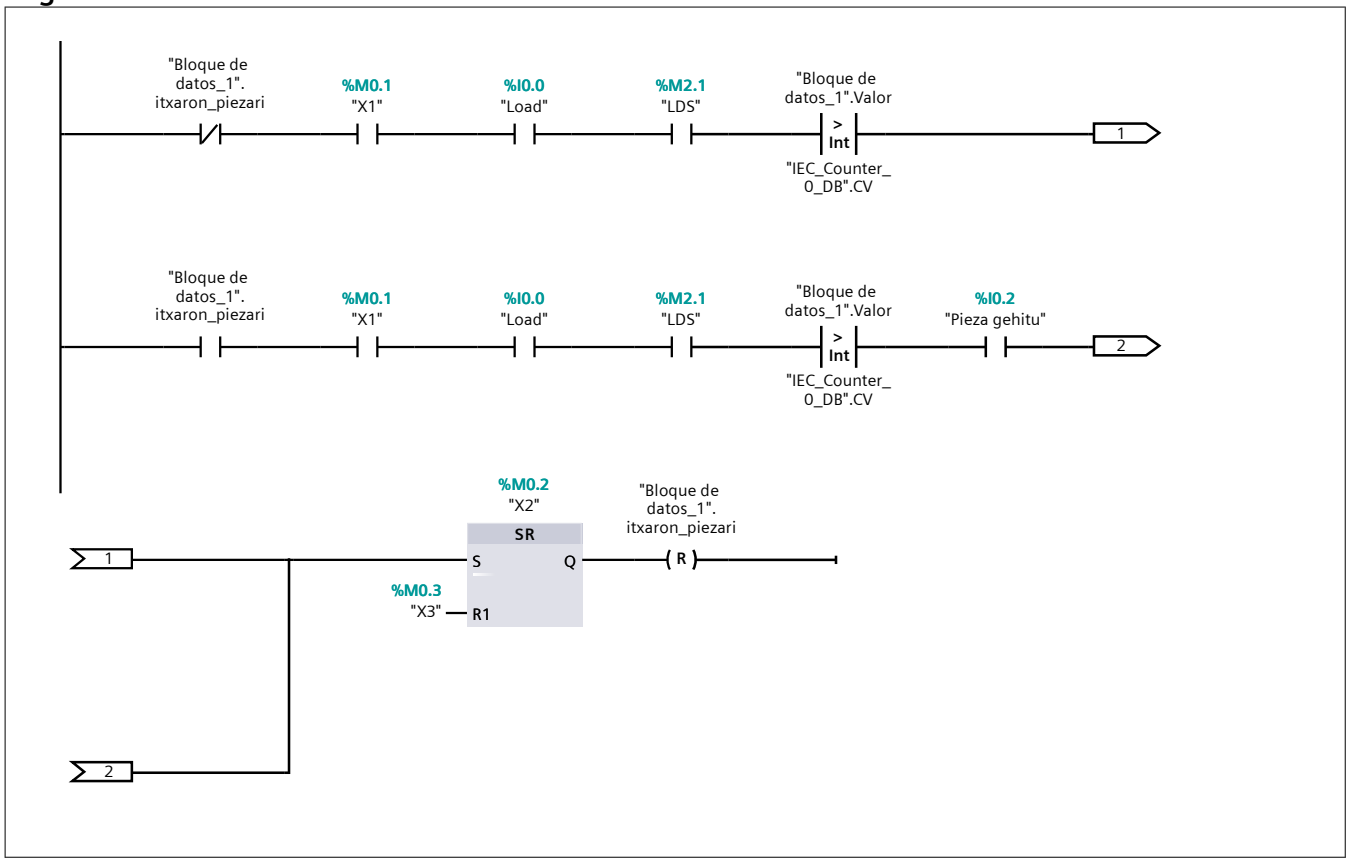
Nombre	Tipo de datos	Valor predet.	Comentario
▼ Input			
Initial_Call	Bool		Initial call of this OB
Remanence	Bool		=True, if remanent data are available
Temp			
Constant			

Segmento 1:

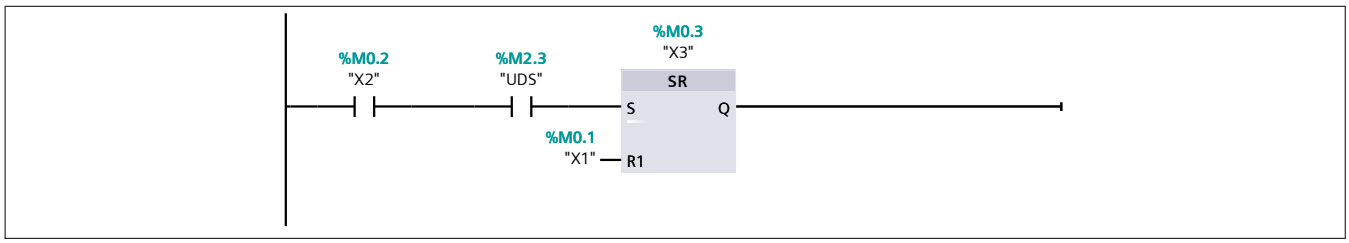


Segmento 2:

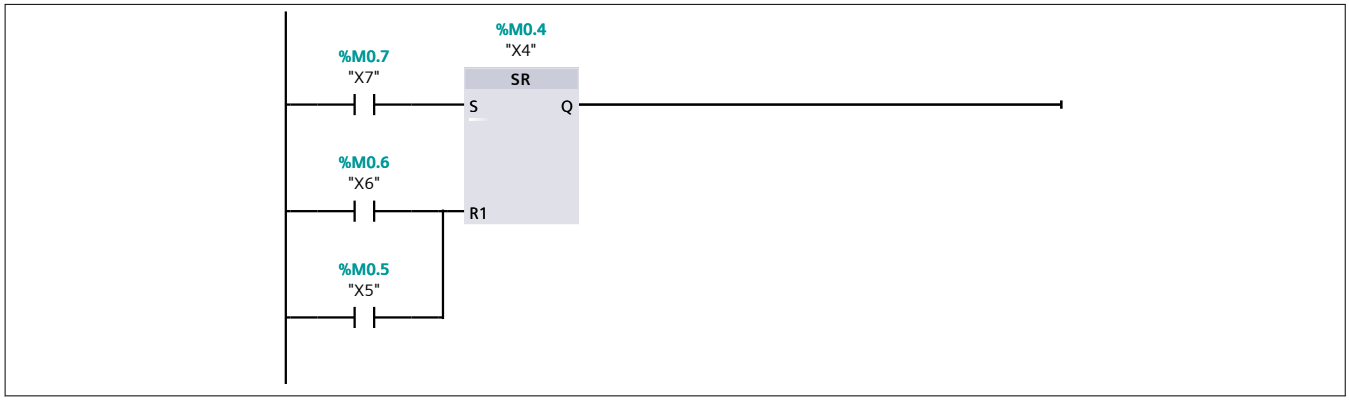
Segmento 2:



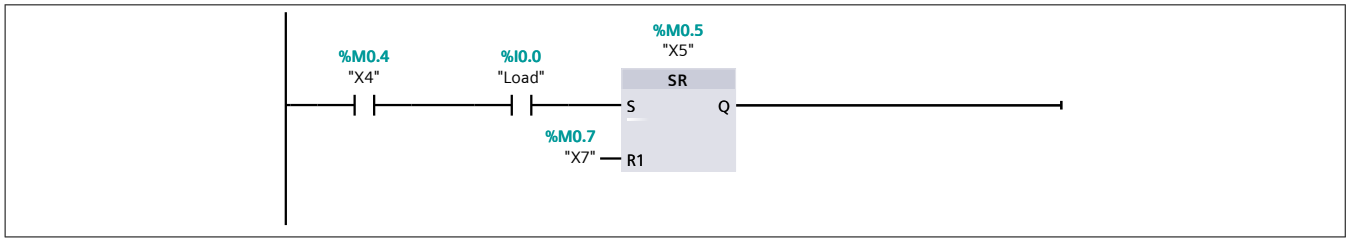
Segmento 3:



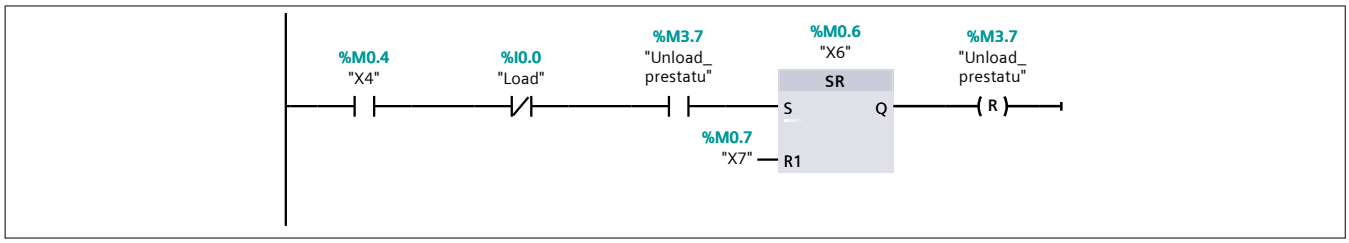
Segmento 4:



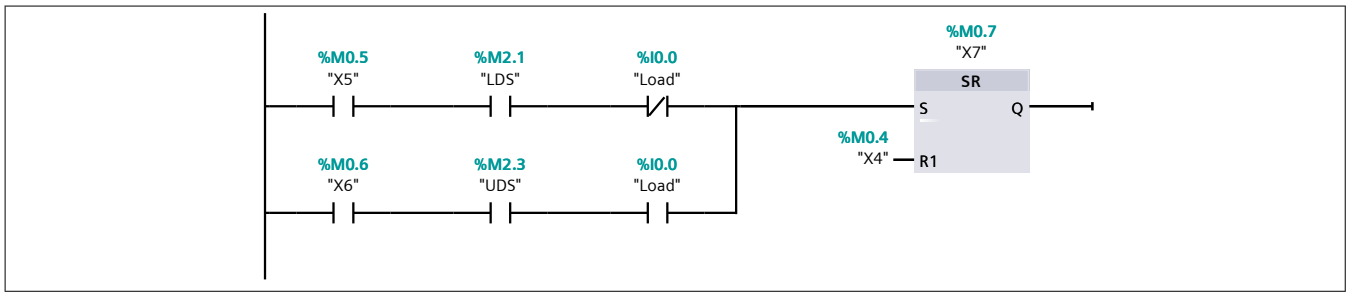
Segmento 5:



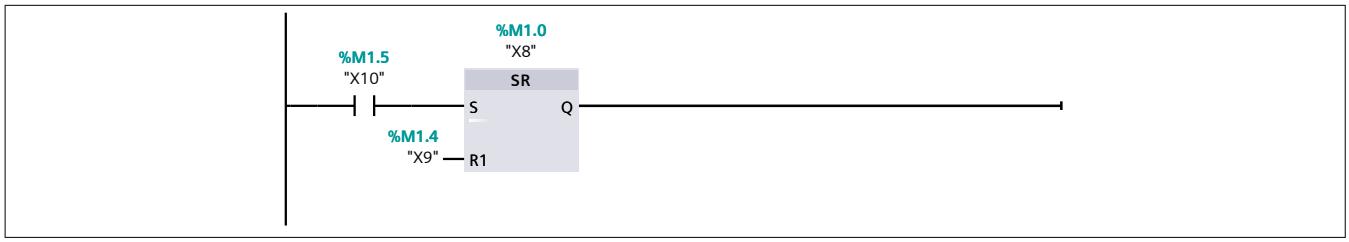
Segmento 6:



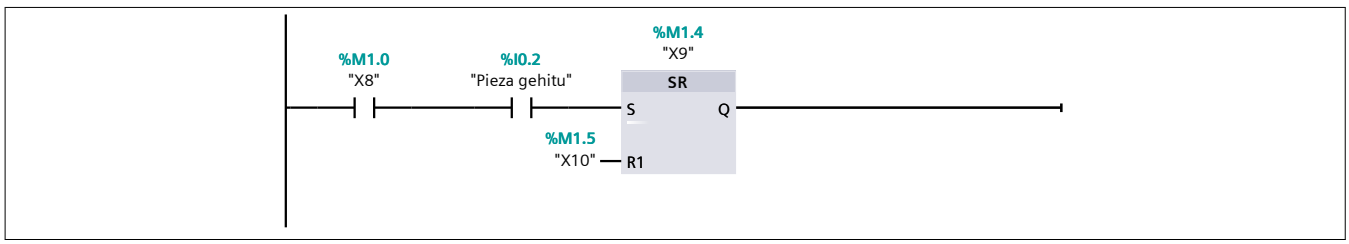
Segmento 7:



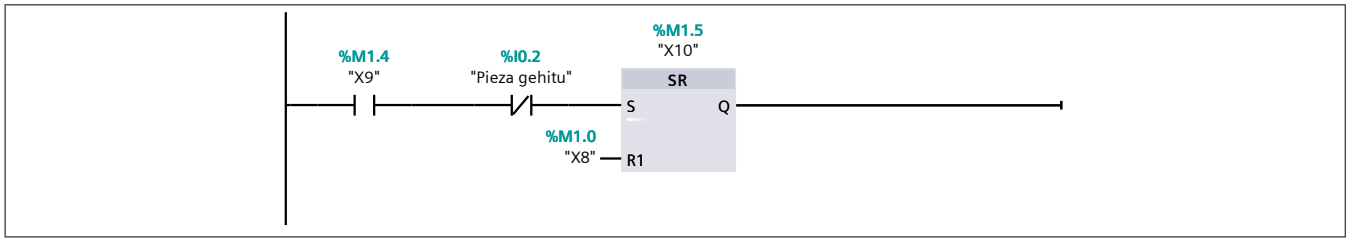
Segmento 8:



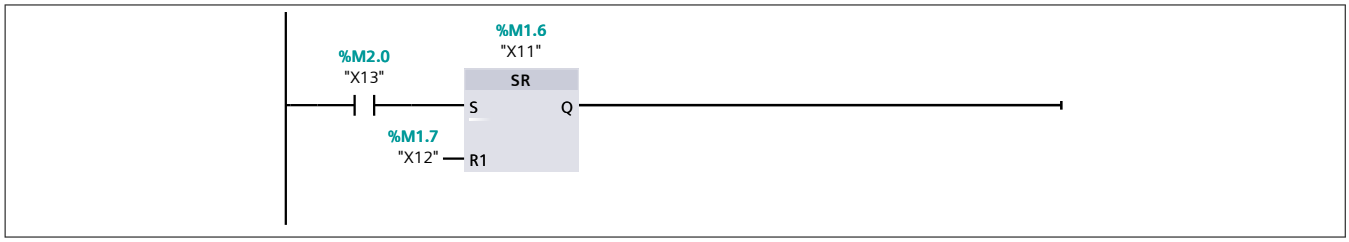
Segmento 9:



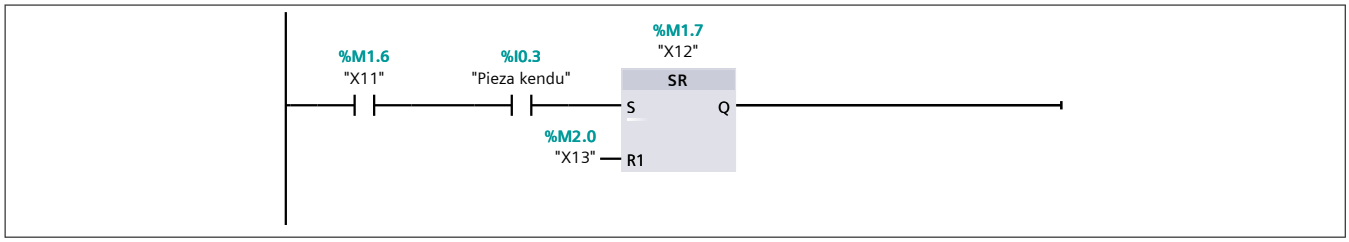
Segmento 10:



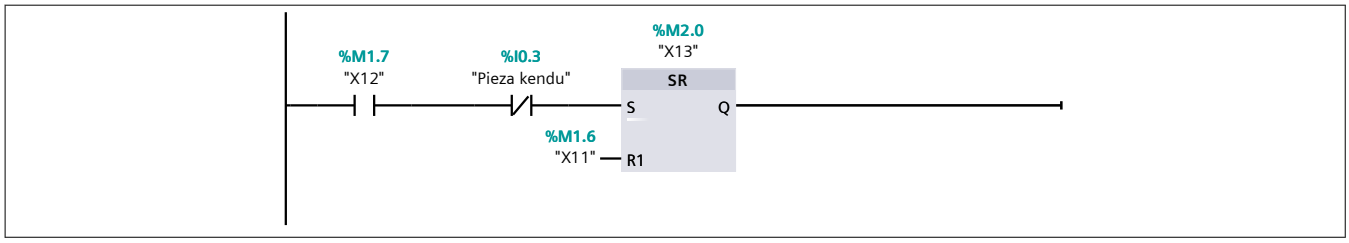
Segmento 11:



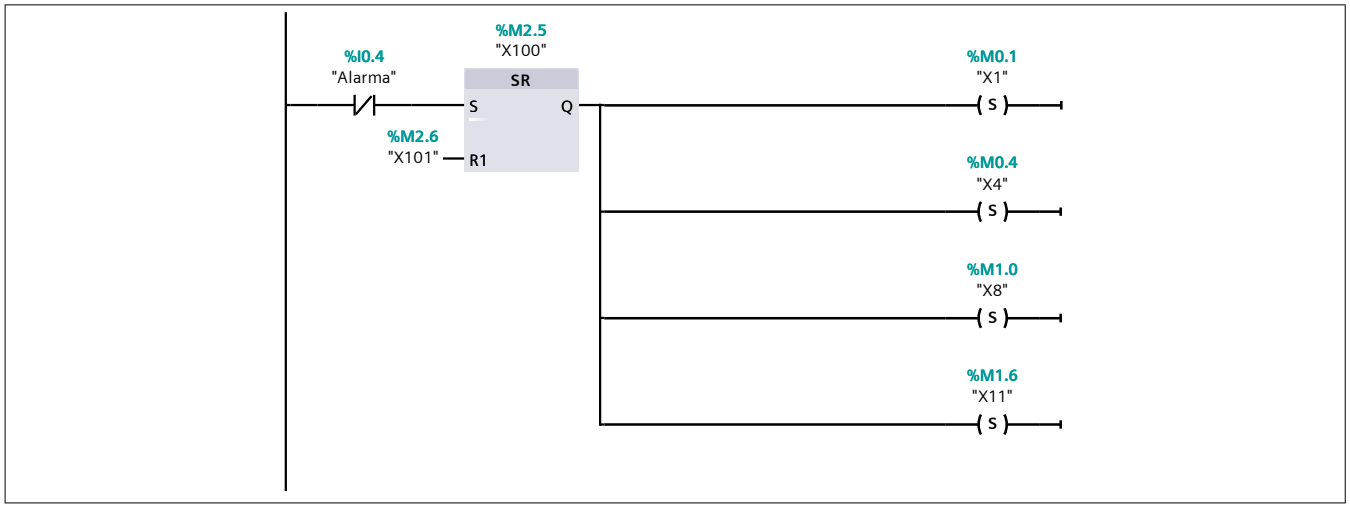
Segmento 12:



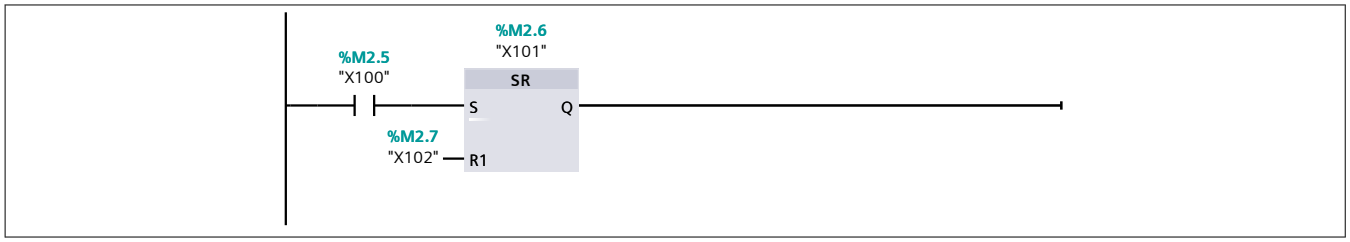
Segmento 13:



Segmento 14:



Segmento 15:



Segmento 16: