**GIPUZKOAKO INGENIARITZA ESKOLA**

**ESCUELA DE INGENIERÍA DE GUIPUZKOA**


**GRADU AMAIERAKO LANA/ TRABAJO FIN DE GRADO**

# Denbora errealeko PLC-aren inplementazioa Linux-a duen ARM mikrokontrolagailuaren bidez

**DOKUMENTUA: MEMORIA**

Gradua: Industria Elektronikaren eta Automatikaren Ingeniaritzako Gradua

Ikaslea: Jon Cortajarena Alcorta

Zuzendaria: Mikel Alberro Astarbe

Ikasturtea: 2017-2018

Defentsa eguna: 2018ko maiatzaren 17a

# LABURPENA

## Proiektuaren proposamena

BeagleBone Black txartela batean CODESYS-ean oinarritutako PLC bat ezarri nahi da. Txartel honek, ARM mikrokontrolagailu bat du Linux txertatuarekin. CODESYS ezarri ondoren, sistema denbora errealean dagoen balioztatzeko probak egin behar dira. Amaitzeko, EtherCAT, PROFINET eta CANopen eremu busak ezarriko dira. Eremu bus hauekin hainbat proba egingo dira urruneko komunikazioekin eta beste PLC batzuekin komunikatuz.

## Proiektuaren garapena

Proiektua Dutt enpresan garatu da Albizturren kokatua dagoena eta 800 ordu inguru sartu dira eskatutako helburuak betetzeko. Gainera, langile bat gainean izan dut laguntza moduan zalantzak argitzen saiatzeko, nahiz eta beraiek Linux-a ez kontrolatu. Orduan, nire kabuz moldatu behar izan naiz gehienbat. Askotan arazo ugari izan ditut baina denborarekin aurrera jarraitzeko kapaza izan naiz.

## Etapak

1. Linux-ari buruz informatu.
2. BeagleBone Black plaka ezagutu eta ordenagailuan instalatu bere softwarea.
3. BeagleBone Black-ean CODESYS ezarri.
4. CODESYS-en bitartez hardwareko irteera bat programatu denbora tarte finkoan eta horrela saiakerak egin denbora errealean balio duen ikusteko.
5. Linuxekin operazio ugari egin aurreko programarekin eta honek irteeran eraginik duen edo ez ikusi.
6. EtherCAT eremu busa BeagleBone Black-era konektatu eta BECKHOFF urruneko komunikazioekin komunikatzen saiatu.
7. PROFINET eremu busa BeagleBone Black-era konektatu eta SIEMENS urruneko komunikazioekin komunikatzen saiatu.
8. CANopen eremu busa BeagleBone Black-era konektatu eta BECKHOFF urruneko komunikazioekin komunikatzen saiatu.

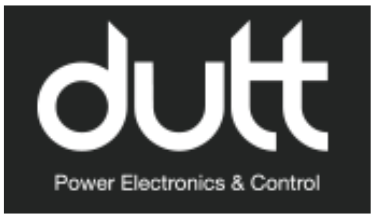
# AURKIBIDEA

# IRUDIEN AURKIBIDEA

# TAULEN AURKIBIDEA

# 1. SARRERA

Dutt, Montelec Montajes Electrónicos,S.L. Albizturren (Gipuzkoa-Spain) kokatua dago eta 1996-tik beraien helburua hurrengoa da: kontrol eta potentzia elektronika behar dituzten enpresei egokitutako elektronika eskaintzea. Beraien arloen interesak energia eta / edo potentziaren kontrol eta bihurketa eskatzen duten edozein sektorek estaltzen dute; hau da, eguzki, industrial, energia biltegiratzea, igogailu industriala eta abar.



Irudia 1: Dutt-eko logoa.

Paperezko industriarako makineriaren fabrikazio talde garrantzitsuenetako bateri dagokio, Pasaban taldeari. Talde honen beharren ondorioz, hasieran hardwarea eta softwarea teknologia propioarekin motor-kontrolerako soluzioak diseinatzen hasi ziren, ondoren beraien gaitasunak handitzeko, energia bihurtzeko eskainitako ekipoen diseinuaren bitartez. Aldi berean aplikazio estatiko eta birakariak sinkronizatzen.

20 urte hauetan eskuratutako esperientziari esker, DSP, mikrokontrolagailuetan, komunikazio industrialetan, softwarean, IGBT driverretan eta potentzia bihurgailuetan bezalako diziplinetan beharrezko ezagutzak garatzea posible izan da.

Espezializazioaren adibide bezala, potentzia-bihurgailuak, inbertsoreak, artezgailuak, DC-DCak, motorreko kontrolagailuak, PLCak eta sistema osatugabeak, esate baterako, maiztasun bihurgailuak eta biltegiratze energetikorako bihurgailuak etab. egiten dira. Gainera, Dutt-eko teknologiarekin eta fabrikazioarekin egiten da dena, 35 kW-tik 1MW-ra arteko potentzia estaliz.

Diseinu guztiak enpresan egiten dira. PCB-tik hasita, potentziako bihurgailuak eta PLC-tik pasaz, armairu modularra muntatu arte. Horretarako, 2.300 m2-ko instalazioetan baliabide propioak dituzte muntatzeko eta SMD probak egiteko, baita muntaketa lerroak eta potentzia elektronikan espezializatutako muntaiak.

Dutt-ek diseinatu eta fabrikatzen dituen ekipamenduaren % 95a esportaziora bideratzen da, bost kontinenteetan zabalduta egonik 365egun / 24h aplikazioetan.

Egunero beste fabrikatzaile eta Ingeniariren bazkide gisa parte hartzen duten proiektuak dituzte, beraien laguntza emanez inbertsoreetan, kargagailuetan, bihurgailuetan eta abarretan garatzen dituzten gaitasunak izanik.

Gertaera gogoangarriak:

**1996** Montelec-en sorkuntza.

**1999** Kontrol txartelen lehenengo generazioa. Serie 2000.

**2003** Lehen bihurgailua Montelec teknologiarekin eta Serie 3000.

**2005** 2.300 m2-ko instalazio berriak Albizturren eta SMD teknologia.

**2006** IGBT kontrol-babesaren teknologia propioaren txertatzea.

**2011** Serie 4000-ren garapena.

**2013** 850 kW-ko potentzia handieneko inbertsore trifasikoa fabrikatzen da.

**2014** Energia biltegiratzeko bihurgailuaren garapena.

**2015** Marka berriaren onartzea: Dutt.

# 2. PROIEKTUAREN ERAKUSKETA

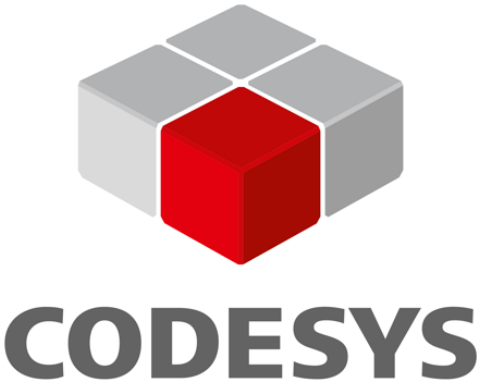
Proiektu honetan, PLC bat ezarri nahi da; non CODESYS programaren bitartez BeagleBone Black plaka erabili nahi den. Honekin, bi funtzio betetzea nahi da. Bata, denbora errealean funtzionatzen duen ikusi beharko da, enpresaren eskakizuna milisegundoko periodoan beraiek zehaztutako kodea exekutatzea delako. Enpresarentzako denbora errealean exekutatzeko plakak sortzen duen Jitterra periodoaren %1-a baino txikiagoa izan behar du. Beste funtzioa, hiru eremu busekin beste hainbat proba egitea da. Hiru eremu busek funtzionatuz gero, Linux-ean gehiago dakien pertsona bat kontratatuko da proiektuarekin jarraitzeko.

Ondoren, proiektu honetako atal nagusiak ezagutuko dira:

## 2.1. CODESYS

### 2.1.1. Sarrera

CoDeSys (controller development system), ingelesetik etorritako akronimo bat da, “kontrolagailuen garapen sistema” esan nahi duena. CODESYS automatizazio industrialaren teknologien plataforma da. Gainera CODESYS, kontrolagailuen aplikazioak programatzeko garapen ingurunea da, non IEC 61131-3 nazioarteko industriako estandarrarekin ados egon behar duten. Azken erabiltzaileei irtenbide integratuak eskaintzen dizkie automatizazio aplikazioetan ingeniaritzako proiektuetan. Helburua erabiltzaileei beren zereginak gauzatzean praktikarako bideratutako laguntza ematea da.




Irudia 2: CODESYS logoa.

### 2.1.2. Ingeniaritza

IEC 61131-3-an definituta dauden bost programazio lengoaiak aplikazioen programaziotarako, eskuragarri daude CODESYS-en garapen ingurunean:

- IL (instruction list) programazio-hizkuntza bezalako mihiztatzailea da.
- ST (structured text) Pascal edo C programazioen antzekoa da.
- LD (ladder diagram) programatzaileak erreleko kontaktuak eta bobinak bateratzeko aukera ematen du.
- FBD (function block diagram) programatzaileak erabiltzaileari espresio boolear eta analogikoetan programatzen uzten dio.
- SFC (sequential function chart) egokia da fluxu eta prozesu sekuentzialak programatzeko.

Editore grafiko gehigarria CODESYS-en, non IEC estandarrean ez dagoen zehaztuta:

- CFC (Continuous Function Chart) esku hutsezko FBD motako editorea da. Sarera bideratutako FBD editoreaz gain, sarrera, operadore eta irteeren arteko konexioak automatikoki konfiguratzen dira, non programatzaileak marraztu behar dituen. Kaxak libreki jar daitezke, eta horri esker, feedback-zikloak aldi baterako aldagairik gabe bideratzeko aukera ematen du.

Gailu-fabrikatzaileek CODESYS erabiltzen dute beraien automatizazio osagarrien konfigurazio eta programazio automatikoak egiteko, ahal den neurrian eskalatzeko, sistemaren funtzio desberdinak erabiliz. Automatizazio sistemaren maila desberdinetako produktu integratuak CODESYS-en arrakastaren oinarria dira. Tresna honek proiektuaren ingeniaritza, programazioa, funtzionamendua, exekuzioa, kontrolaren edo unitatean aplikazioaren kodearen arazketa eta eremu gailuen ebaluazioa estaltzen du.

Konpilazio integratuek CODESYS-ek sortutako aplikazioaren kodea makina kodea (bitar kodea) bihurtzen dute, non kontrolagailura deskargatzen den. 16 eta 32 biteko CPU familia garrantzitsuenak onartzen dira, hala nola C166, TriCore, 80x86, ARM / Cortex, Power Architecture, SH, MIPS, BlackFin eta gehiago.

Behin CODESYS online jartzean, arazteko funtzionalitate zabala eskaintzen du, esate baterako, monitorizazio/idazketa/behartutako aldagaiak ebaketa-puntuak zehazteko/banakako urratsak eginez edo kontrol-linean dauden aldagaiak erregistratzen dira eraztun buffer batean (laginaren bidea).

CODESYS Application Composer, aplikazioak sortzeko balio du existitutako moduluak erabiliz. Erabiltzaileak konposatu, parametrizatu eta beharrezko moduluak konektatzen ditu aplikazio osoa osatzeko. Konfigurazio honek ez du PLC programazioari buruzko ezagutza behar; beraz, programazioan esperientzia ez duten teknikariek ere lan egin dezakete. Barne sorgailuek IEC 61131-3 osoko aplikazioak sortzen dituzte eta ondo egituratua dago, mapak eta I / O pantailak barne dituelarik. Application Composer-ek moduluak garatu eta osatzeko baimena behar du. Gainera, erabilera libreko moduluak (Persistence Manager, Device Diagnostics…) daude lizentzia gabe erabili ahal izateko.

### 2.1.3. Runtime-a

CODESYS Control Runtime System ezarri ondoren, gailu adimentsuak CODESYS bidez programatu daitezke. Kargatutako tresna-kate batek exekuzio-sistema hau iturburu-kode eta objektu gisa eskaintzen du. Plataforma desberdinetara bideratu daiteke.

### 2.1.4. Fieldbus teknologia

Eremu bus desberdinak CODESYS programazio sisteman erabil daitezke zuzenean. Horretarako, tresnak sistema ohikoarentzat konfiguratzaileak integratzen ditu, PROFIBUS, CANopen, EtherCAT, PROFINET eta EtherNet / IP bezalakoak. Sistema batzuetarako, protokolo gehigarri eskuragarrien pilak daude liburutegi forman, non geroago CODESYS-en kargatu daitezkeen. FDT (Field Device Tool) Frame aplikazioan softwarearen osagarri baten bidez kanpoko hornitzaileen gailu osagarrien erabiltzailearen interfaze espezifikoak integratu daitezke. Interfaze horien arteko komunikazioa, Device Manager-en (DTM) bidez egingo da.

## 2.2. Denbora erreala

### 2.2.1. Sarrera

Denbora errealeko sistema operatibo (RTOS, Real-Time Operating System) bat, denbora errealeko sarrera batean erreakzionatzen duen informatika ingurune bat da. Denbora errealeko denbora oso txikia izan daiteke eta sistemaren erreakzioak berehalako dirudi.

Sistema operatibo baten beharrenetarako bat egin behar diren hainbat jarduerei ordenagailuko baliabideak esleitzea da. RTOS batean prozesu hau oso konplexua da, zenbait jarduera aldi baterako kritikoak dira eta batzuek besteek baino lehentasun handiagoa dutelako. Antolatzaileak (scheduler) lehentasunen eskema baten arabera antolatzeko, zereginei lehentasunak esleitzeko medio bat egon behar du. Horrez gain, fitxategiak, I/O gailuak eta erabilgarritasun-programak bateratzeko hainbat atazen eta ezaugarri estandarren artean memoria kudeatzeko gaiak gehitu behar ditugu. Sistema osoaren kontrola CPU-aren esleipeneko zereginen moduluan erortzen da. Modulu hau sarritan monitore edo exekutibo gisa ezagutzen da.

RTOS-etan aplikazioa eta sistema operatiboa oso estu lotuta daude. RTOS batek aurrez zehaztutako tarteak edo kanpoko gertaerak modu determinista erantzuteko gai izan behar du, beste zeregin edo prozesu batzuek izan dezaketen eraginaz gain. Denbora errealeko ingurune batean, zeregin kritikoek behar dituzten sistemako baliabideak jaso behar dituzte behar den unean, beste zeregin batzuek sor ditzaketen ondorioak izan arren.

Izan ere, sistema operatiboen portaera ere baldintzatzen du beste ikuspegien aurrean, hala nola fitxategi sistema eta I/O gailuen kudeaketa.

### 2.2.2. Denbora errealeko Sistema operatiboen baldintzak

RTOS-en baldintza bereziak bost eremu orokorretan aurkezten dira:

1. Determinismoa
2. Sentsibilitatea
3. Erabiltzailearen kontrola
4. Fidagarritasuna
5. Hutsegite tolerantzia

1. Determinismoa:
   Hortaz, sistema operatibo bat determinista da instante finko eta aurrez zehaztuak edo aurrez zehaztutako denbora tarteetan eragiketak egiten baditu. Prozesu ugari baliabideak lortzeko lehiaketan daudenean, prozesadoreak barne, sistema ez da guztiz determinista izango. Sistemak modu deterministan eskakizunak asetzeko gai den puntuaren araberakoa den jakiteko, lehenik eta behin, eskatutako denbora tartean eskakizun guztiak kudeatzeko ahalmen nahikoa duen ikusi behar da. Modu deterministan jarduteko sistema operatibo baten gaitasunaren neurketa erabilgarria lehentasun handiko etenaldiaren etorrerako gehienezko atzerapena da, elkartutako errutinaren zerbitzua hasten den arte. RTOS batean, denbora hau mikrosegundo gutxi batzuetatik 1 milisegundora joan daiteke. Denbora errealean ez dauden sistema operatiboetan, atzerapena hamarretik ehuneko milisegundo bitartekoa izan daiteke.

2. Sentsibilitatea:
   Aurrekoaren antzeko ezaugarria da, eten bat aintzat hartzeko sistema operatibo batek irauten duen denbora da. Denbora hori, etena onartu ondoren, zerbitzua emateko denbora zehatza da. Honen araberakoa da:
   Etenaldiaren kudeaketarekin hasteko eta tratamenduaren errutina exekutatzen (ISR, Interrupt Service Routine) hasteko behar den denboraren araberakoa da. ISR-en exekuzioak prozesu aldaketa eskatzen badu denbora prozesua luzeagoa izango da.

3. Erabiltzailearen kontrola:
   Oro har, askoz handiagoa da RTOS-ean. Denbora elkarbanatuan, erabiltzaile batek ezin die beren prozesuei lehentasuna eman, plangintza algoritmoari buruz erabaki, zein prozesuek egon behar duten beti memorian, etab.

4. Fidagarritasuna:
   Oro har, askoz ere garrantzitsuagoa da RTOS-ean. Denbora errealeko sistema batek ingurumenean eta bere denbora tarteetan egiten diren gertaerak kontrolatzen ditu; non kontrolatzen duten sistemaren galerek edo degradazioek ondorio kaltegarriak izan ditzakete.

5. Hutsegite tolerantzia:

RTOS bat hutsegiteen aurrean erantzuteko diseinatu behar da, non gehienezko ahalmena eta gehieneko datu posibleak mantentzen saiatzea den helburua hutsegiteren bat gertatzen den kasuan. Esate baterako, memoriaren edukia fitxategi batean gordetzea eta saihesteko programa ixtea guztiz debekatuta daude hutsegite bat agertzean. RTOS bat arazoa zuzentzen edo efektuak minimizatzen saiatuko da exekuzioarekin aurrera jarraitu baino lehen.

Hutsegite tolerantziari lotuta egonkortasuna dago. Sistema bat egonkorra izango da eginkizunak betetzeko epeak betetzen ez badira, baina gutxienez epe horretan lehentasun kritiko eta handienekoak betetzen badira.

### 2.2.3. Denbora errealeko sistema operatiboen ezaugarriak

Aurreko baldintzak betetzeko, RTOS-ek eskainitako ezaugarriak honela laburbil daitezke:

1. Denbora errealeko prozesuen plangintzarako laguntza.
   RTOS batek hainbat prozesu sortzea, ezabatzea eta plangintza egiteko laguntza eman behar du, eta horietako bakoitzak aplikazio baten zati bat monitorizatu edo kontrolatu behar du. Normalean, RTOS batean, prozesu eta etenetarako lehentasunak definitu daitezke.

2. Lehentasunezko plangintza (pre-emptive).
   RTOS batek lehentasunezko prozesu bat ziurtatu behar du exekutatzeko prest dagoenean lehentasun gutxiagoko prozesu baten aurrean. Sistema operatiboak baldintza ezagutzeko gai izan behar du (normalean, eten baten bidez), exekutatzen ari den prozesuaren aurretik igaro eta lehentasun handiagoko prozesu bat gauzatzeko testuinguru aldaketa azkarra egin.

3. Etenaldien aurrean erantzunaren garantia.
   RTOS batek etenaldi edo ekitaldi baten agerraldia oso azkar ezagutu behar du, eta ekintza determinista bat hartu behar du ekitaldi horri erantzuteko. Hardware eta software motako etenei erantzun behar zaie. Sistema operatiboa etenik gabea eta berrabiagarria izan behar du. Etenaldiak indeterminismoaren iturri sarrera dira eta latentziaren agerpena inposatzen dute.

4. Prozesuen arteko komunikazioa.
   RTOS batek prozesuen arteko komunikazioak modu fidagarrian eta zehatzean jasateko gai izan behar du, hau da, semaforoak, mezuen igortzea eta memoria partekatua adibidez. Erraztasun horiek prozesuen exekuzioa sinkronizatzeko eta koordinatzeko erabiltzen dira, baita datuen babesa eta baliabideak partekatzeko ere.

5. Abiadura handiko datuak eskuratzea.
   Beharrezkoa da sistemak datu multzoak maneiatzen gai izatea eskuratze-abiadura handi batekin. Modu honetan, RTOS batek diskoan datuen biltegia optimizatzeko aukera ematen du, batez ere I/O bufferretan. Bestelako funtzionalitate osagarriak batzuk adibidez, fitxategietan alboko disko blokeak aurrez esleitzea izan daitezke

(biltegiratze sekuentziala) eta erabiltzaileari bufferren gaineko kontrolak ematea.

6. I/O-en laguntza.
   Denbora errealeko aplikazioek normalean I/O interfaze ugari dauzkate. RTOS batek I/O gailu zehatzak erraz txertatzeko tresnak eman behar ditu. Gailu estandarretarako, I/O liburutegi estandarra nahikoa izango litzateke. Gainera, I/O asinkronoak ere jasan behar dituzte, eta ondoren exekuzioarekin jarraitu I/O eragiketa egiten ari den bitartean. Alderdi honetan, gogoratu behar da prozesadore zehatzak (I/O kanalak, DMA kontrolagailuak ...) existitzen direla eta I/O eragiketak egiten dituztela CPU-aren laguntzarik gabe.

7. Sistemaren baliabideak kontrolatzea erabiltzailearen partetik.
   RTOS baten funtsezko ezaugarri bat sistemaren baliabideen kontrol espezifikoak dituzten erabiltzaileei edukiera ematean datza; non, CPU-a, memoria eta I/O baliabideak barne dauden. CPU-aren kontrola, lehentasunezko plangintza batean oinarritzen da eta bertan erabiltzaileek prozesuen lehentasunak ezarri ditzakete. Horrez gain, denbora errealeko tenporizadoreak eta funtzioak daude eskuragarri gertaerak eta itxaronaldiak planifikatzeko. RTOS batek memoriaren blokeoa erraztu beharko luke, horrela, programa bat edo horren zati bat memoria batean geratzen dela ziurtatzen da, testuinguru aldaketak lehenago egin ahal izateko etenaldia gertatzen denean. Erabiltzaileari bufferren esleipenean eta fitxategiak eta gailuak blokeatzeko edo desblokeatzeko gaitasuna baimentzeko kapaza izan beharko luke. Kontrol hori orokorrean askoz ere handiagoa da RTOS-ean sistema operatibo generikoan baino. Azken honetan, erabiltzaile batek ez du bere prozesuen lehentasunean, plangintza algoritmoan, memoria birtualen orrialdeak ixterakoan eta diskoko espazioa erreserbatzearen kontrola.

### 2.2.4. Lehentasunen egitura

RTOS batean, diseinatzaileak zereginei lehentasunak eman behar dizkie, lehentasun handiagoa emanez denbora-tarte zehatzagoak dituztenei eta sistemaren funtzionamendu egokia izateko beharrezkoak direnei. Oro har, zereginak lehentasuneko hiru mailatan banatu daitezke.

- Etenaldiak: maila honetan etenaldi-zerbitzuen errutinak daude, non erantzuna abiadura handian egiten duten zeregin eta gailuetarako (milisegundotan neurtuta). Zeregin horietako batzuk, denbora errealeko erlojua eta zereginak aztertzen dituen erlojua dira.
- Erlojua: prozesatze errepikakorra eskatzen duten zereginak denboran zehar, adibidez, laginketa-aldian exekutatu beharreko kontrol algoritmoak.
- Oinarria: Lehentasun txikiko zereginak dira eta exekuzioan atzerapenak izan ditzake, ez direlako denborazko zehaztapen sendoak edo sistemarentzat ezinbestekoak. Zeregin hauen adibide tipikoak erabiltzailearentzako interfazea zaintzen dutenak izan daitezke, datuak eskatuz edo pantailan erakutsiz.

#### 2.2.4.1. Etenaldi maila

Etenaldi batek CPU-aren lanaren berrantolaketa indartzen du eta sistemak berrantolaketa horren denboraren kontrola galtzen du. Hori dela eta, beharrezkoa da errutinen kodea azkar exekutatzea. Gainera, beharrezkoa da kode zati bat edukitzea informazio hegakorra biltegiratzeko, esate baterako, erabiliko dituen erregistroen edukia, horrela errutina gero eta lehentasun txikiagoan exekutatuz (erlojua edo oinarri maila), berreskuratutako informazio horrekin lan egin eta planifikazioa leheneratuz.

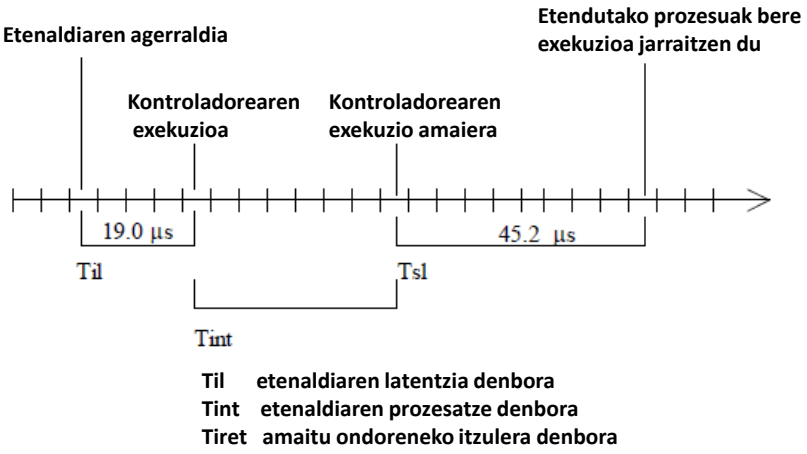
Hardware etenaldien barruan lehentasun mailak ere ezartzen dira. Normalean hardware espezifikoek zehaztu ohi dute, non etenaldi lerro batzuk eta lehentasunezko erabakimen zirkuitu bat dituen.

### 2.2.5. Sistemen prestazioa denbora errealean

Ordenagailuen abiadura areagotu den arren, hau ez da amaigabea, eta kontrol-sistemen konplexutasunean antzeko neurria handitzen lagundu du. Hori dela eta, aplikazio batzuetan, ezinbestekoa da CPU-aren zikloak alferrik ez botatzea. Era berean, garrantzi handikoa da kanpoko ekitaldi bat agertzetik ekitaldi horrekin erlazionatutako kodearen exekuzioraino denbora tarte hori txikitzea. Denbora hori latentzia bezala ezagutzen da. Edozein sistematan latentzia modu asko aurki daitezke, sistemaren prestazioak karakterizatzeko erabiltzen ohi direnak.

#### 2.2.5.1. Etenaldien latentzia

Hardware etenaldi baten seinalearen harreratik etenaldien kontrolagailuko lehenengo software instrukzioa exekutatua izan arteko denbora tarteari deritzo. Kontuan hartu behar da etenaldi bat agertzen denean programaren kontadoreko balioa pilan gorde behar dela, ziurrenik egoera hitza ere. Etenaldien bektoreen taula batera joan behar da eta programaren kontadorean dagokion balioa kargatu behar da. Ekintza guzti hauek hardware bidez egiten dira automatikoki eta denbora gutxi kontsumitzen dute, baina etenaldiren bat aldi baterako desgaituta egoten bada, kasu horretan latentzia handiagoa izango da.



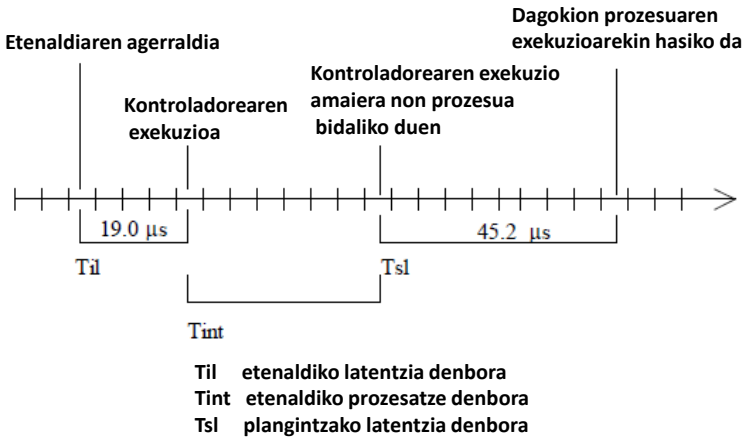

Irudia 3: Etenaldien latentzia.

Latentzia denbora hori prozesadoreko erregistroen biltegiratze egoerari dagozkion atomiko txikiagotan banatu daiteke, edo hainbat etenaldi gertatzen diren aldi berean gertatu daitezkeen beste latentzia batzuk agertzea eta orduan, hardwareak lehenbailehen erabaki behar du zeini jaramon egin lehenago.

#### 2.2.5.2. Plangintzaren latentzia

Zenbait kasutan, ekitaldi bat agertzean, prozesu berri bat abiaraztea beharrezkoa izango da. Plangintzaren latentzia, etenaldi kontrolagailuaren amaieraren eta etenaldiak sortutako prozesuaren exekuzioaren arteko denbora igaro ahala zehaztuko da. Denbora honek zein prozesu planifikatu, testuinguruan aldaketak egin eta denbora bat non sistemak prozesu berrien “preemption”-a desaktibatuta edukiz egiaztatzeko beharrezko atzerapenak barne hartzen ditu. Denbora hau beharrezkoa da erakusleak doitzeko eta sistemako datu esklusiboen egituretara sartzeko.

Bi latentzia motetan batez besteko denborak azter daitezke, baina kasu kritikoetarako latentzia denbora okerrenak kontuan hartu behar dira. Denbora hauek etenaldien eskaerak eskatzean agertuko lirateke aldi berean, non etenaldiei arreta aldi baterako desaktibatuta egongo diren edo proiekzio zerrendak oso beteta dauden.




Irudia 4: Plangintzaren latentzia.

## 2.3. BeagleBone Black

### 2.3.1. Sarrera

BeagleBone Black gaur egun gehien saldu den plaka da merkatuan, BeagleBone familiako merkeena delako. BeagleBone Black, BeagleBone familiako plakarik erabiliena da, non hainbat aplikazio eta hardwarean oinarritutako proiektuetarako erabiltzen diren. Gainera, programatzaileentzat eta afizionatuentzat euskarria du mundu guztian. Sitara™ AM335x 1GHz ARM Cortex A8 prozesadore bat du eta bere konfigurazioagatik, 10 segundo baino gutxiagotan Linux-a abiarazi dezake.

BeagleBone Black, Debian GNU / Linux ™ -ekin bidaltzen da FLASH erantsi batekin bere ebaluazio eta garapenarekin hasteko. Linux-eko beste hainbat distribuzio eta sistema operatibo ere BeagleBone Black-arekin bateragarriak dira, besteak beste:

- Ubuntu
- Android
- Fedora

BeagleBone Black-aren gaitasunak zabaltzeko, BeagleBone Black-aren 46 pineko konektoreetan kapak izeneko plakak erabiltzen dira. Kapak eskuragarri daude hurrengoetarako: VGA, LCD, motor kontrola, prototipoak, bateria eta beste funtzionalitate batzuetarako.

### 2.3.2. Ezaugarri nagusiak

BeagleBone Black-a BeagleBone familiako plakarik berriena da. Gainera, merkea da eta Texas-eko prozesadore indartsu bat erabiltzen du. BeagleBone plakaren antzekoa da

baina ezaugarri berri ugari ditu eta beste batzuk hobetuak izan dira. Ondorengo taulan ikus daitezke beraien arteko diferentziak:

| | BeagleBone Black $55 | BeagleBone $89 |
|---|---|---|
| Processor | AM3358BZCZ100, 1GHZ | AM3359ZCZ72, 720MHz |
| Video Out | HDMI | None |
| DRAM | 512MB DDR3L 800MHZ | 256MB DDR2 400MHz |
| Flash | 4GB eMMC, uSD | uSD |
| Onboard JTAG | Optional | Yes, over USB |
| Serial | Header | Via USB |
| PWR Exp Header | **No** | **Yes** |
| Power | 210-460 mA@5V | 300-500 mA@5V |

Taula 1: BBB eta BB-aren arteko konparaketa.

Ondorengo taulan ikus daitezke BBB-aren ezaugarri nagusiak:

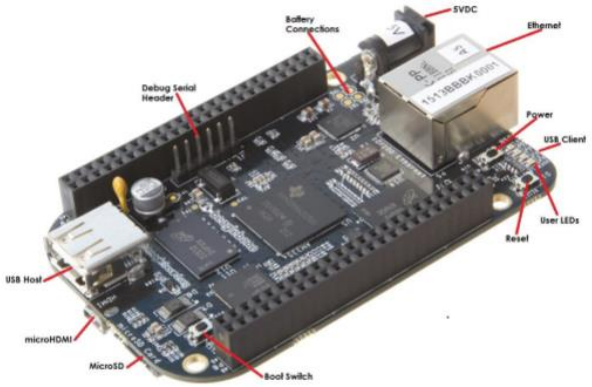
| | Feature | |
|---|---|---|
| Processor | Sitara AM3358BZCZ100<br>1GHz, 2000 MIPS | |
| Graphics Engine | SGX530 3D, 20M Polygons/S | |
| SDRAM Memory | 512MB DDR3L 800MHZ | |
| Onboard Flash | 4GB, 8bit Embedded MMC | |
| PMIC | TPS65217C PMIC regulator and one additional LDO. | |
| Debug Support | Optional Onboard 20-pin CTI JTAG, Serial Header | |
| Power Source | miniUSB USB or DC Jack | 5VDC External Via Expansion Header |
| PCB | 3.4" x 2.1" | 6 layers |
| Indicators | 1-Power, 2-Ethernet, 4-User Controllable LEDs | |
| HS USB 2.0 Client Port | Access to USB0, Client mode via miniUSB | |
| HS USB 2.0 Host Port | Access to USB1, Type A Socket, 500mA LS/FS/HS | |
| Serial Port | UART0 access via 6 pin 3.3V TTL Header. Header is populated | |
| Ethernet | 10/100, RJ45 | |
| SD/MMC Connector | microSD , 3.3V | |
| User Input | Reset Button<br>Boot Button<br>Power Button | |
| Video Out | 16b HDMI, 1280x1024 (MAX)<br>1024x768,1280x720,1440x900 ,1920x1080@24Hz<br>w/EDID Support | |
| Audio | Via HDMI Interface, Stereo | |
| Expansion Connectors | Power 5V, 3.3V , VDD_ADC(1.8V)<br>3.3V I/O on all signals<br>McASP0, SPI1, I2C, GPIO(69 max), LCD, GPMC, MMC1, MMC2, 7 AIN(1.8V MAX), 4 Timers, 4 Serial Ports, CAN0, EHRPWM(0,2),XDMA Interrupt, Power button, Expansion Board ID<br>(Up to 4 can be stacked) | |
| Weight | 1.4 oz (39.68 grams) | |
| Power | Refer to Section 6.1.7 | |

Taula 2: BBB-aren ezaugarri nagusiak.

### 2.3.3. BBB-aren osagaiak

#### 2.3.3.1. Konektoreak, LED-ak eta switchak

5 . irudian ikus daitezke BBB-ren I/O konektoreak, LED-ak, switchak, eta komunikaziorako portuak besteak beste.




Irudia 5: BBB-ko konektoreak, LED-ak, switchak etab.

- 5VDC: 5Vko tentsioa ematen dion DC sarrera nagusia.
- Ethernet: LAN-erako konexioa.
- Power: pizteko eta itzaltzeko erabiltzen den botoia.
- USB client: ordenagailura konektatzeko erabiltzen den USB esklabua. Honek bakarrik ere tentsioa eman dezake.
- User LEDs: erabiltzaileak erabiltzen dituen lau LED-ak.
  - USR0, plaka abiaraztean taupada moduan kliskatzeko konfiguratuta dago.
  - USR1 microSD txartelaren sarbidean zehar piztuta egoteko konfiguratuta dago.
  - USR2 CPU-aren jardueran zehar piztuta egoteko konfiguratuta dago.
  - USR3 eMMc-aren sarbidean zehar piztuta egoteko konfiguratuta dago.

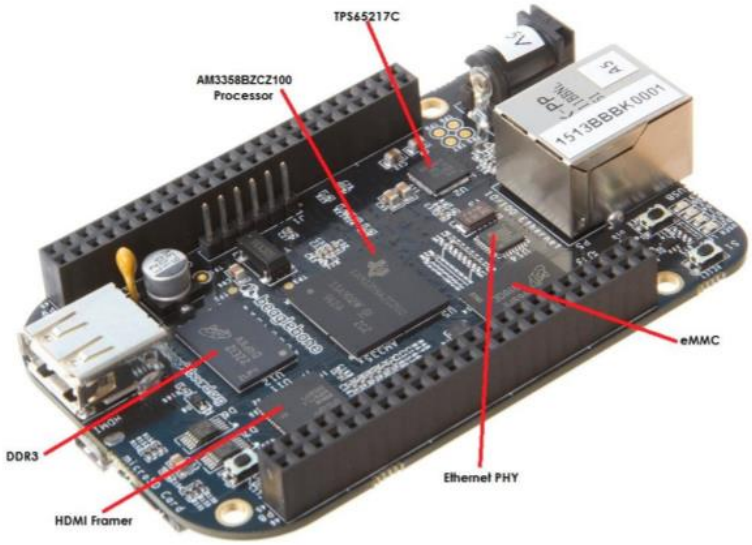

Irudia 6: BBB-ko USR LED-ak.

- Reset: prozesadorea berrabiarazteko botoia.
- Boot Switch: Elikadura plakan egiten bada botoi hau erabiliz microSD txarteletik abiaraztea behartu daiteke elikadura kenduz eta plakari elikadura aplikatuz berriz.
- MicroSD: microSD txartela instalatzeko duen zirrikitua.

- MicroHDMI: Pantaila konektatzeko erabiltzen da.
- USB host: USB interfaze ezberdinak konektatu daitezke. Adibidez, teklatua eta arratoia.
- Debug Serial Header: Debug Serial portua.
- Battery connections: bateria ezarri nahi bada konektatzeko.

#### 2.3.3.2. Osagai nagusiak

- Sitara AM3358BZCZA100: plakaren prozesadorea.
- TPS65217C: Tarjetako hainbat osagaientzako energia bideak hornitzen ditu.
- eMMC: MMC txip integratua da, non 4GB-ko datu kapazitatea duen.
- Ethernet PHY: sareko interfaze fisikoa.
- HDMI Framer: HDMI pantaila egokitzaile baten bitartez kontrolatzeko balio du.
- DDR3: Dual Data Rate RAM memoria.




Irudia 7: BBB-ko osagai nagusiak.

## 2.4. EtherCAT

### 2.4.1. Sarrera

**EtherCAT, Ethernet for Control of Automation Technology:** ("EtherCAT, Automatizazio Teknologien Kontrolerako Ethernet-a") errendimendu handiko eta kode irekiko protokolo informatiko bat da; non Ethernet protokoloak erabiltzeko asmoa duen ingurune industrial batean. Informazioa bidali ahala, EtherCAT-ek prozesatu egiten du eta honi esker gaur egun komunikazioko sistemarik azkarrena da. EtherCAT-eko komunikabideek ondo egokitzen dira industria edo kontroleko inguruneetarako, etengailu

edo etengailu gabe funtzionatu dezaketelako. EtherCAT estandar irekia da, eta IEC espezifikazioak betetzen dituen teknologia bat da.

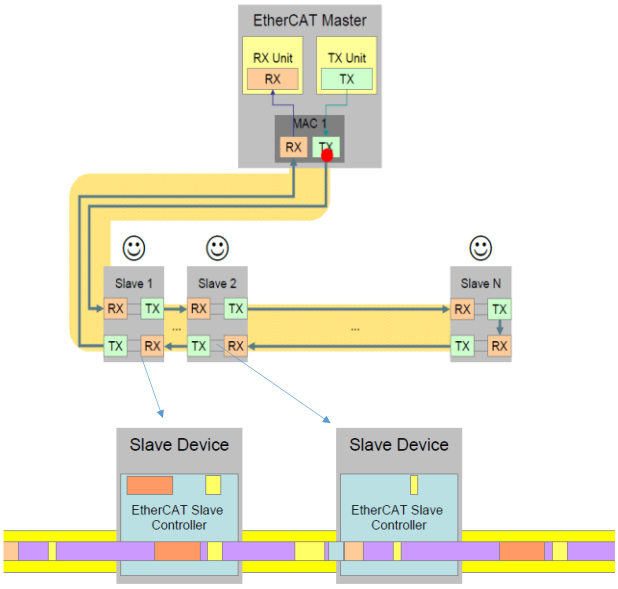

Irudia 8: EtherCAT logoa.

EtherCAT teknologiak Ethernet-eko beste soluzio batzuen sistema mugak gainditzen ditu: Ethernet paketea ez da jasotzen; orduan, konexio bakoitzean datuak prozesatzeko moduan interpretatu eta kopiatzen dira. Horren ordez, Ethernet esparrua prozesatzen da: FMMU berria (memoria kudeaketako eremu unitatea) nodo esklabo bakoitzean bideratutako datuak irakurtzen ditu, telegrama hurrengo gailura bidaltzen den bitartean. Era berean, sarrerako datuak telegramara gehitzen dira bertatik pasatzen den bitartean. Telegramak nanosegundo gutxi batzuen atzerapena besterik ez dauka.

### 2.4.2. Funtzionamendua

EtherCAT maisuak nodo bakoitzetik pasatzen den telegrama bat bidaltzen du. EtherCAT esklabu dispositibo bakoitzak berari bidalitako datuak irakurtzen ditu eta bere datuak txertatzen ditu markoan (Irudia 9). Marko hori hardwarearen hedapenaren atzerapen denboragatik atzeratu daiteke soilik. Segmentuan azken nodoak portu ireki bat antzematean nodo nagusiari mezu bat bidaltzen dio erabateko duplex funtzioak dituen Ethernet teknologiaren bitartez.

Telegramaren gehienezko eraginkortasuna duen transmisio tasa %90a baino gehiago hazten da eta erabateko duplex funtzioaren ondorioz, teorikoki eraginkorra den datuen tasa 100Mbit/s baino handiago da.



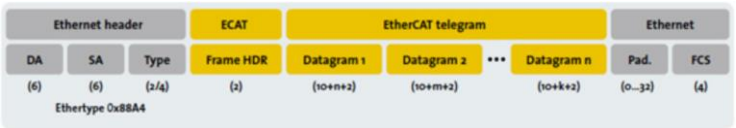

Irudia 9: EtherCAT komunikazio begizta eta prozesua.

EtherCAT-eko maisua da EtherCAT-eko marko baten barruan aktiboki nodo bat bidaltzeko gai den segmentu bakarra. Beste nodo guztiak markoak behera bidaltzeko mugatuak dira. Kontzeptu honek ezusteko atzerapena saihesten du eta denbora errealeko gaitasunak bermatzen ditu.
Maisuak Ethernet Media Access Controller (MAC) estandarreko sarbide kontroladorea erabiltzen du komunikazio osagarririk gabeko prozesadorerik gabe. Honek, maisu bat eskura dagoen Ethernet portu batekin inplementatzen baimentzen du edozein hardware plataforman, edozein sistema eragile denbora errealean edota edozein software aplikazio erabiltzen delarik.

EtherCAT-eko esklabu dispositiboek EtherCAT Slave Controller (ESC) erabiltzen dute fotogramak unean eta erabat hardwarean prozesatzeko. Honek, sarearen errendimendua aurreikusitakoa eta independentea eginez esklabu indibidual dispositiboaren inplementazioarekin.

### 2.4.3. Protokoloa

EtherCAT-ek bere karga baliagarria Ethernet estandarreko marko batean sartzen du (Irudia 10). Markoa (0x88A4) identifikatzailearekin identifikatzen da EtherType eremuan. EtherCAT protokoloa prozesu ziklikoko datu laburrez optimizatua dago; orduan, protokolo pilen erabilpena ezabatu daiteke, esaterako TCP/IP edo UDP/IP.




Irudia 10: EtherCAT, Ethernet-eko marko estandarrean (IEEE 802.3-ren arabera).

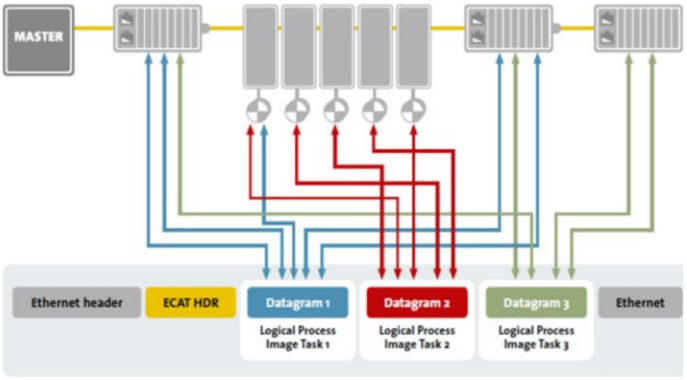
Nodoen arteko Ethernet IT komunikazioa ziurtatzeko, TCP/IP konexioak buzoi kanal baten bitartez garraiatuak izan daitezke denbora errealeko datuen transferentziari eraginik izan gabe.

Abiaraztean, maisu dispositiboak esklabu dispositiboetan konfiguratu eta mapeatu egiten ditu prozesuko datuak. Esklabu bakoitzarekin datu kantitate ezberdinak trukatu daitezke. Bit batetik bit gutxi batzuetara edota kilobyteko datuetara.

EtherCAT-eko markoak datagrama bat edo gehiago ditu. Datagramaren goiburuak adierazten du zein motako sarbidea exekutatu nahi duen maisu dispositiboak.

-Irakurri, idatzi eta irakurri-idatzi.
-Esklabu dispositibo jakin batean zuzeneko helbideratze baten bitartez sartzea edo hainbat esklabu dispositiboetan logikoko helbideratze baten bitartez sartzea.

Logikoko helbideratzea prozesuko datuen truke ziklikoa egiteko erabiltzen da. Datagrama bakoitza, EtherCAT-eko segmentuan prozesuaren irudiko parte espezifiko batera zuzentzen da; non 4GBytes-eko helbide-espazioa erabilgarri dagoen. Sarearen hasieran zehar, esklabu dispositibo bakoitzari helbide-espazio global honetan helbide bat edo gehiago esleituko zaio. Hainbat esklabu dispositibori eremu berdinean helbideak esleitzen badira, denak datagrama batekin bideratzea nahikoa izango litzateke. Datagramek datu-sarbideekin erlazionatutako informazio guztia dutenez, maisu dispositiboak noiz eta zein datu sartu ditzakeen erabaki dezake (Irudia 11).




Irudia 11: Datuak idatzi eta irakurri mezua bidaiatzen ari demean.

Datu ziklikoez gain, komunikazio asinkronoetarako edo ekitaldietan oinarrituriko datagrama gehigarriak ere erabil daitezke. Gainera, logikoko helbideratzeaz gain, maisu dispositiboak esklabu dispositiboari sareko posizioaren bitartez ere zuzentzeko aukera dauka. Metodo hau, sarearen hasieraren zehar erabiltzen da sarearen topologia zehazteko eta aurreikusitakoarekin konparatzeko.

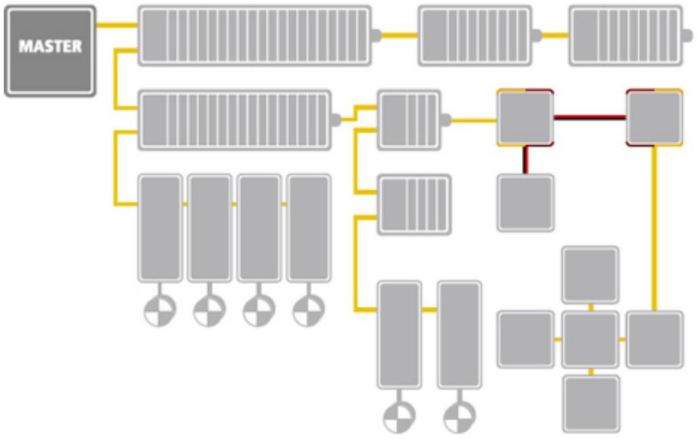
Sare konfigurazioa egiaztatu ondoren, maisu dispositiboak nodo bakoitzari nodo konfiguratutako helbidea izendatu diezaioke eta nodo horrekin helbide finko horren bitartez komunikatu daiteke. Honek dispositiboei aukeratze-sarbidea baimentzen die, nahiz eta sarearen topologia funtzionamenduan zehar aldatu.

Bi ikuspegi ezberdin daude esklabu-esklabu komunikaziorako. Esklabu dispositibo batek sarean beherago konektatuta dagoen esklabu dispositibo bati zuzenean datuak bidaltzeko aukera dauka. EtherCAT-eko tramak etorkizunean bakarrik prozesatzen direnez,

zuzeneko-komunikazio mota hau sarearen topologiaren araberakoa da eta bereziki makina konstante baten diseinuan esklabu-esklabu komunikaziorako egokia da. Aitzitik, libreki konfiguratzekoa den esklabu-esklabu komunikazioak maisu dispositiboa zeharkatzen du eta bi bus ziklo behar ditu.

### 2.4.4. Topologia

Line, tree, star or daisy-chain: EtherCAT-ek ia topologia guztiak onartzen ditu (Irudia 12). EthherCAT-ek kommutadoreak (switch) edo kontzentradoreak (hub) kaskadan egotetik sortzen diren mugaketak normalean alde batera utziz, bus topologia edo ehundaka nodo posible dituen lerroa egiten du.




Irudia 12: Topologi malgua. Line, tree, star.

Sistema kableatzean, lerroekin erorketa-adarrak edo erorketa-lerroak konbinatzean onuragarria da: Adarrak sortzeko behar diren beharrezko portuak S/I-ko modulu askotan integratuta daude. Beraz, kommutadore gehigarriak (switch) eta azpiegitura aktiboko osagaiak ez dira beharrezkoak.

Modular edo erremienta aldatzaileko makinak, sareko segmentuak edo banakako nodoak funtzionamenduan zehar konektatzea eta deskonektatzea behar dute. EtherCAT-eko esklabu kontrolagailuek “Hot Connect” funtzioarentzako oinarria barneratua dute. Bizilagun-geltokia ezabatzen bada, portua automatikoki itxiko da gainerako sareak interferentziarik gabe funtzionatzen jarrai dezan. Detektatze denborak oso motzak <15µs, aldaketa leun bat bermatzen du.

Malgutasun osagarria eskaintzen da kable mota posible guztiei buruz. Ethernet kable industriala erabili daiteke 100m-rainoko distantzia duten bi nodoen artean 100BASE-TX moduan. Power over EtherCAT (IEEE 802.3af-arekin bateragarri) opzioak, dispositiboen konexioa egiten uzten du lerro bakar bateko sentsorea izango balitz bezala. Gainera, zuntz optikoak (100BASE-FX bezala) erabili daitezke, 100m-ko baino handiagoa den nodo distantziarentzat adibidez.

EtherCAT-en 65.535 dispositibo konektatu daitezke, orduan sarearen espantsioa birtualki mugagabea izango da. Ethernet-en ohikoa den bezala, kapa fisikoen artean hautazko aldaketak onartzen dira.

## 2.5. PROFINET

### 2.5.1. Sarrera

PROFINET, Ethernet-eko PROFIBUS & PROFINET International (PI)-ean oinarritutako automatizazio estandarra da. PROFINET, automatizazio industrialaren eta enpresetako sareen eta kudeaketa korporatiboko informazioen plataforma teknologikoa bateratzeko prozesu baten alde egiteko garatu zen. PROFINET S/I-ko kontrolagailuen eta S/I-ko dispositiboen arteko datuen trukaketarako erabiltzen da. S/I PROFINET-ek PROFIBUS DP-aren probaturiko komunikazio eredua eta aplikazioaren ikuspegia erabiltzen ditu eta Ethernet bidez komunikazio euskarri bezala hedatzen da.

Irudia 13: PROFINET logoa.

### 2.5.2. Komunikazio ereduak

PROFINET-ek prestazio ezberdinetako hiru komunikazio eredu eskaintzen ditu:

- Denbora kritikoa ez den aplikazioetarako TCP/IP eta DCOM ereduetarako.
- Denbora erreala (SRT, Soft Real Time), denbora automatizatuaren aplikazio tipikoetarako. (10 ms-ko ziklo denbora):

Denbora errealeko funtzionalitatea denbora kritikoa den prozesu datuetarako erabiltzen da; hau da, erabiltzaile ziklikoen datuetarako edo gertaeren bidez kontrolaturiko alarmetarako. PROFINET-ek automatizazio prozesuen denbora errealeko beharrak optimizatzeko denbora errealean dagoen komunikazio kanal bat erabiltzen du.

Modu honetan, ziklo denborak txikitu egiten dira eta prozesuko datuak eguneratzean errendimendua hobetzen da (Irudia 14). Prestazioak eremu busekin alderagarriak dira eta 1 eta 10ms arteko erantzun denborak onartzen dira. Aldi berean, komunikaziorako behar

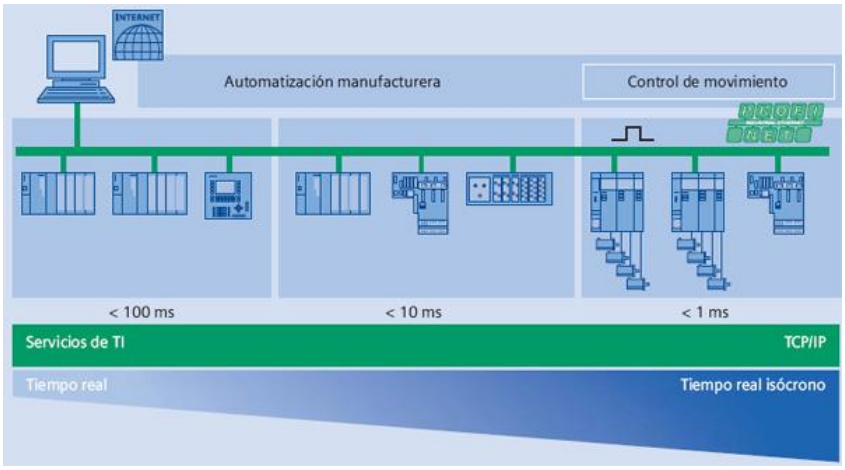
den gailuaren prozesadore potentzia nabarmen murrizten da. Soluzio moduan, sare estandarreko osagaiak erabil daitezke.

- Denbora erreal isokronoa (IRT, Isochronous Real-Time), mugimenduaren kontroleko aplikazioetarako. (1ms-ko zikloetarako):

Denbora erreal isokronoa, mugimenduaren kontrola eta automatizazio manufakturako errendimendu altuko aplikazio motako aplikazio zorrotzetarako erabilgarria da. IRT-arekin 1ms baino txikiagoko ziklo denbora lortzen da, 1µs baino txikiagoko gorabeherak izanik. Horretarako, komunikazio zikloa bi zatitan banatzen da: zati irekia eta zati determinista. IRT zikliko telegramak kanal deterministatik transmititzen dira; RT telegramak eta TCP/IP-ak ordea, kanal irekitik. Orduan, bi datu transmisio mota hauek elkarren artean oztopatu gabe batera existitzen dira. Esate baterako, erabiltzaileek ordenagailu portatil bat konektatu dezakete instalazioko edozein kokapenean gailuetako datuetara sartzeko kontrol isokronoan eraginik izan gabe.

Eremu busen sistemak bi forma ezberdinetan integratu daitezke:

- Eremu busen unitateen integrazioa Proxy-ren bidez: Eremu unitate bakoitzak PROFINET-eko osagai bat irudikatzen du, non bere komunikazioa beste osagai batzuekin konfiguratzen da PROFINET-eko konexio editorearen bitartez. Kasu honetan, Proxy-k Ethernet komunikazioko eremu unitate guztiak irudikatzen ditu.
- Eremu busen aplikazioen integrazioa: Eremu bus baten segmentuak PROFINET-eko osagai bat irudikatzen du; non horren Proxy-ak (adibidez, kontrol bat) PROFINET interfaze bat duen barne. Horrela, menpeko eremu-busaren funtzio guztiak eskuragarri daude Etherneteko osagaia izango balitz bezala.




Irudia 14: PROFINET busaren komunikazio denborak.

### 2.5.3. PROFINET I/O

PROFIBUS-en bezala, PROFINET-eko sarrera eta irteeren irudikapenak eremu unitateen deskribapenera sartzeko ahalbidetzen du; gainera, PROFINET-en banatzen diren periferiak ere integratzen dira.

Integrazio honen funtzio nagusia PLC erabiltzailearen programak eremu banatuetako unitateen sarrera eta irteeren datuak prozesatzen ditu.
PROFINET I/O-k protokolo elementuak eskaintzen ditu honako funtzioetarako:

- Datu produktiboen transmisio ziklikoa.
- Alarmen transmisio aziklikoa.
- Prozesuaren eta diagnostikoen datuen transmisio aziklikoa.

PROFINET I/O-ren definizioa gailuen modeloetako IEC 61158 arauan oinarrituta dago. Espezifikazio honek baldintza hauek onartzen ditu: PROFIBUS DP gailu garaikidetik (masterra edo esklabua) PROFINET I/O gailura (I/O kontrolagailua edo I/O gailua) bihurketa erraza eta, ahal den guztietan I/O gailuen ikuspegi bera mantentzen saiatuz gaurkotasunean PROFIBUS DP gailu esklaboekin gertatzen den bezala (ikuspegi teknikotik, HMI, erabiltzaile-programa, OPC zerbitzaria...).
Komunikazioa denbora errealean:

Denbora erreal sinkronoa (SRT). 10ms baino gutxiagoko ziklo denboratan automatizazioko denbora errealaren eskaerei erantzuteko, PROFINET-eko 2 bertsioak Ethernet-en oinarritutako denbora errealeko komunikazio optimizatua zehaztu zuen. Soluzio honek komunikazio-pila baten exekuzio-denbora murrizten du eta automatizazio datuen eguneratze abiadurari dagokion errendimendua hobetzen du.

Denbora errealeko irtenbide honek 5ms-raino ahalbidetzen du erantzun denbora. Aurreko irtenbidea abilagoa izatea nahi badugu eta datuen trukaketak isokrono egoeran izan behar badu, hardware berezi bat erabiltzen da; non, kommutazio funtzioak ere onartzen dituen. Ethernet-eko “Normal” komunikazioa posible da ardurako hardware hau erabiltzen ari garen unean. Denbora erreal isokronoa (IRT) eskuragarri dago PROFINET-eko 3. bertsioan. PROFINET-ek horrela mugimenduaren kontrolerako aplikazioetan denbora errealeko eskakizunak zorrotz beteko ditu (150 ardatz, 1 ms ziklo bakoitzeko).

## 2.6. CAN BUS

### 2.6.1. Sarrera

CAN (Controller Area Network) komunikazio protokolo bat da, bus topologia batean oinarrituta, ingurune banatuetan mezuen transmisiorako. CPU(prozesatzeko unitate zentralak) askoren arteko komunikazioen kudeaketarako irtenbide bat ere eskaintzen du.

CAN komunikazio protokoloak ondoko prestazioak eskaintzen ditu:

- Interferentziekiko immunitate handia eskaintzen du, auto-diagnostikatzeko gaitasuna eta datuen akatsak konpontzeko.
- Komunikazio protokolo estandarizatua da, sare komun edo bus bati buruz fabrikatzaile ezberdinen azpisistemak komunikatzeko zeregina erraztu eta aurrezten duena.

- Anfitrioi prozesadoreak komunikazio-karga periferiko adimendun batera banatzen du, beraz anfitrioi prozesadoreak denbora gehiago dauka bere zereginak exekutatzeko.
- Multiplexatutako sarea izanik, kableatua nabarmen murrizten du eta puntuz puntuko konexioak ezabatzen ditu, loturetan izan ezik.

### 2.6.2. Ezaugarriak

CAN, produktore/kontsumitzaile ereduan oinarritzen da, zein datuen komunikazio paradigma edo kontzeptua den eta produktore baten eta hainbat kontsumitzaileen arteko harremana deskribatzen duen. CAN, mezuetara bideratutako protokolo bat da, hau da, trukatzera doan informazioa mezuetan banatzen da, zeini identifikatzaile bat esleitzen zaien eta transmisiorako markoetan kapsulatzen diren. Mezu bakoitzak sarean identifikatzaile bakarra du eta nodoek mezu hau onartu edo ez erabakitzen dute. Bere ezaugarri nagusien artean hauek daude:

- Mezuen lehentasuna.
- Latentzia denboren garantia
- Konfigurazioaren malgutasuna.
- Datuen kontsistentziarekiko sistema sendoa.
- Maisu bat baino gehiagoko sistema.
- Erroreen detekzioa eta seinalizazioa.
- Akatsezko markoen bidaltze automatikoa.
- Sareko nodoen aldi baterako akatsen eta eten iraunkorraren arteko bereizketa, eta nodo akastunen deskonektatze autonomoa.

### 2.6.3. CAN-ean oinarritutako protokoloak

CAN-aren estandarrak lehenengo bi geruzak bakarrik zehazten ditu: geruza fisikoa eta datuen lotura geruza, OSI ereduaren arabera. CANek ez ditu barne hartzen geruza handien zereginak, hala nola helbide elektronikoa, sarbide kontrola, datu-blokeak garraiatzea, fotograma bat baino gehiago, etab. Orduan, protokoloak gainean jartzen diren geruzetan agertzen hasi dira CAN-ean oinarrituta daudenak, batez ere aplikazio geruzetan. Kasu honetan, CANopen protokoloa erabili da, automatizazio industrialean gehiena erabiltzen delako.

### 2.6.4. CANopen

Irudia 15: CANopen logoa.

#### 2.6.4.1. Protokoloa

CANopen-a helbideratze eskema batean, komunikazio protokoloetan eta dispositiboaren perfilaren araberako kapan oinarritzen da. Komunikazio protokoloek, sare kontrol bat, nodoak monitorizatzea eta bere komunikazioa hornitzen dute. Protokolo hau, CAN-ean oinarritzen da; non lotura-kapa eta kapa-fisikoa inplementatzen diren. CANopen-ek CAN-aren gain abantaila handi bat du bere objektuetan oinarritutako komunikazioan. Honek, datuen komunikazioa, akatsen detekzioa, sare kontrola etab. baimentzen du forma errazean CAN-eko markoen abstrakzioei esker. Objektu bakoitzak funtzionalitate ezberdin bat dauka eta dispositiboko objektu guztiak "objektuen hiztegia" izenekoan biltegiratzen dira.

Objektuen hiztegia

CANopen-eko kontzeptu nagusiena objektuen hiztegia da. Hau, objektu-multzo ordenatu bat da; non dispositibo bakoitzaren funtzionalitatea deskribatzen duen forma estandarizatuan eta berezko bus-aren bidez bere konfigurazioa ahalbidetzen du mezuen bitartez (SDO: Service Data Objects).

Objektu horiek beraien funtzionalitatearen arabera sailkatzen dira. Horretarako, objektuak 16bit-eko indize eta 8bit-eko azpiindize batez osatutako biltegiratze direkzio baten bitartez antolatzen dira. Indize bakoitza bere azpiindizearekin sarrera bezala izendatzen da objektuen hiztegian. Objektu bat hainbat sarreraz osatuta egon daiteke.

Maisuak esklabuen objektuen hiztegian sartzeko eskubidea dauka eta bere objektuen balioak aldatzea, soilik dispositiboko objektuen hiztegira sartzeko sarrera baimendurik badago. Maisuak dispositiboko sarreretara irakurketa edo irakurketa/idazketa bezala sartzeko aukera du. Sareko nodo bakoitzak bere objektuen hiztegia dauka; non dispositiboa deskribatzen duten parametro guztiak eta sareko bere portaera gordetzen diren.

Objektuak, euren funtzioaren arabera multzokatzen dira hiztegian. Ondorengo irudian ikus daitekeen bezala.

| Rango de Índices | Nombre del grupo de objetos |
|---|---|
| 0x0001-0x0FFF | Data Types |
| 0x1000-0x1029 | Communication Parameters |
| 0x1200-0x12FF | SDO Parameters |
| 0x1400-0x15FF | Receive PDO Paramameters |
| 0x1600-0x17FF | Receive PDO Mapping |
| 0x1800-0x19FF | Transmit PDO Paramameters |
| 0x1A00-0x1BFF | Transmit PDO Mapping |
| 0x2000-0x9FFF | Manufacturer Specific |
| 0x6000-0x9FFF | Standardized Device Profile |

Taula 3: Objektuen hiztegiaren osaketa.

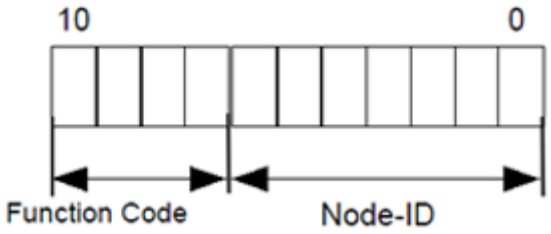
## Mezuen identifikazioa

CANopen protokoloak mezu bakoitzarentzat identifikatzaile bat sortuko du NMT, SYNC eta "Time stamp" mezuentzat soilik. CANopen sare baten konfigurazio denbora murrizteko 11bit-eko esleipen-identifikatzailea duen aurretik definitutako eskema bat dago mezuentzat, "Predefined Connection Set" izenez ezagutzen dena. Eskema honek hurrengo identifikatzaileak zehazten ditu:

- Nodo bakoitzeko emergentzia mezua.
- Sinkronizazio mezua.
- "Time stamp" mezua.
- SDO (bi identifikatzaile hartuz)
- NMT mezuak
- Transmisioko 4 PDO eta hartzeko 4 PDO dispositibo bakoitzeko.

11 biteko identifikatzailea COBID bezala ezagutzen da eta bi zatietan banatzen da:

- 4 bit funtzio-kodearentzat. Kode hauek 4. taulan agertzen dira.
- 7 bit nodo identifikatzailerako (Node-ID).




Irudia 16: CANopen-aren mezu identifikatzailearen egitura.

Identifikatzaileen banaketa maisu-esklabu motako egiturari dagokio. Maisuak konektatuta dauden esklabu guztien ID identifikatzaileak (127 gehienez) ezagutzen ditu. Aldera, esklabuek ezin dute beraien artean komunikatu beste esklabuen identifikatzaileak ezagutzen ez dituztelako. 4. taulan identifikatzaileen banaketa orokor lehenetsia ikus daiteke mezu motaren arabera, non aurkitzen den objektua objektuen hiztegiaren indizeari erreparatuz. COBID txikiena duenak, lehentasun handiena duen mezuak izango ditu.

| Objeto | Código de Función | COBID | Índice |
|---|---|---|---|
| NMT | 0000 | 0x00 | --- |
| SYNC | 0001 | 0x80 | 0x1005h, 0x1006h, 0x1007h |
| TYME STAMP | 0010 | 0x100 | 0x1012, 0x1013 |
| EMERGENCY | 0001 | 0x81-0xFF | 0x1014, 0x1015 |
| PDO1 (tx) | 0011 | 0x181-0x1FF | 0x1800 |
| PDO1(rx) | 0100 | 0x201-0x27F | 0x1400 |
| PDO2 (tx) | 0101 | 0x281-0x2FF | 0x1801 |
| PDO2(rx) | 0110 | 0x301-0x37F | 0x1401 |
| PDO3 (tx) | 0111 | 0x381-0x3FF | 0x1802 |
| PDO3(rx) | 1000 | 0x401-0x47F | 0x1402 |
| PDO4 (tx) | 1001 | 0x481-0x4FF | 0x1803 |
| PDO4(rx) | 1010 | 0x501-0x57F | 0x1403 |
| SDO (tx) | 1011 | 0x581-0x5FF | 0x1200 |
| SDO(rx) | 1100 | 0x601-0x67F | 0x1200 |
| NMT Control de error | 1110 | 0x701-0x77F | 0x1016, 0x1017 |

Taula 4: Mezu mota ezberdinen COBID-ak.

4. taulan ikusi daitekeen bezala, lehentasun handiena duten mezuak NMT objektuaz definitutakoak dira, beraien COBID-a 0x00h izanik. CANopen sare batean 127 da nodo kopuru maximoa; orduan, mezu mota bakoitzak COBID maila ezberdin bat izango du. Adibidez, “PDO1 tansmit”-en (TxPDO1) maila COBID= 0x180 + Node-ID balioarekin hasten da. Hau da, NodeID=0x01 identifikatzailea duen nodoa “COBID=0x181h”-tik NodeID=0x7F=127d identifikatzailea duen nodoa “COBID=0x1FF”-ra arte. NMT, SYNC, eta “Time Stamp” mezuak dira COBID bakarra dutenak.

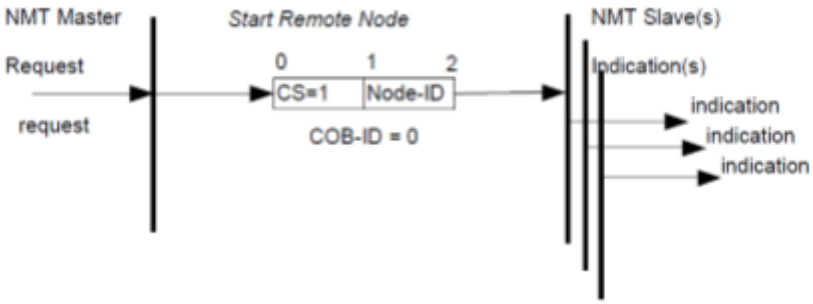
## Protokolo motak

Komunikazioa hainbat mota ezberdinetan banatzen da eta komunikazioaren kudeaketaz arduratuko dira:

## Sarearen kudeaketa. NMT (Network management)

NMT-a komunikazioaren kudeaketaz eta komunikazioko egoera ezberdinak kudeatzeaz arduratzen da. Maisu-esklabu komunikazio ereduarekin egiten da lan; non dispositibo bat NMT maisua den eta beste guztiak NMT esklabuak diren. Esklabu bakoitzak Node-ID izeneko identifikatzaile bakarra du 1 eta 127 tartekoak izanik balio horiek. 0 identifikatzailea maisuaren Node-ID-a da. NMT zerbitzuen bitartez CANopen sareko nodoak aurkitzen diren egoerak aldatu daitezke eta horrekin batera beraien funtzionalitatea.

NMT kontrol mezuak NMT maisuak soilik bidali ditzake, baina esklabu guztiek eman behar die sostengu bat NMT kontrol zerbitzuei. NMT mezu batentzat ez dago erantzunik.




Irudia 17: NMT protokoloa

Irudiko CS-a (command specifier):

- 1: Start.
- 2: Stop.
- 128: Pre-operational.
- 129: Nodoa berrabiarazi.
- 130: Komunikazioa berrabiarazi.

## NMT-ko egoera makina

CANopen sarearen barruan nodoek izan ditzaketen egoera ezberdinak honakoak dira: Initialisation, pre-operational, operational eta stopped. Elikatzen diren momentuan dispositiboak “Initialisation” egoeratik “Pre-operational” egoerara pasako da automatikoki hasieraketa egoera horretan zehar errorerik gertatzen ez bada. Beste egoeren arteko trantsizioak sareko maisuarengatik bakarrik ahal dira ordenatu. Jarraian, egoera ezberdin horiek azaltzen dira:

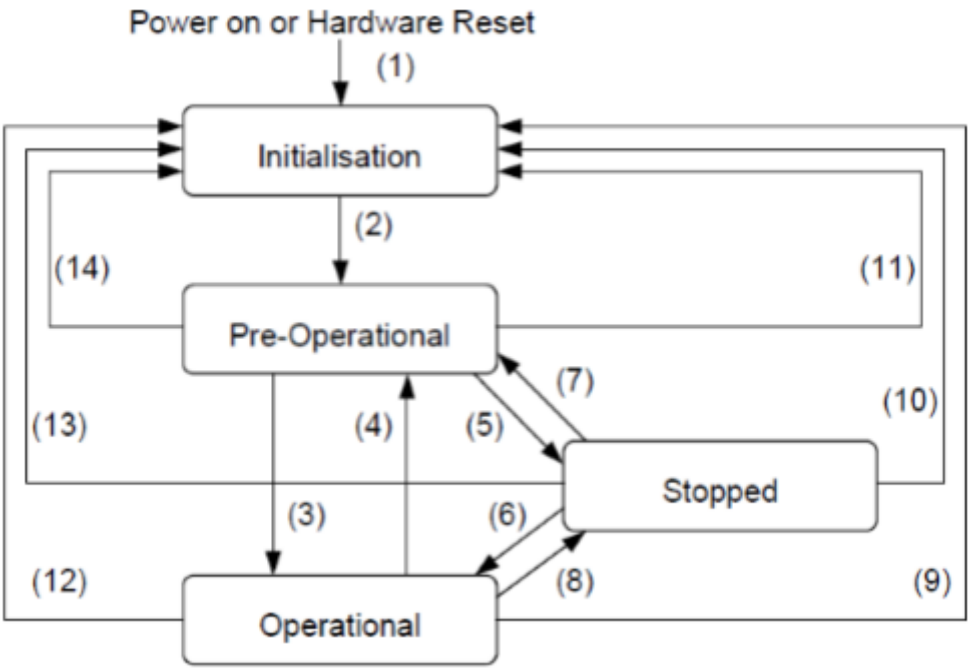
-Initialisation: Dispositiboa pizten zuzenean sartzen da egoera honetara. Hasieraketako zereginak egin ondoren nodoak “Boot-up” mezua transmititzen du eta orduan “Pre-Operational” egoerara pasatzen da.

-Pre-operational: Egoera honetan dispositiboa SDO-en bitartez konfiguratua izan daiteke. SDO-ak, larrialdi mezuak, sinkronizazio mezuak, “time stamp” eta NMT mezuak jaso eta bidali ditzake.

-Operational: Dispositiboa konfiguratua izan da jada eta normalki jarduten du. Komunikazio objektu guztiak martxan daude eta horrela PDO-ak bidali eta jaso ditzake.

-Stopped: Komunikazioko objektu guztiak martxan egoteari uzten die. PDO-ak eta SDO-ak ezin dira ez bidali ez jaso, NMT-ak bakarrik bidali eta jaso daitezke.

Ondorengo irudian, egoera ezberdinen arteko trantsizioak ikus daitezke eta ondorengo taulan, bidaltzeko beharrezkoak diren seinale ezberdinak agertzen dira trantsizio horiek gertatzeko.




Irudia 18: CANopen egoera makina

| Nº Transición | Señal de transición |
|---|---|
| (1) | Alimentación del dispositivo o reinicio del hardware |
| (2) | Inicialización terminada y paso automáticamente al estado pre-operacional |
| (3),(6) | Señal "Start Remote Node" |
| (4),(7) | Señal "Enter_PRE-OPERATIONAL_State" |
| (5),(8) | Señal "Stop_Remote_Node" |
| (9),(10),(11) | Señal Reset_Node |
| (12),(13),(14) | Señal Reset_Communication |

Taula 5: Egoera trantsizioen seinaleak.

## Service Data Objects (SDO)

SDO mezuek dispositiboko objektuen hiztegiko entraden irakurketa eta idazketa egitea uzten dute bere transferentzia eta konfiguraziorako. Transmisio mota guztietako maisua da konfigurazioko datuen transferentziarekin hasten dena, zein esklabuaren hiztegia den erabiltzen ari dena. PDO-ekin alderatuta lehentasun baxuko mezuak dira.

Maisuak indize eta azpiindizeari esker objektuen hiztegiko sarrera kontrolatu dezake, non datuak idatzita edo irakurrita izango diren. SDO mezu bidez eginiko datu trukea, identifikatzaile ezberdinak dituzten bi tramen bidalketan datza. Trama bat maisutik esklabura doa eta bestea, COBID ezberdinarekin baina esklabutik maisura. Garrantzitsua da CANopen-en byte bat baino gehiago dituzten parametroetan lehenengo LSB-a (Less Significative Bit) bidaltzen dela kontutan izatea.

SDO mezuak segmentuen sekuentzia bat bezala transferitzen dira. Hau da, transmititu beharreko datu kopurua handia denez trama ugari bidali behar dira. Trama horiek, 7 byte-tako datuak ere eduki ditzake, segmentu bezala ezagutzen direlarik. Hasieraketa fase bat existitzen da, non maisua eta esklabua segmentuak transmititzeko preparatzen diren bidali nahi den beraien zenbakia adieraziz. Ondoren, transferentzia fasea hasiko da eta erabaki den segmentuen zenbakia transmitituko da. Transmititu behar den datu kopurua 4 byte

baino txikiagoa bada, hasieraketa fasean zehar egitea posible izango litzateke. Kasu hau, transferentzia librea bezala ezagutzen da.

Process Data Objects (PDO)

CANopen sare bateko dispositiboak PDO (Process Data Objects) mezuen bidez ere komunikatu daitezke. Mezu mota honek datuak denbora errealean trukatzea baimentzen du. Datuak, maisu batetik hainbat esklabuetara transmititzen dira.

Nodo edo dispositibo baten PDO mezuak bi mailatan banatzen dira:

- Transmisio mezuak (TxPDO): Nodoak mezuen bitartez transmititzen duen informazioa da.
- Hartze mezuak (RxPDO): Nodoak mezuetan irakurtzen duen informazioa da.

PDO baten edukia, bere identifikatzailearekin bakarrik zehaztuta dago. Bai igorleak bai hartzaileak ezagutu behar dute, bere barne egitura interpretatzeko. PDO bakoitza hiztegiko bi objektuekin deskribatzen da:

- PDO Communication Parameter: PDO-ak erabiltzen duen COBID-a, transmisio mota, inhibizio denbora eta tenporizadorea dauzka.

- PDO Mapping Parameter: objektuen hiztegitik erabiltzen diren PDO mezuko objektuen zerrenda dauka.

PDO-en transmisioa egiteko CANopen-ek hainbat komunikazio mekanismo definitzen ditu:

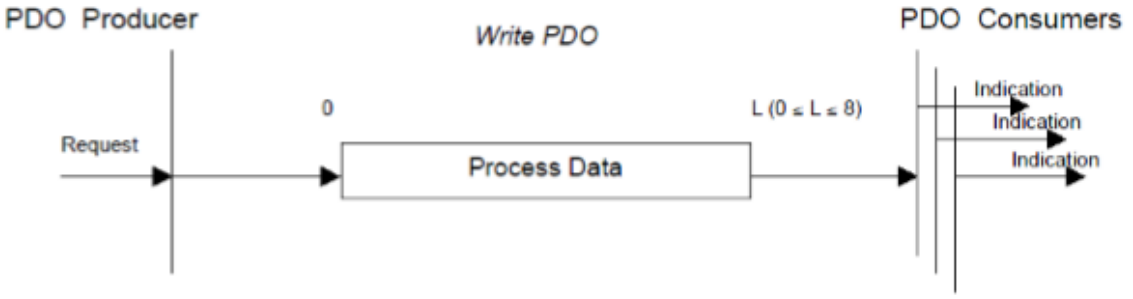
- Transmisio asinkronoa: PDO mezuen bidalketa, gertaera jakin baten agerraldiarekin gertatzen da. Adibidez, urruneko trama batek datuak eskatzen dituenean.

    - Gertaerak (Event Driven): Mezu baten transmisio, dispositiboaren perfilean definiturik dagoen gertaera zehatz baten agerraldiarekin gertatzen da.
    - Tenporizadorea (Timer Driven): behin eta berriz denbora zehatz bat igaro ondoren sortzen du transmisioa.
    - Urruneko aplikazioa (Remotely requested): PDO mezuen transmisio asinkronoak, beste dispositibo batetik bidalitako urruneko aplikazio (trama RTR) bat jasotzean hastea posible du.

- Transmisio sinkronoa: PDO mezuen transmisio sinkronoa, transmisio periodo bat iraungitzean aktibatzen da; non, SYNC objektu batzuk jasotzean sinkronizatzen den. Hau da, SYNC mezu bat iristen den uneoro transmisio sinkronoko leiho bat irekitzen da. PDO sinkronoak leiho horretara bidalduak izan behar dira. Bi transmisio mota ezberdin daude:
    - Zikliko mota: SYNC objektuak ireki duen leihoan zehar transmititzen diren mezuak dira. Ez dira leiho guztietan transmititzen, aldizkakotasun batekin baizik; non, dagokion “Communication Parameter”-eko “Transmission Type” eremuak zehazten duen.
    - Azikliko mota: Aplikazio gertaera bat gertatu ondoren transmititzen diren mezuak dira. Leihoaren barruan transmititzen dira baina ez aldiro.

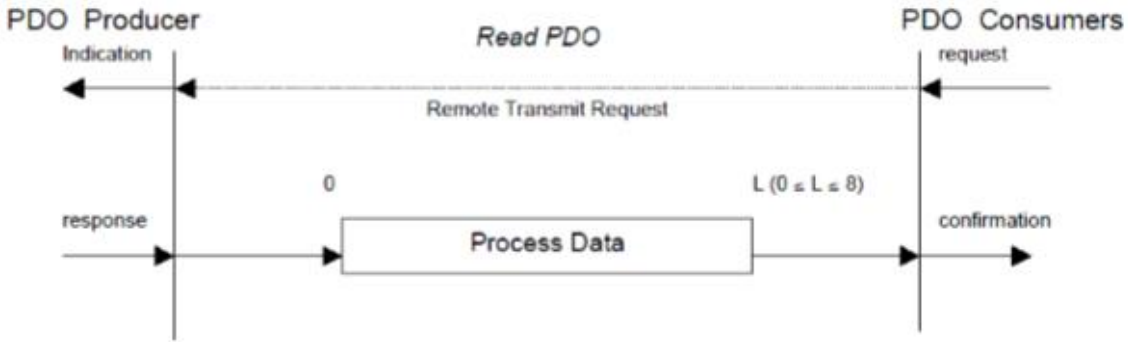
Gainera, PDO-ak idazteko edo irakurtzeko erabili daitezke:

- Write PDO: “Push” komunikazio eredua erabiltzen da. Kontsumitzaileak egon daitezke edo ez eta ekoizle bakarra erabiltzen da PDO mezu bakoitzeko. Zerbitzu honen arabera, ekoizleak jasotako objektuen helbideak PDO mezu bidez bidaltzen dizkio kontsumitzaileari. Ez da berrespena existitzen.




Irudia 19: PDO-en Idazketa protokoloa.

- Read PDO: “Pull” komunikazio eredua erabiltzen da. Gutxienez bat edo bi kontsumitzaile eta ekoizle bakarra erabiltzen da PDO mezu bakoitzeko. Zerbitzu honen arabera, ekoizleak kontsumitzaileak aurretik eskatutako objektuen helbideetan aurkitzen diren datuak bidaltzen ditu urruneko tramaren bitartez. Berrespena existitzen da.



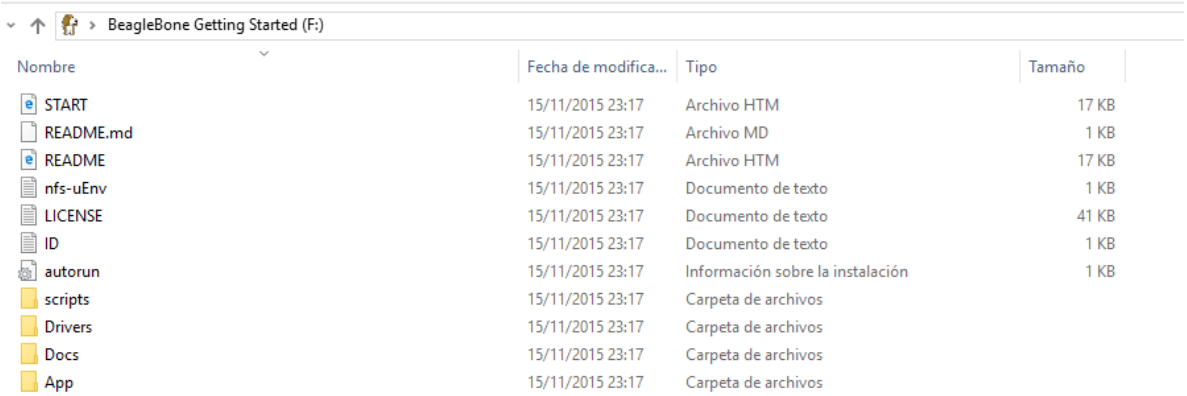

Irudia 20: PDO-en Irakurketa protokoloa.

# 3. PROZESUA

## 3.1. BBB-aren konfigurazioa

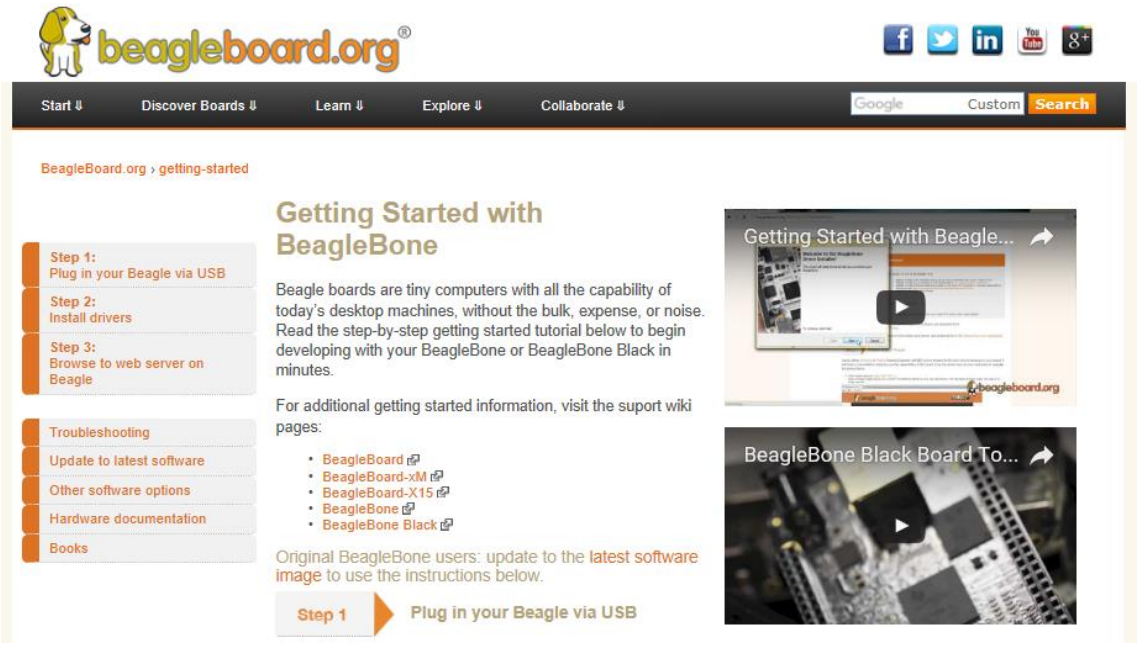
### 3.1.1. BBB-aren abiaraztea

BeagleBone Black-aren konfigurazioarekin hasteko, BBB-a USB bidez ordenagailura konektatu behar da. Konektatzerakoan, ondorengo irudian ikusten den karpeta agertzen da.



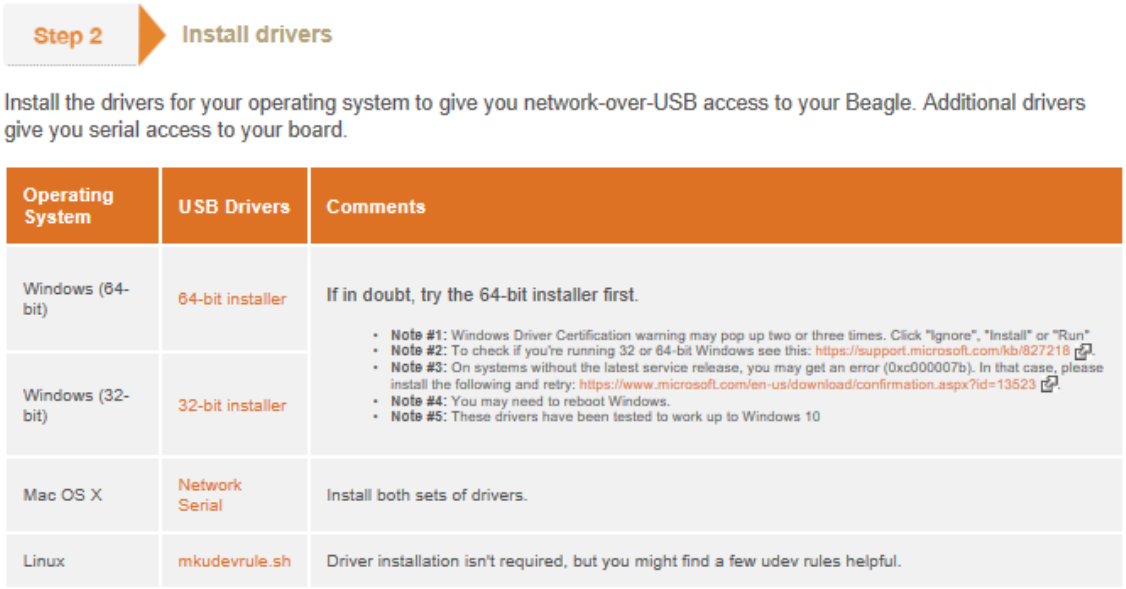

Irudia 21: BBB-ko hasierako fitxategia.

Aurreko irudiko fitxategian, START.htm artxiboa exekutatu behar da eta honela BeagelBoard-aren web orrialdera bideratzen gaitu hurrengo irudian ikusten den bezala.



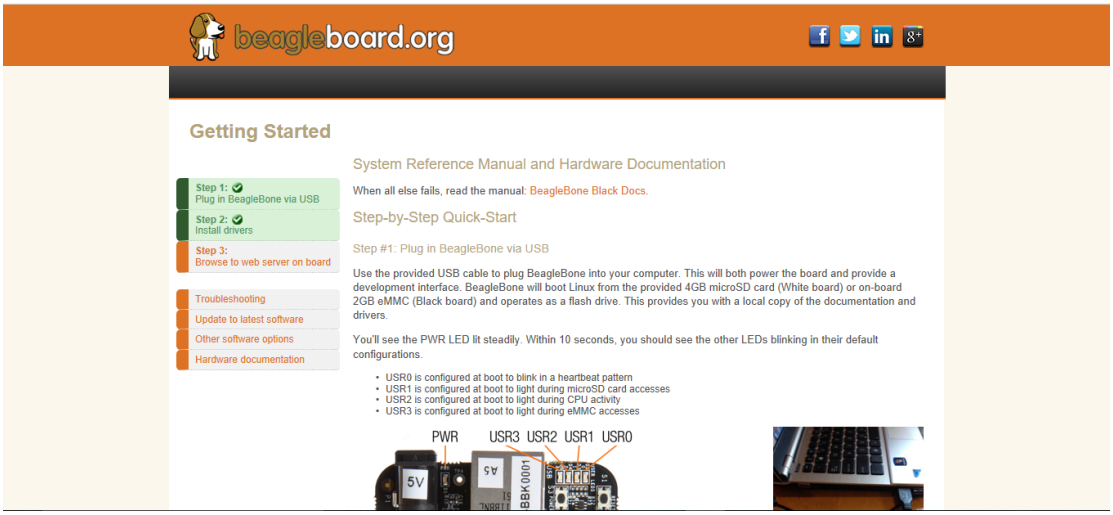

Irudia 22: BeagleBoard-aren web orria.

2. irudian ezkerrean dauden hiru pausoak betez BBB-a instalatuta geratuko da. Lehenengo pausoan, BBB-a konektatzean eta web orriak ezagutzen badu "Step 1" berdea jarriko da esanez bigarren pausoarekin jarraitzea dagoela. Bigarren pausoan, ordenagailuan plakarekin komunikazio seriea egiteko behar dituen driverrak instalatzen dira. Hurrengo irudian, sistema eragile ezberdinetako driverrak agertzen dira.



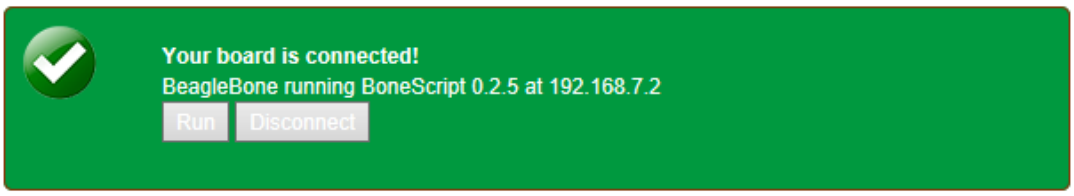

Irudia 23: 2.pausoa. Sistema eragile ezberdinetako driverrak.

Gure kasuan, Windows (64-bit) driverra instalatu da eta bukatutakoan "Step 2" berdez agertzen da hurrengo irudian ikusten den bezala.



Irudia 24: “Step 1” eta “Step 2” modu egokian amaiturik.

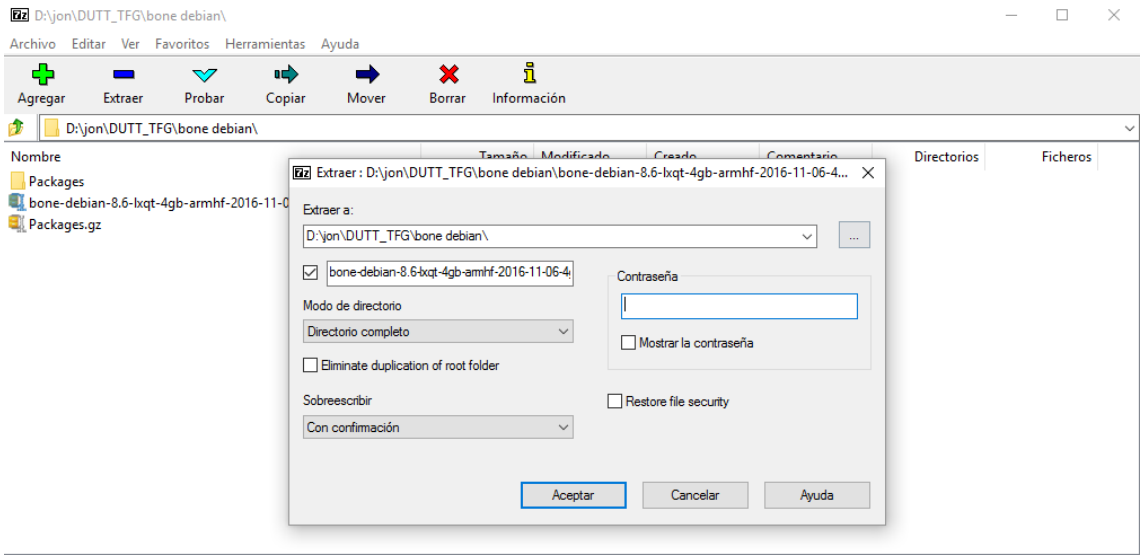
Driverra instalatu ondoren, hirugarren pausua http://192.168.7.2 webera konektatzea izango da. Horrela, konprobatuko dugu BBB-arekin konexioa daukagun edo ez. Helbide hori BBB-aren USB bidezko IP-a da. Konektatuz gero hurrengo irudiko mezua ikusten da.



Irudia 25: “Step 3”. BBB-aren konexioaren mezua.

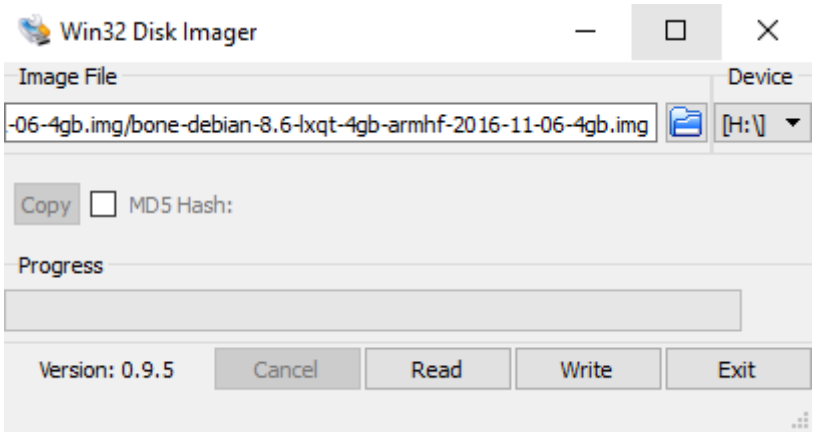
### 3.1.2. BBB-aren sistema eragileak eta flasheatzea

BeagleBone Black hasieran Angstrom Linux sistema eragilearekin garatu zen. Gaur egun, Debian da BBB-arekin erabiltzen den sistema eragilea. Sistema eragileak BBB-arekin baliagarria izateko ARMHF-ekiko bateragarria izan behar du. Hau da, ARM arkitekturako 7. bertsioa edo berriago izan behar du. Horretarako, https://beagleboard.org/latest-images linkera sartu behar da. Bertan, Linuxeko Debian irudiak daude eta berriena aukeratu da. Gure kasuan, Debian 8.6-a.
Deskargatu ondoren, 7zip bidez deskonprimatu da.



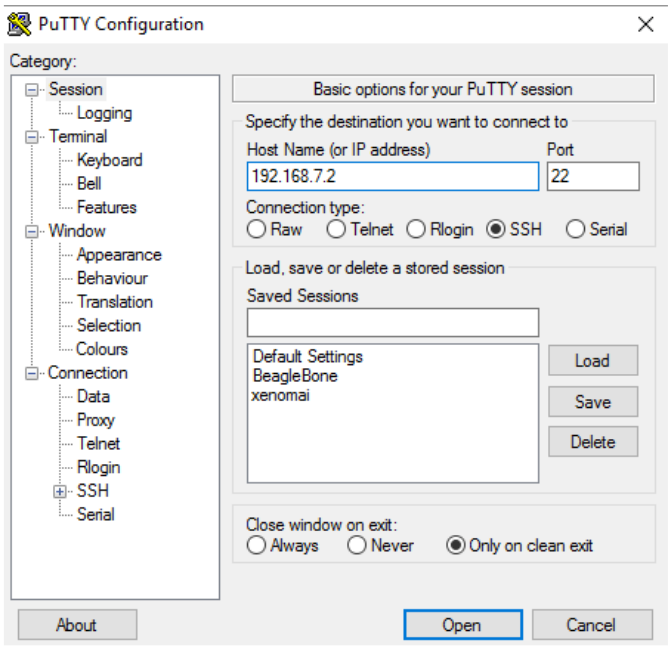

Irudia 26: 7zip programaren bidez Debian irudiaren deskonprimatzea.

Deskonprimatu ondoren, .img irudi bat lortzen da eta hori SD txartelean ezarri da “Win32 Disk Imager” programaren bitartez.



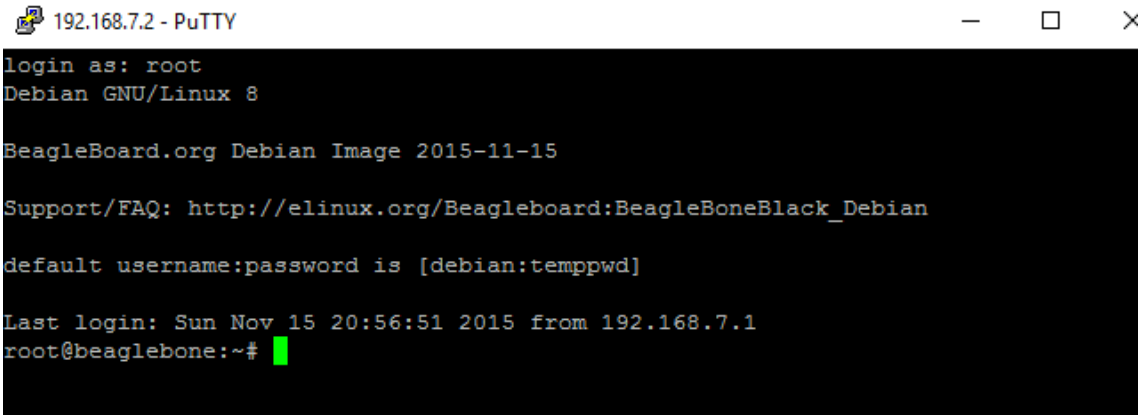

Irudia 27: Win32 Disk Imager. Debian-en irudia SD-an grabatzeko.

Irudia SD-an grabatu eta gero, http://www.putty.org/ web orrian sartuz “putty” aplikazioa deskargatu da. Aplikazio hau, BBB-aren terminal bezala erabiltzeko da. Horretarako, “putty”-ri BBB-aren IPa (192.168.7.2) ezarri behar zaio hurrengo irudian erakusten den bezala.



Irudia 28: Putty-ren konfigurazioa IP zuzena ezarriz.

Flashean irudirik ez bada grabatzen eta SD-an ez badugu irudirik, BBBak flashean fabrikatik dakarren irudia erabiltzen du sistema abiarazteko unean. Hau hurrengo irudian frogatzen da. Ikusten den bezala 2015eko irudia du grabatua.

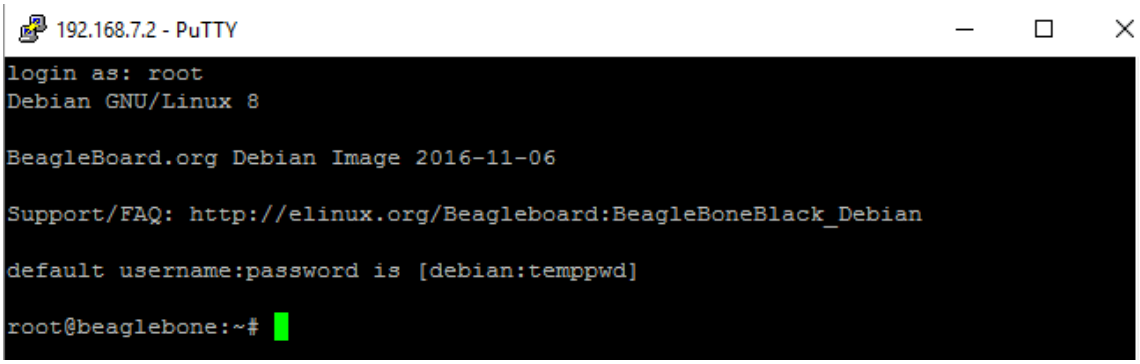


Irudia 29: Putty-ren bidezko BBB-aren terminala. BBB-aren jatorrizko sistema eragilea.

Terminalaren bidez konektatzerakoan BBB-a, “login as:” galdera egiten du. Administradore bezala sartu nahi bada “root” idatzi behar da. Debianeko erabiltzaile bezala sartu nahi bada “debian” idatziko da.
Debianeko erabiltzaile bezala sartuz gero pasahitza behar da komandoak exekutatu ahal izateko. Pasahitza hau “temppwd” da jatorriz. Egoera honetan komandoak exekutatu nahi badira, komando bakoitzaren aurrean “sudo” hitza idatzi behar da. Adibidez “cd” (change directory) komandoa nahi bada exekutatu, “sudo cd” idatzi beharko da.

Aldiz, administradore bezala sartuz gero ez du pasahitzik behar eta exekutatzen den komandoaren aurrean ez da ezer jarri behar. Aurreko adibidean “cd” idatziko genuke.
BBB-a abiaraztean, SD-an sistema eragileren baten irudia dagoen begiratzen du. Irudia aurkitzen badu, BBB-a SD-ko irudiarekin abiarazten da. Ez badu SD-an irudirik aurkitzen

flashean grabatutakoarekin abiatuko da. Adibide bezala, hurrengo irudian ikus daiteke nola BBB-ak guk SD-an sartutako sistema eragilearen irudirik berriena hartu duen.




Irudia 30: BBB-aren SD bidezko irudi berria.

BBB-ari sistema eragilerik berriena ezarri eta gero, sistema eragile horren nukleoa edo kernela aldatu behar da bere denbora erreala hobetze arren. Horretarako, https://rcn-ee.com/repos/debian/pool/main/l/linux-upstream/ linkean sartu behar da eta nahi den kernela aukeratu.
Ondoren, terminalaren bitartez (Putty) "tools" fitxategira joan behar da eta "git pull" komandoa exekutatu, egindako aldaketak kernelean gorde ditzan. Amaitzeko, aukeratutako kernel berria "./update_kernel.sh" komandoaren bitartez guk aurretik aukeratutako linkera joango da nahi dugun kernela bilatzera eta instalatzera. Hau guztia, terminalean ondorengo sekuentziaren bidez idazten da:

```
cd /opt/scripts/tools
git pull
./update_kernel.sh --kernel 4.4.14-bone-rt-r11
```

BBB-ak hainbat "cpufreq Governor" ditu: conservative, ondemand, userspace, powersave eta performance. BBB-a jatorrian "ondemand" konfigurazioarekin dator, non CPU-aren frekuentzia 300MHz eta 1GHz tartean egiten duen lan. Gure aplikazioan garrantzitsua denez denbora errealean lan egitea, "performance" moduan jartzea komeni zaigu; zeren eta 1GHz-ko maiztasun finkoan lan egingo duen. Horretarako, "cpufrequtils" fitxategian sartu behar da eta aipatutako aldaketa egingo da. Hurrengo sekuentzian adierazten den bezala:

```
Edit file /etc/init.d/cpufrequtils

GOVERNOR="performance"
```

Pausu hauek egin eta gero, SD txarteleko sistema eragilearen irudia eta kernela eMMC-an flasheatuko dugu. Horretarako:

```
cd /opt/scripts/tools/eMMC/
./init-eMMC-flasher-v3.sh
```

Orduan, BBB-ko LED-ak kliskatzen hasiko dira eta itzali arte itxaron egin beharko da. Operazio honek 30-45minutu irauten du. Hau egin eta gero SD-a kentzea posible da. Orain, SD gabe konektatzean irudi eta kernel berria izango du BBB-ak. Hori konprobatzeko, "dmesg" (kernela ikusteko) eta "cat /etc/issue" (irudia ikusteko) komandoak erabili behar dira.

## 3.2. BBB-aren erabilpena CODESYS-en bitartez

### 3.2.1. BBB-aren Ethernet konfigurazioa

Ordenagailuko Codesys softwareak BBB-arekin komunikatu egin behar du eta Ethernet bidez konektatu nahi bada, BBB-aren Etherneteko IP helbidea finkatu behar da. Hau terminalaren bidez egiten da ondorengo pausoak jarraituz:

```
nano /etc/network/interfaces
```

Hau egitean fitxategi barruan sartuko da eta ondorengoa ezarri behar da:

```
auto eth0
iface eth0 inet static
address XXX.XXX.XXX.XXX → Adibidez: 192.168.1.69
netmask XXX.XXX.XXX.XXX→ Adibidez: 255.255.255.0
gateway XXX.XXX.XXX.XXX→ Adibidez: 192.168.1.197
```

Ondoren, fitxategia gordetzeko “Ctrl+x” egin behar da eta ondoren baiezkoa emateko “y” zapaldu behar da eta orduan terminalera bueltatuko da.
Jakiteko zein Ethernet portu dauden hurrengo komandoa idatziko da:

```
connmanctl services
```

Gure kasuan, bi aukera hauek agertzen dira:

```
*AR Wired                ethernet_68c90bc37cb4_cable
*AR Wired                ethernet_da8859189f70_cable
```

Irudia 31: Etherneteko MAC helbideak.

Bi Etherneteko MAC helbide ikusten dira. Zein den jakiteko “ifconfig” komandoa idatziko dugu eta hurrengo irudian ikus daiteke gure eth0 helbide fisikoa edo MAC helbidea zein den.

```
eth0      Link encap:Ethernet  HWaddr 68:c9:0b:c3:7c:b4
          UP BROADCAST MULTICAST DYNAMIC  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Interrupt:173
```

Irudia 32: “eth0”-ko informazioa.

Gure helbide fisikoa aukeratuz ondorengoa egin behar da BBB-ari Etherneteko IP-a zehazteko:

```
connmanctl config ethernet_68c90bc37cb4_cable --ipv4 manual 192.168.1.69(gure address-a)
255.255.255.0(gure netmask-a) 192.168.1.197(gure gateway-a) --nameservers 8.8.8.8
```

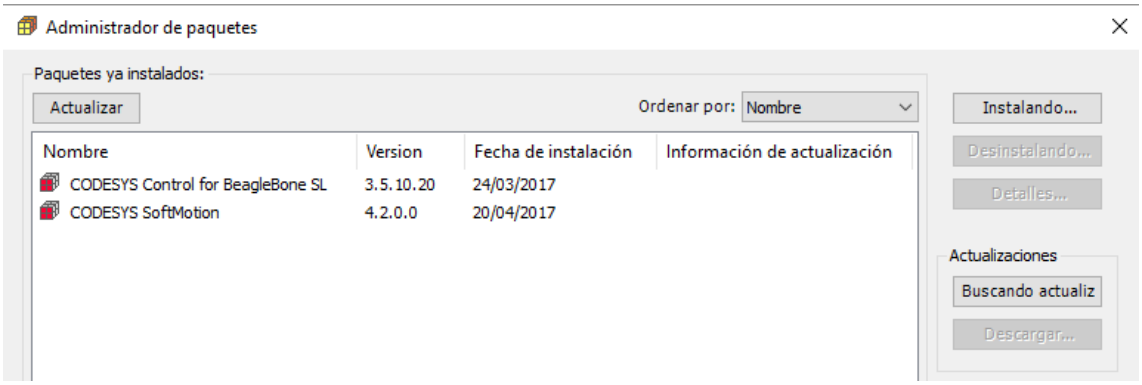
### 3.2.2. CODESYS-en instalazioa

CODESYS instalatzeko, lehenik eta behin “CODESYS Store”-ko web orri ofizialera joan behar da. Bertan, “CODESYS Development System V3” programa deskargatu behar da. Honetaz gain, “CODESYS Control for BeagleBone SL” paketea deskargatu behar da beharrezkoa delako CODESYS BBB-arekin erabili nahi bada. Hurrengo bi irudietan ikus daitekeen bezala, CODESYS programarekin jardutea doan da; baina, BBB-arekin lan egin nahi bada “CODESYS Control for BeagleBone SL” paketea erosi egin behar da.


Irudia 33: CODESYS aplikazioa.

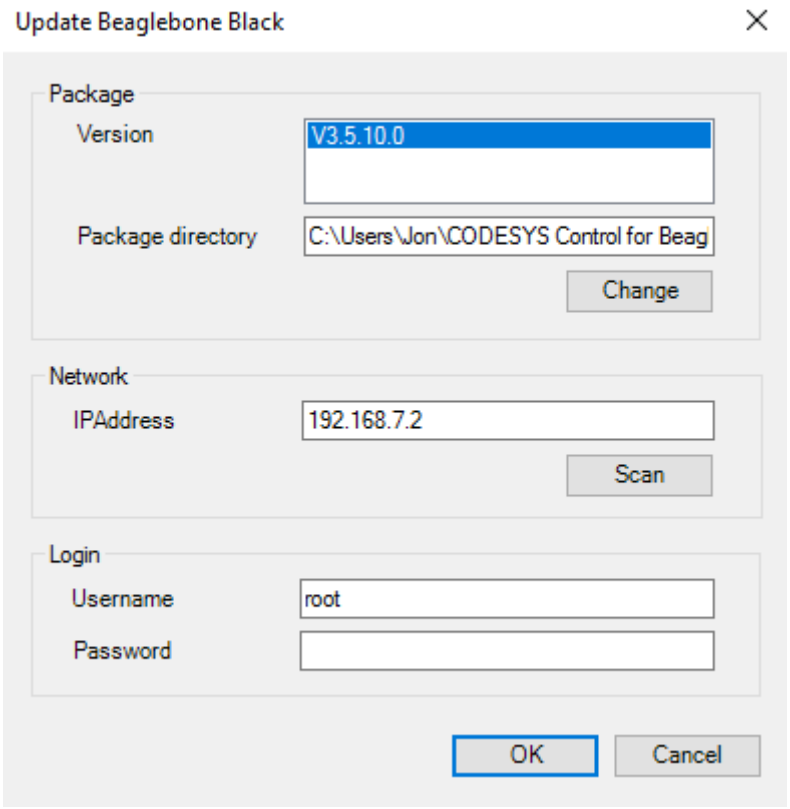

Irudia 34: CODESYS-ekin BBB-a kontrolatzeko paketea.

Biak deskargatu ondoren CODESYS-eko programa instalatuko dugu gure ordenagailuan eta bertan sartuko gara. Lehenengo aldiz sartzerakoan, “Herramientas” fitxa irekitzean “Update BeagleBone Black” ez da agertzen. Hori ager dadin BeagleBone-ren paketea CODESYS-eko programan instalatu behar da. Horretarako, “Herramientras →Administrador de paquetes…”-en sartu behar da. Hurrengo irudian ikusten den leihoa irekiko da.




Irudia 35: CODESYS-eko “Administrador de paquetes” leihoa.

“Instalando…” jartzen duen botoian zapaldu eta paketea aukeratuz instalatu egiten da. Instalatu ondoren CODESYS programa itxi eta berriz sartzean “Update BeagleBone Black” agertuko da “Herramientas” fitxan. Hor sartzean, ordenagailura konektatuta dagoen BBB-ari runtime-a bidaliko dio. Hau egiteko hurrengo irudian ikusten da konfigurazioa.

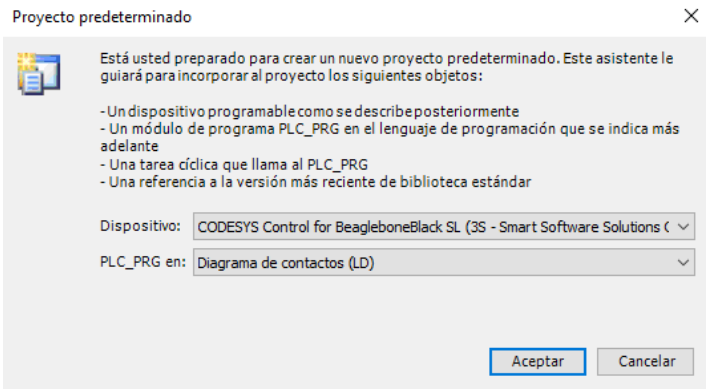



Irudia 36: “Update Beaglebone Black” leihoa.

### 3.2.3. CODESYS eta BBB-aren arteko komunikazioa eta I/O kontrola

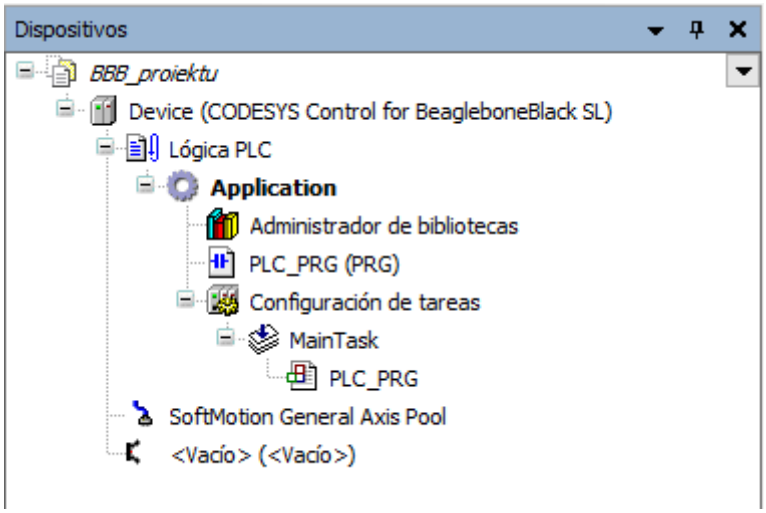
Behin Softwareak instalatuta, proiektu sinple bat egin nahi da. Beraien arteko komunikazioa eta BBB-aren sarrera irteera digitalak kontrolatzeko gai garen ikusi nahi da.

Horretarako, “Nuevo proyecto→Proyecto standard→Nombre eta Ubicación” proiektuaren izena eta helbidea aukeratu eta gero “Aceptar” sakatuko da. Ondoren hurrengo irudia agertuko da. Bertan, “Dispositivo”-n “CODESYS Control for BeagleBone SL” aukeratu behar da. “PLC_PRG en”-en erabili nahi den lengoaia aukeratu behar da. Gure kasuan, “Diagrama de contactos (LD)” aukeratu da.



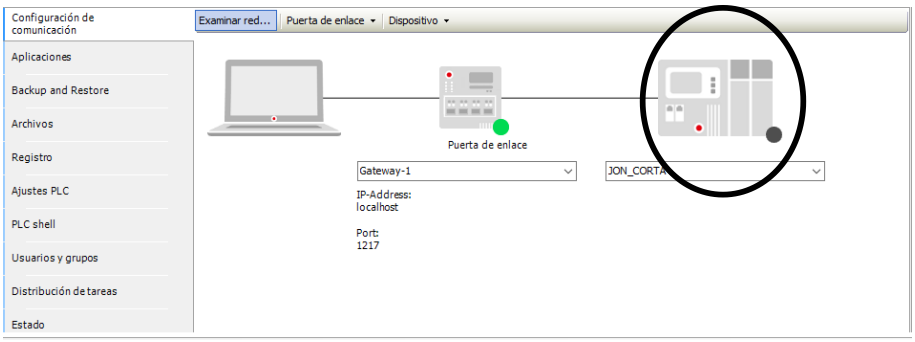

Irudia 37: Proiektu berri bat sortzeko aukeratu behar diren gailua eta programazio lengoaia.

“Aceptar” sakatu ondoren, CODESYS programan “Dispositivos” izeneko leiho bat agertuko da ondorengo irudian ikus daitekeen bezala.



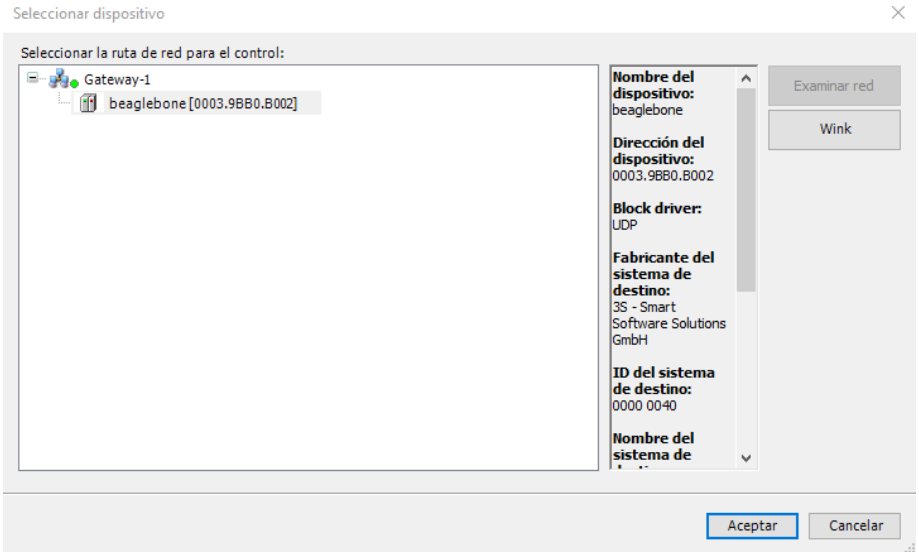

Irudia 38: BBB-aren “Dispositivos” leihoa hasieran.

“Device” bi aldiz klikatu ondoren, hurrengo irudian ikusten den leihoa irekiko da. Irudian ikusi daitekeen bezala dispositiboaren LED-a itzalita dago, komunikaziorik ez dagoela adieraziz. Lortu nahi dena CODESYS BBB-arekin komunikatzea da eta horretarako “Examinar red…” klikatu behar da.



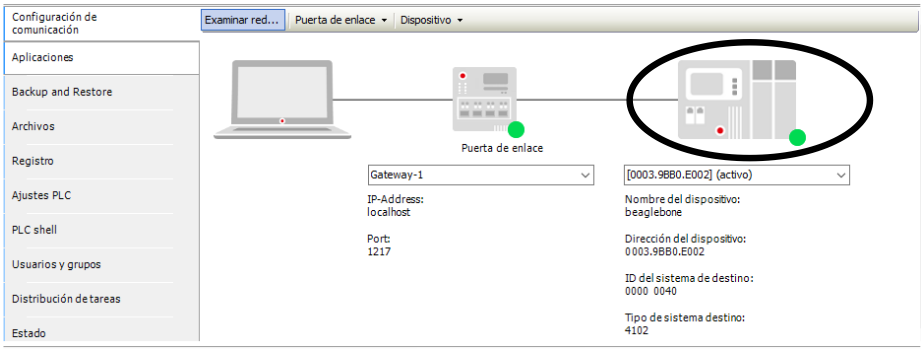

Irudia 39: “Device” leihoa komunikazio sarearen egoera erakutsiz.

BBB-a bilatzen badu, ondorengo irudian “beaglebone” aukeratu eta “Aceptar” sakatu.



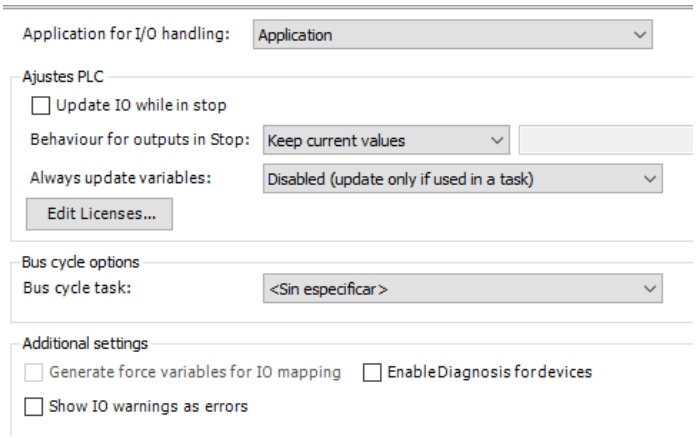

Irudia 40: “Examinar red” sakatu ondoreneko leihoa.

“beaglebone” aukeratu ondoren ikus daiteke konexioa ondo egina dagoela gure dispositiboaren LED-a berdea jarri delako irudian ikusten den bezala.



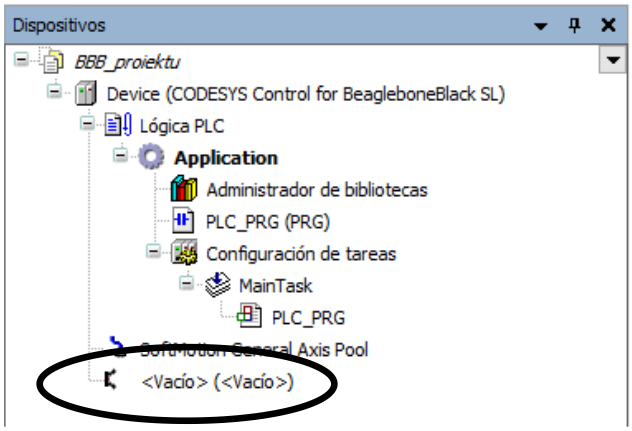

Irudia 41: “Device” leihoan BBB-a aktibatuta.

“Device”-n bertan “Ajustes PLC”-ra joan behar da aldaketa batzuk egitera egin nahi den aplikazioak funtziona dezan. “Always update variables” atalean, “Disabled” ordez “Enabled 2” jarri behar da.



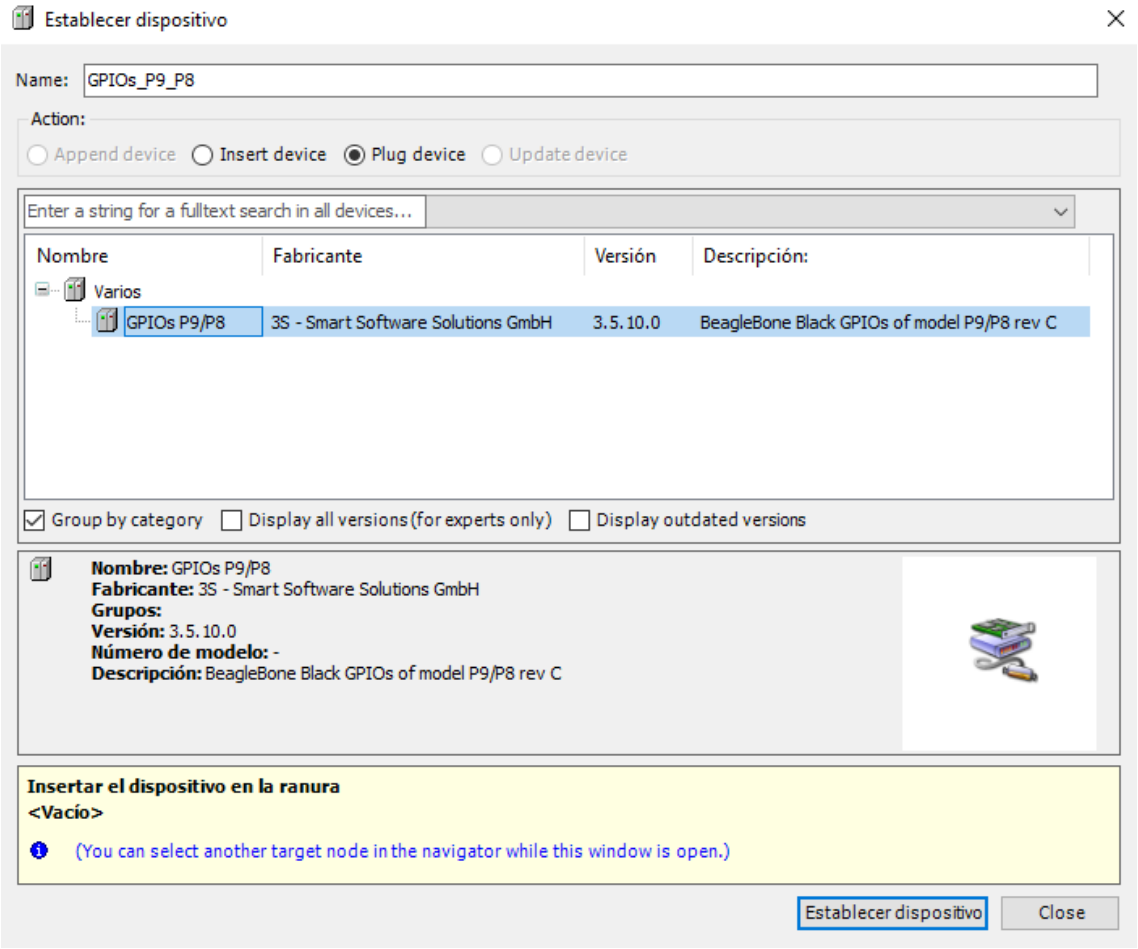

Irudia 42: “Device”-ko “Ajustes PLC”-ko leihoa.

Aldaketak egin eta gero, “Dispositivos” leihora bueltatu behar da, baina orain irudian ikusten den bezala “<vacio>” jartzen duen aukera horretan xaguaren eskuineko botoiarekin sakatuko dugu.



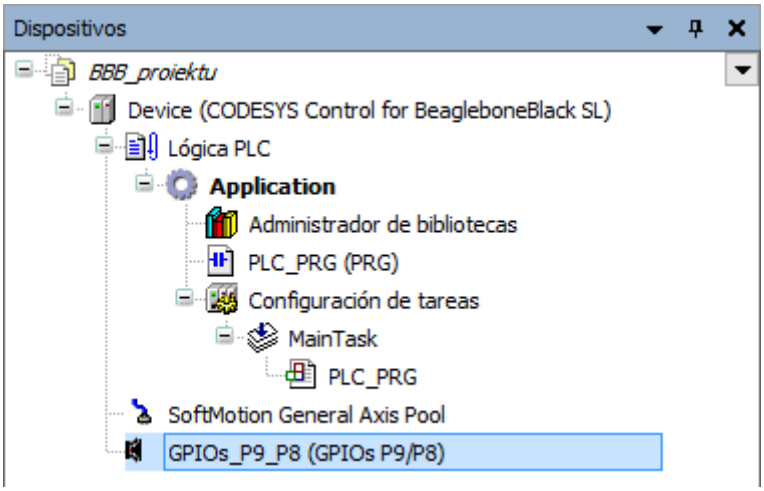

Irudia 43: “Dispositivos” leihoa GPIO-ak konfiguratzeko.

Orduan, “Establecer dispositivo” aukeratuko dugu hurrengo irudia agertuz. Irudian agertzen diren BBB-aren GPIO-ak aukeratuko dira.



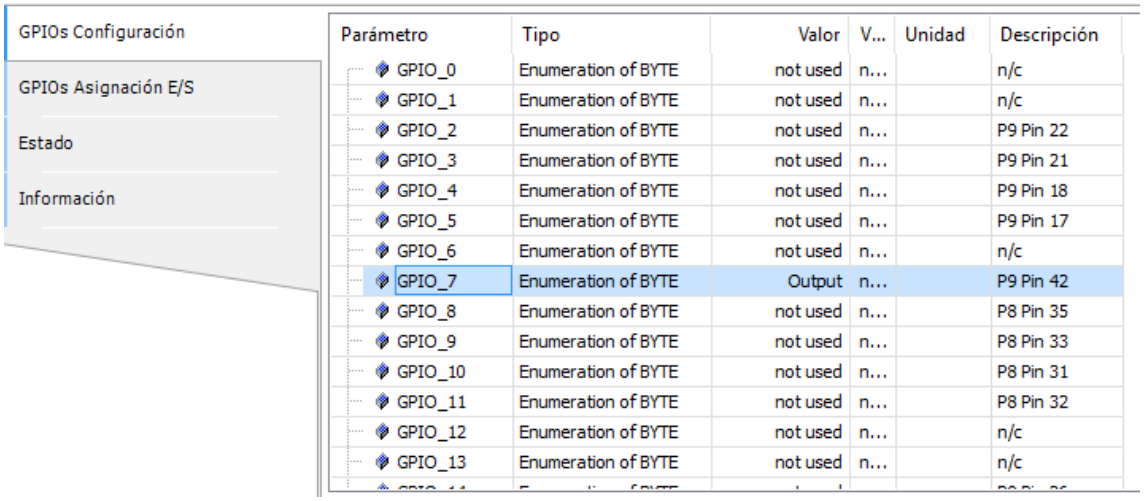

Irudia 44: “Establecer dispositivo” leihoa GPIO-ak aukeratzeko.

Aukeratutakoan, “Establecer dispositivo” klikatu eta aukera hori hurrengo irudian ikusten den bezala “Dispositivos”-eko leihoan agertuko da.



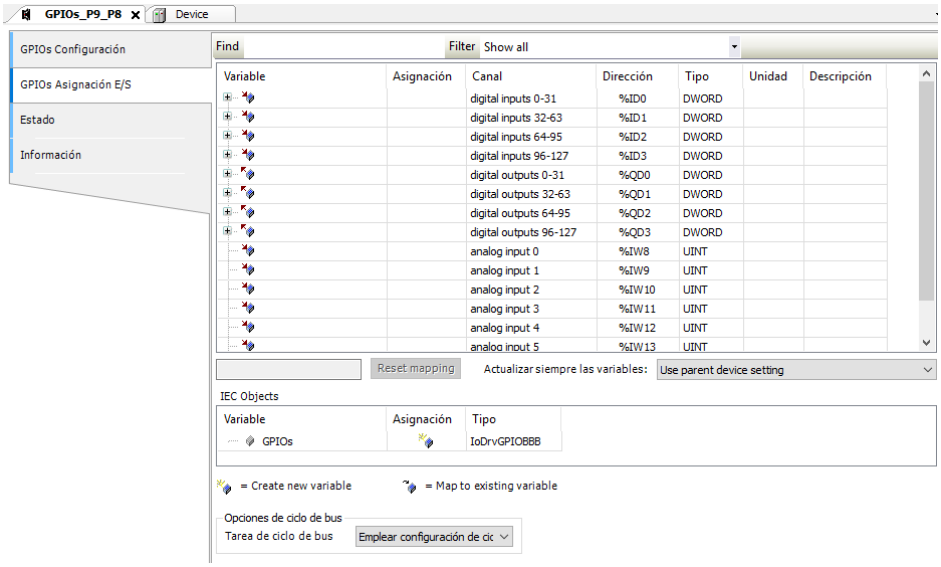

Irudia 45: “Dispositivos”-en “GPIOs_P9_P8” ezarrita.

“GPIOs_P9_P8” atalean sakatuz, hurrengo leihoa irekitzen da. Hemen ikusi daitezke BBB-an erabilgarriak diren pinak eta CODESYS-ekin pin bakoitzak daukan erlazioa.




Irudia 46: “GPIOs_P9_P8”-ko “GPIOs Configuración” leihoa.

Proba egiteko BBB-ko “P9 Pin 42”-a aukeratu da eta irteera moduan jarri da. Orduan, pin horren parametroa GPIO_7 izango da. Parametro horri, izena “GPIOs Asignación E/S”-ean eman behar zaio (Irudia 47). Irteera bezala ezarri dugunez “digital outputs 0-31” aukeratu beharko litzateke tarte horretan dagoelako.



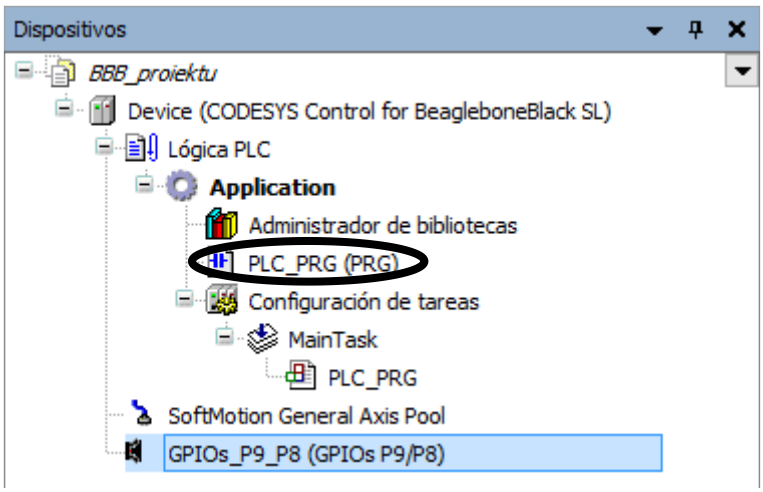

Irudia 47: “GPIOs_P9_P8”-ko “GPIOs Asignación E/S” leihoa.

“digital outputs 0-31”-ren barnean GPIO bakoitzaren kanala dago eta hor aukeratu dugun GPIO_7 kanalari izena ezarri behar zaio. Gure kasuan, “LED” izena.

| Variable | Asignación | Canal | Dirección | Tipo | Unidad | Descripción |
|---|---|---|---|---|---|---|
| | | digital inputs 96-127 | %ID3 | DWORD | | |
| | | digital outputs 0-31 | %QD0 | DWORD | | |
| | | GPIO_0 | %QX0.0 | BOOL | | n/c |
| | | GPIO_1 | %QX0.1 | BOOL | | n/c |
| | | GPIO_2 | %QX0.2 | BOOL | | P9 Pin 22 |
| | | GPIO_3 | %QX0.3 | BOOL | | P9 Pin 21 |
| | | GPIO_4 | %QX0.4 | BOOL | | P9 Pin 18 |
| | | GPIO_5 | %QX0.5 | BOOL | | P9 Pin 17 |
| | | GPIO_6 | %QX0.6 | BOOL | | n/c |
| LED | | GPIO_7 | %QX0.7 | BOOL | | P9 Pin 42 |
| | | GPIO_8 | %QX1.0 | BOOL | | P8 Pin 35 |
| | | GPIO_9 | %QX1.1 | BOOL | | P8 Pin 33 |
| | | GPIO_10 | %QX1.2 | BOOL | | P8 Pin 31 |
| | | GPIO_11 | %QX1.3 | BOOL | | P8 Pin 32 |

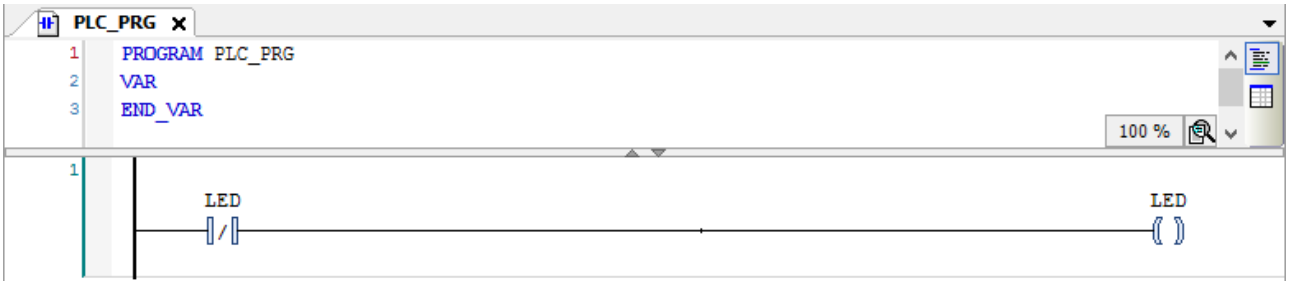
Irudia 48: “GPIO_7” kanalari “LED” izena jarrita “GPIOs Asignación E/S” leihoan.

Behin pinaren konfigurazioa eginda, I/O kontrola frogatzeko programa sortu behar da. Programa sortzeko “Dispositivos”-eko leihoan “PLC_PRG (PRG)”-n sartuko gara (Irudia 49).




Irudia 49: “Dispositivos” leihoa programaren sorrera aukeratzeko.

Ondorengo irudian agertzen den programa sortu da BBB-ko I/O “P9 pin 42”-an LED bat ezarriz. Programa honen helburua LED hori guk jarritako denbora zehatz batean piztea eta itzaltzea txandakatzen joatea da.




Irudia 50: CODESYS-en sortutako programa.

### 3.2.4. Denbora errealeko frogak

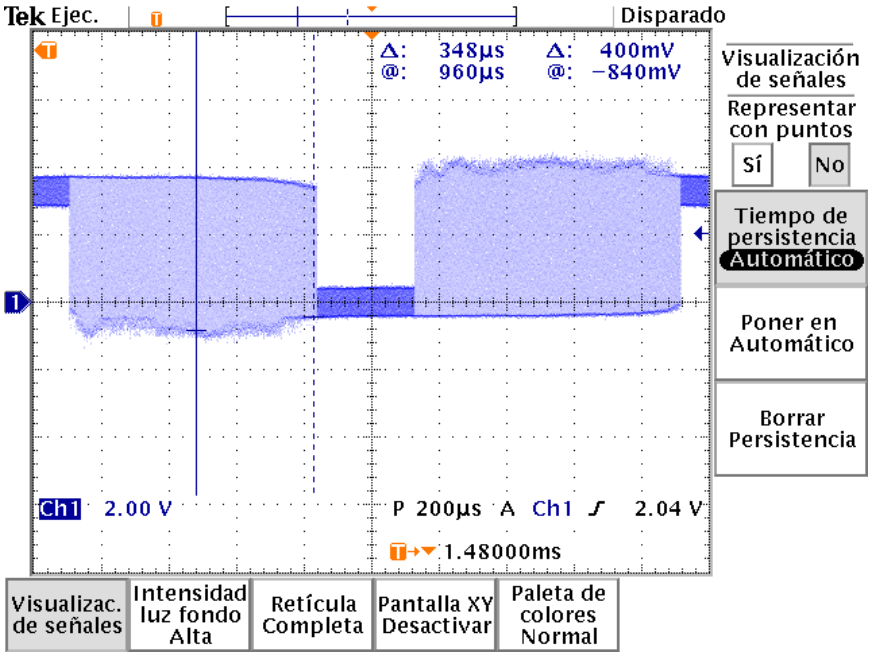
Enpresaren helburua 1ms-an programa zati bat exekutatzea da. Denbora tarte honek zehatza izan behar du, hau da 1ms ±%1 errorearekin. Horretarako exekutatzen den programaren denbora tartea 1ms jarriko dugu "Dispositivos"-en dagoen "MainTask"-ean "Intervalo"→1ms jarrita hurrengo irudian ikusten den bezala.


Irudia 51: "Main Task"aren konfigurazio leihoa.

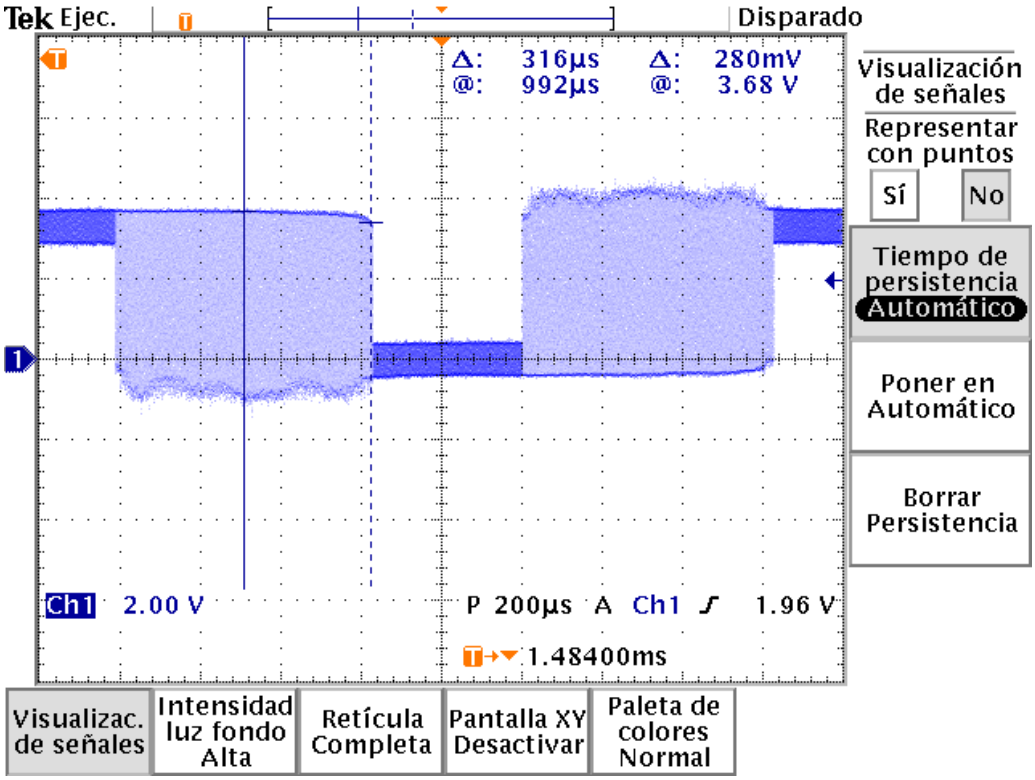
Gure denbora tartea neurtzeko, irteera aldatuko dugu 1ms-ro. Horrela neurtu daiteke irteeraren Jitterra osziloskopioarekin. Ondoren, "Compilar→Compilar" → "En linea→Iniciar sesión"→ "Depuración→Inicio" egingo dugu. Hau egin ondoren, martxan jarriko litzateke programa. Geratu nahi izanez gero, "Depuración→Parada" eta aldaketaren bat egin nahi izanez gero "En línea→Salida".

Programa ondo doala ikusi eta gero osziloskopioa martxan jarriko dugu eta 4 modu ezberdinetara egingo ditugu frogak.

Lehenengo froga 4.4.55-ti-rt-r94 kernela erabiliz egingo da eta Ethernet kablearen eta USB bitartez egingo dira. Hurrengo irudietan irteeraren aldaketa ikusten da eta Jitterraren efektua onar ezina da. Neurtzen den Jitterraren efektua ±300-350µs ingurukoa da, periodoa 1ms izan behar duenean.

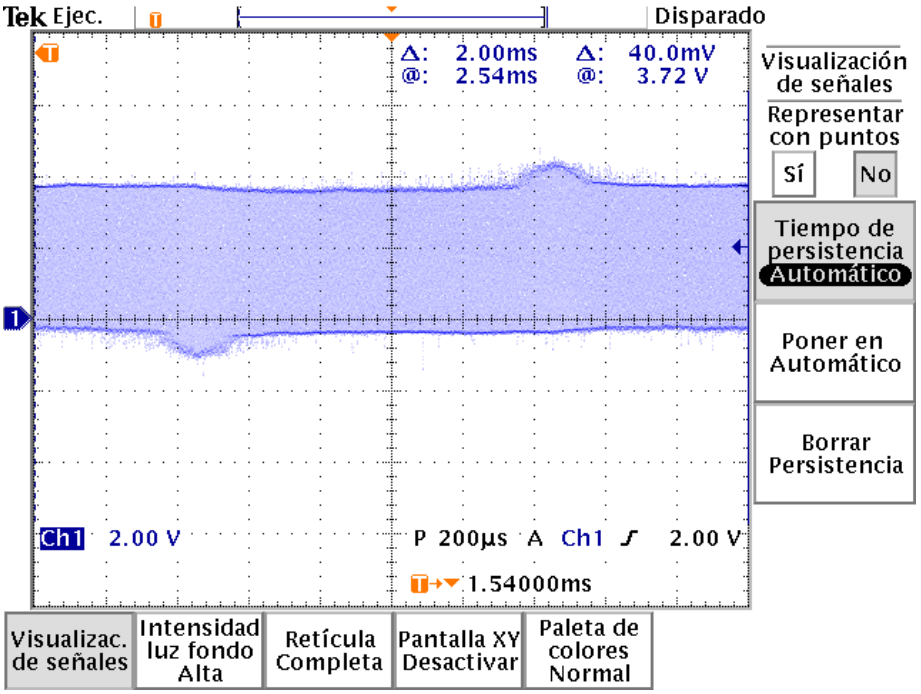



Irudia 52: 4.4.55-ti-rt-r94 kernelarekin sortutako Jitterra. Komunikazioa Ethernet kablearen bitartez.




Irudia 53: 4.4.55-ti-rt-r94 kernelarekin sortutako Jitterra. Komunikazioa USB bitartez.

Bigarren froga 4.4.55-ti-xenomai-r94 kernela erabiliz egingo da eta Ethernet kablearen eta USB bitartez egingo dira. Hurrengo irudietan irteeraren aldaketa ikusten da eta Jitterraren efektua onar ezina da. Neurtzen den Jitterraren efektua, 2ms-koa da, periodoa 1ms izan behar duenean.




Irudia 54: 4.4.55-ti-xenomai-r94 kernelarekin sortutako Jitterra. Komunikazioa Ethernet eta USB kablearen bitartez.

Frogak egin ondoren, ikusi da enpresako eskakizunak ez direla betetzen, Jitterra oso handia baita frogatu den bezala. Enpresak hartu duen erabakia hau ikusi eta gero kernel minimo bat sortzea da Jitter onargarria lortzeko. Horrela Jitter handi hori desagerraraztea lortu nahi da beraiek nahi duten denbora erreala lortzeko.

## 3.3. Eremu busak

Eremu busekin komunikazioak egin behar dira. Horretarako, erabiliko diren hiru eremu busen moduluak ezagutu behar dira. Lehenengo, EtherCAT-ekin hasiko gara. Bigarren, PROFINET-ekin jarraituko da eta amaitzeko, CANopen-ekin jarriko gara. Eremu bus bakoitzarekin jazarpen ezberdin bat egin behar da, non jarraian azalduko den bakoitzaren prozesua.

### 3.3.1. EtherCAT

#### 3.3.1.1. EtherCAT modulua

EtherCAT-eko moduluaren terminalen eta CODESYS-en arteko erlazioa:

- **Sarreren moduluak:**

- **EL1004**

Moduluko 1pina → CODESYS-eko 0bita
Moduluko 2pina → CODESYS-eko 1bita
Moduluko 7pina → CODESYS-eko 2bita
Moduluko 8pina → CODESYS-eko 3bita

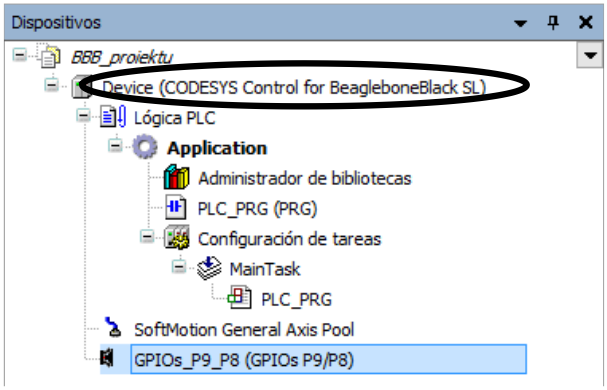
- **Irteeren moduluak:**

- **EL2004**
  Moduluko 1pina → CODESYS-eko 0bita
  Moduluko 2pina → CODESYS-eko 1bita
  Moduluko 7pina → CODESYS-eko 2bita
  Moduluko 8pina → CODESYS-eko 3bita

- **EK1100 GOIBURUA**

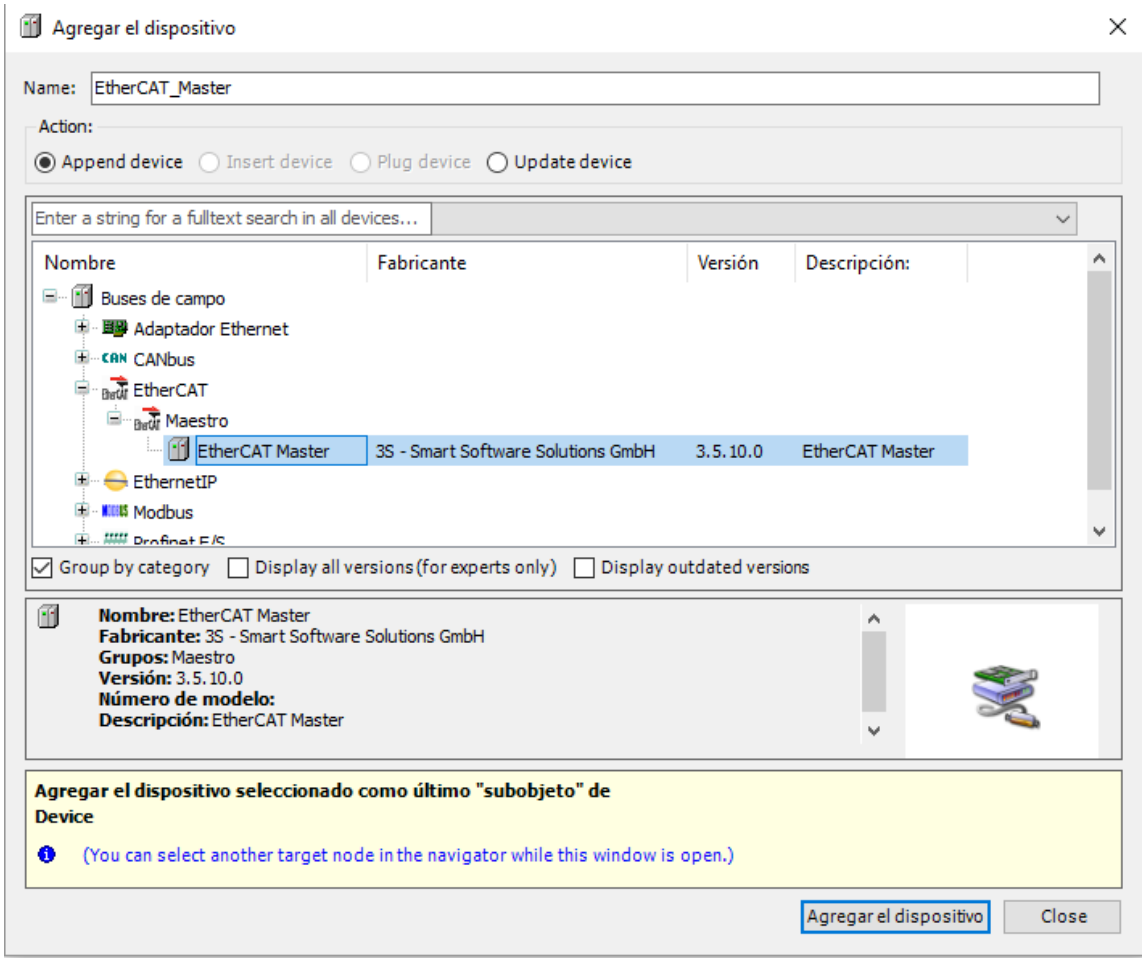
#### 3.3.1.2. EtherCAT-en funtzionamendua

EtherCAT-eko moduluak lortu ondoren eta bakoitzean egin beharreko konexioak egin eta gero, Ethernet kablearen bitartez BBB-ra konektatu behar da. Behin konektatuta dagoelarik, CODESYS—ean egin behar da EtherCAT-eko konfigurazioa. Horretarako, CODESYS-ean "Dispositivos"-eko "Device"-ren gainean eskuineko botoiarekin sakatu behar da hurrengo irudian ikusten den bezala.



Irudia 55: "Device" aukeratzeko "Dispositivos" leihoa.

Agertutako menuan "Agregar el dispositivo" zapaldu eta ondorengo argazkian agertzen den leihoa irekiko da eta hor "EtherCAT" aukeratu beharko da.

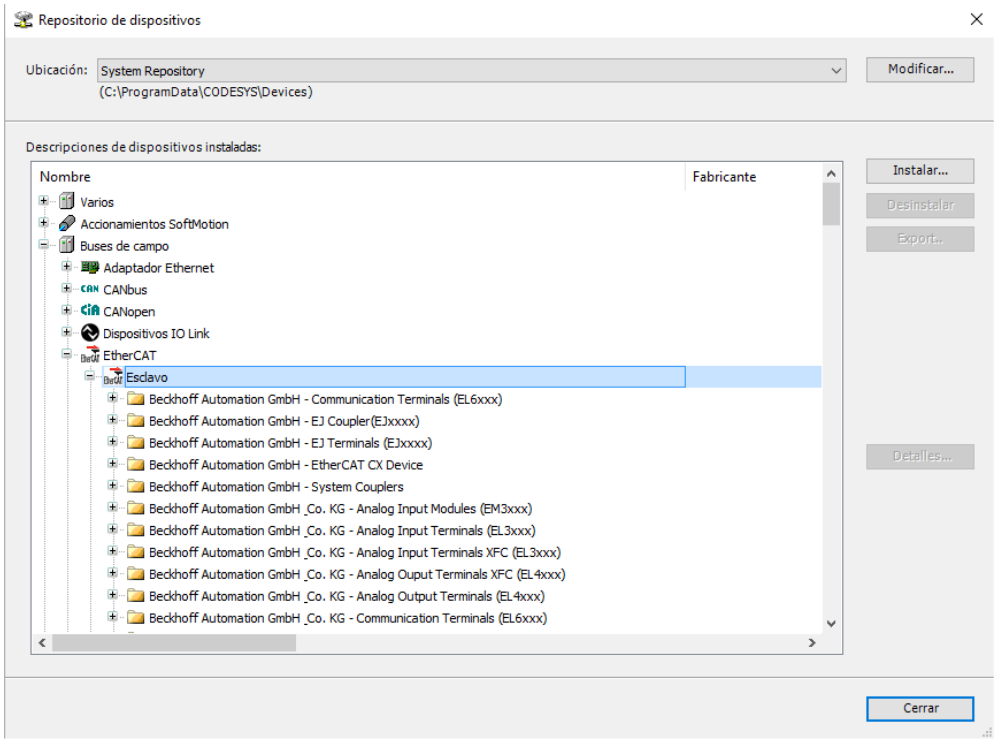



Irudia 56: “Agregar el dispositivo” leihoa EtherCAT aukeratuta.

EtherCAT aukeratutakoan, “Agregar el dispositivo” sakatu eta “Dispositivos”-en agertuko da. Orduan, CODESYS-en “Iniciar sesión” (Irudia 57) egingo da eta “Dispositivos” menuko “EtherCAT”-en gainean eskuineko botoiarekin “Buscar el dispositivo” zapalduko dugu.

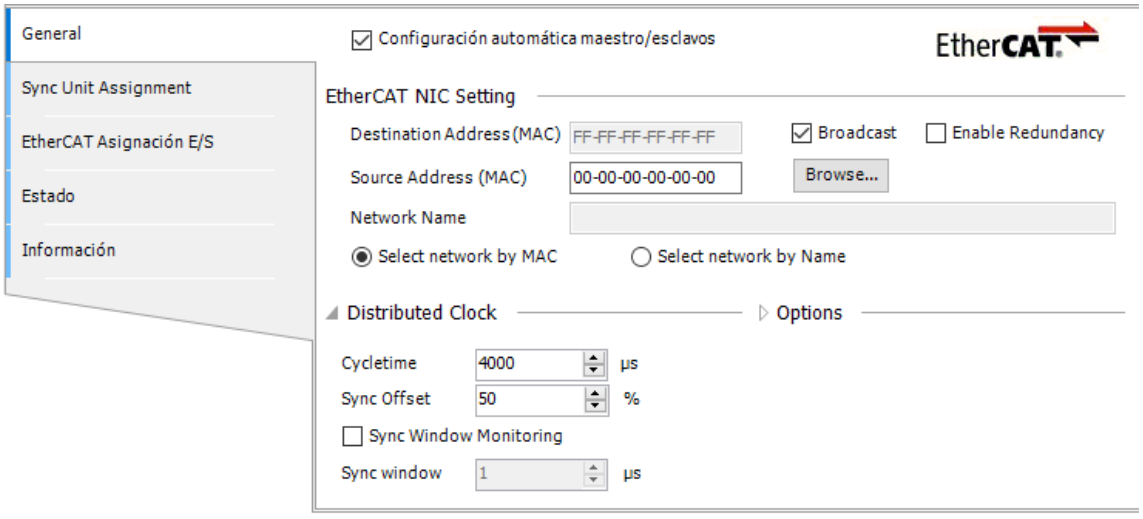

Irudia 57: Iniciar sesion botoia.

Honela, BBB-ra konektatu ditugun EtherCAT moduluak agertu beharko lirateke. Baina horretarako, BECKHOFF-eko orrialdean sartuz gure moduluetako .xml estentsioak deskargatu behar dira CODESYS-ek ezagutu ditzan. Hau da, CODESYS-ek gure moduluak bilatzeko erreferentzia bat izan behar du. Orduan, “Herramientas”→ “Repositorio de dispositivos” eta hemen gure .xml-ak aukeratu eta “instalar” klikatuz gordeko lirateke gure proiektuan (Irudia 58).



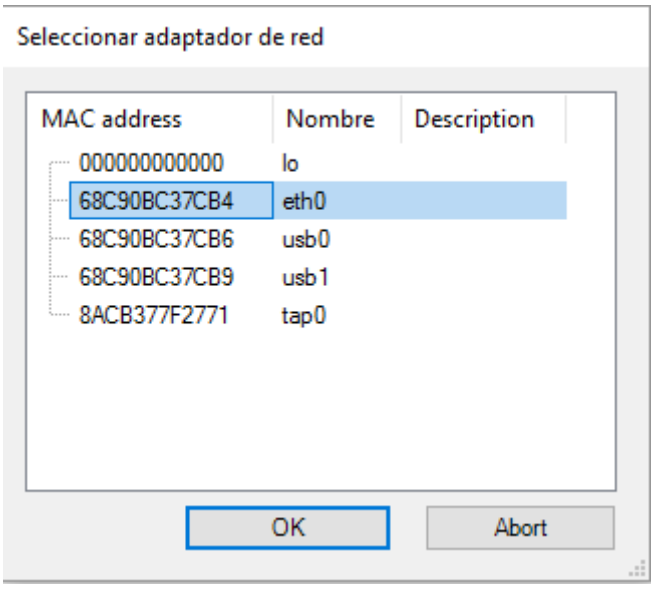
Irudia 58: “Repositorio de dispositivos”.

Gure proiektuan gordetakoan, “Dispositivos”-en “EtherCAT” agertuko litzateke eta bertan bi aldiz klikatuz ondorengo leihoa irekiko da.



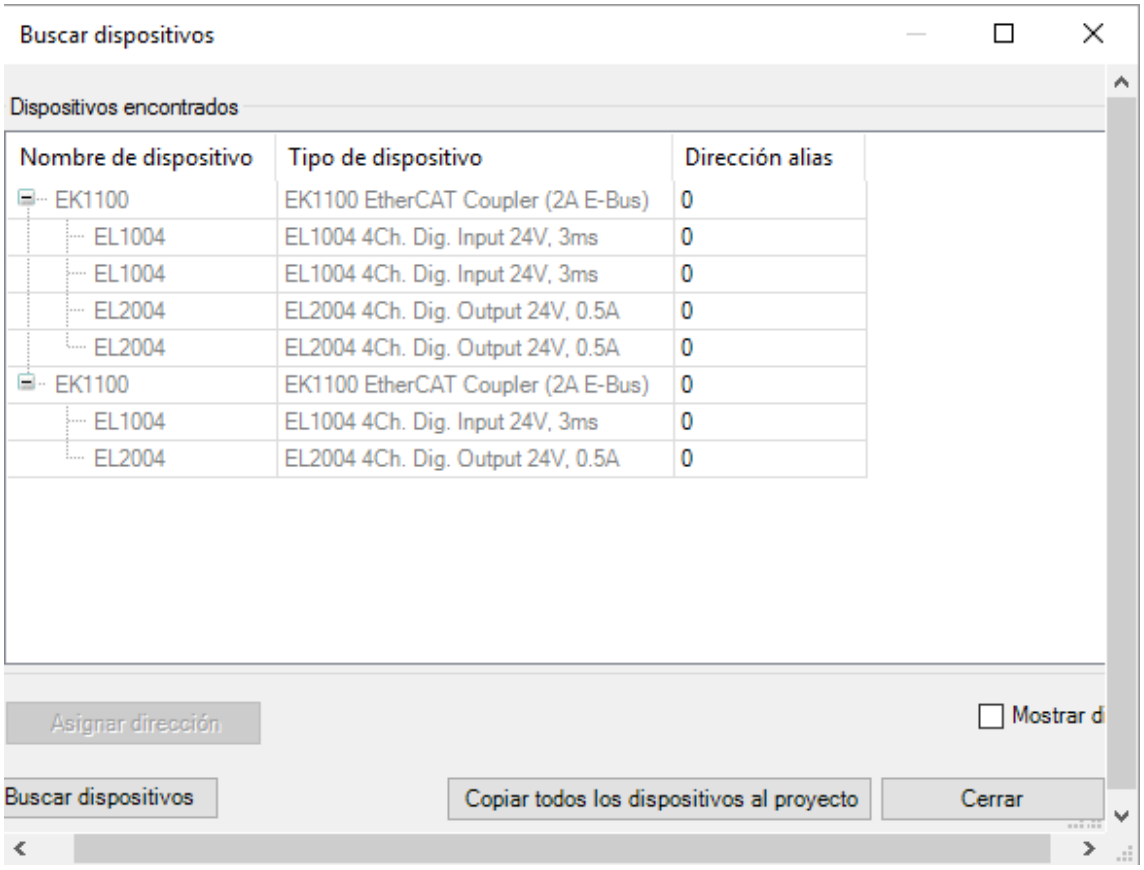
Irudia 59: EtherCAT leihoa.

Bertan, “Source Address” hori da aldatu behar dena. Orduan, “Browse” zapalduz “eth0” aukeratuko da hurrengo irudian ikusten den bezala, gure Etherneteko MAC-a izanik.



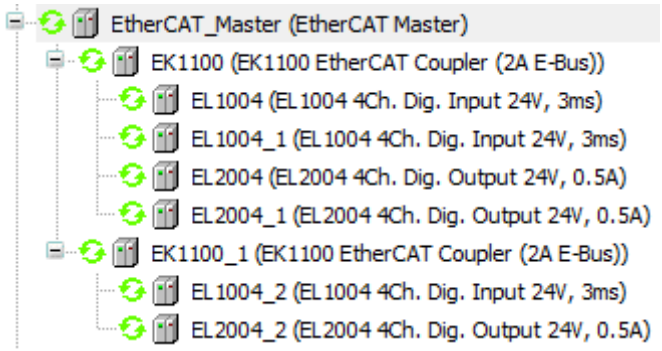

Irudia 60: “Source Address”-eko “Browse” leihoa.

Hau egin ondoren, berriz “Iniciar sesión” egin eta “EtherCAT”-en gainean eskuineko botoia sakatuz “Buscar dispositivos” zapalduko dugu. Horrela, lehen instalatu ditugun .xml-ak gure moduluetan aurkituko ditu hurrengo irudian ikusten den bezala (Ikus erreferentzia [14]).



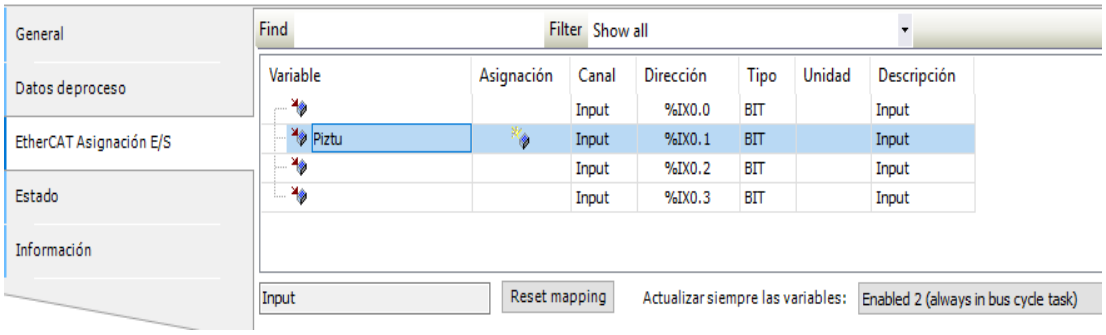

Irudia 61: “Buscar dispositivos” leihoa.

Gure moduluko gailu guztiak agertzean “Copiar todos los dispositivos al proyecto” zapalduko dugu eta honela denak “EtherCAT master”-en barruan sartuko dira esklabu bezala. Orduan, “Iniciar sesión” zapaldu eta gailu guztiak “berdez” jarriz gero ezagutu dituela esan nahi du. Honekin ez dagoela arazorik jakiten dugu eta orain pinen konfigurazioa faltako litzateke. Ondorengo irudian ikus daiteke moduluko gailu guztiak ezagutu dituela eta ondo konektatuta daudela.



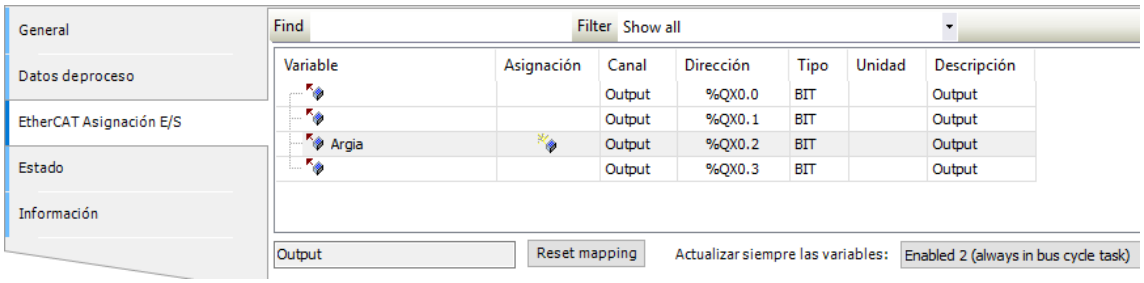
Irudia 62: “Dispositivos”-en EtherCAT konektatuta.

“EL1004” gailuak sarrerak dira eta “EL2004” gailuak ordea, irteerak dira. Hau jakinda, bakoitzaren konfigurazioa egin behar da funtzionatzen duen ikusteko.
Edozein “EL1004” moduluetan egin daiteke sarreren konfigurazioa; baina, zein modulu aukeratu den kontutan izan behar da bertako pinak aktibatuko direlako. Orduan, sarrera moduan erabili nahi den gailuan bi aldiz klikatuz ondorengo leihoa irekiko da. Bertan, “EtherCAT Asignación E/S” aukeratu eta moduluan ikusi nahi den pineko BIT-a aukeratu behar da. Gure kasuan, “EL 1004” moduluko %IX0.1 direkzioa duen BIT-a aukeratu da; hau da, modulu horretako 2. pina aktibatuko da.



Irudia 63: EtherCAT sarrerako moduluaren “EtherCAT Asignación E/S” leihoa.

Sarrera konfiguratu eta gero, irteera konfiguratu behar da. Orain, “EL 2004” irteeretako edozein moduluetako bat aukeratuko dugu eta sarrerarekin bezala konfigurazioa egingo dugu. Kasu honetan, “EL 2004” moduluko %QX0.2 direkzioa duen BIT-a aukeratu da; hau da, moduluko horretako 7. pina aktibatuko da.

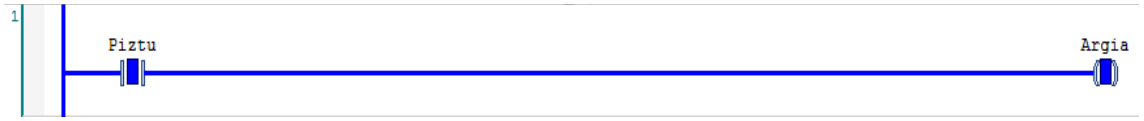



Irudia 64: EtherCAT irteerako moduluaren "EtherCAT Asignación E/S" leihoa.

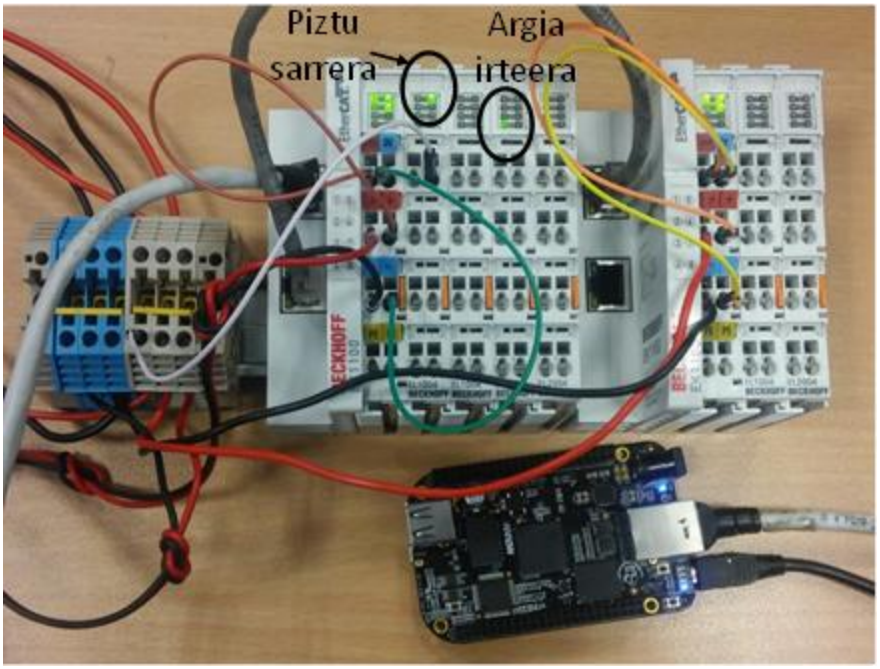
I/O pinak konfiguratu ondoren, programa txiki bat egingo dugu egindako konfigurazioa eta moduluko pinak ea bat datozen ikusteko. Orduan, lehen sortutako sarrera eta irteerarekin hurrengo irudiko programa sortu da.


Irudia 65: EtherCAT probatzeko programa.

Egindako programa konpilatu eta "Iniciar sesión" sakatzean, moduluan konfiguratutako sarrerako pinari (Piztu) tentsioa emango diogu eta irteerako pina (Argia) programan eta moduluan piztuko da. Ondorengo irudietan ikus daitekeen bezala.




Irudia 66: Programa martxan.




Irudia 67: BBB-a eta EtherCAT modulua martxan sarrera eta irteera aktiboak direla programatuta bezala.

## 3.3.2. PROFINET

### 3.3.2.1. PROFINET modulua

PROFINET-eko moduluaren terminalen eta CODESYS-en arteko erlazioa:

- **Sarreren modulua:**

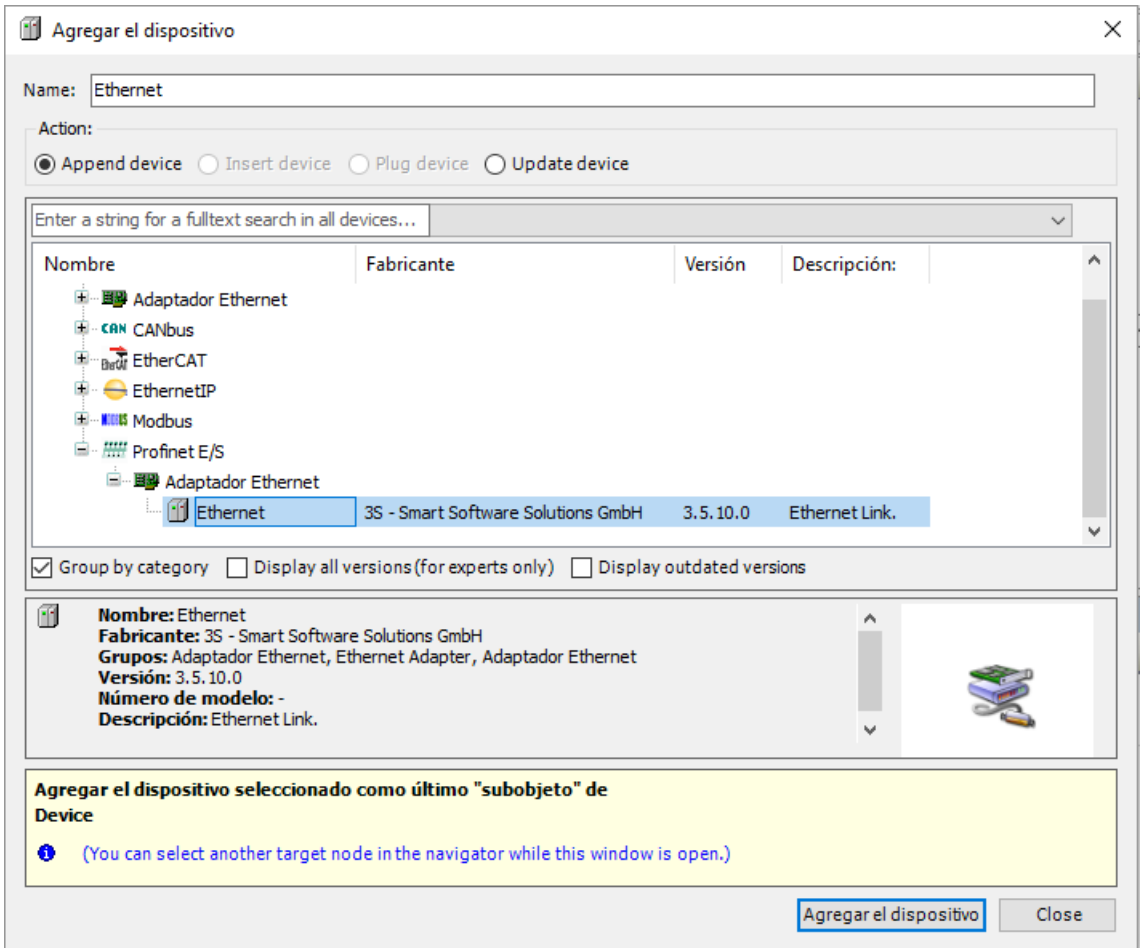
  - **4 DI DC24V ST (6ES7 131-4BD01-0AA0)**
    Moduluko 1pina → CODESYS-eko 0bita
    Moduluko 5pina → CODESYS-eko 1bita
    Moduluko 2pina → CODESYS-eko 2bita
    Moduluko 6pina → CODESYS-eko 3bita

- **Irteeren modulua:**

  - **4 DO DC24V/0.5A ST(6ES7 132-4BD02-0AA0)**
    Moduluko 1pina → CODESYS-eko 0bita
    Moduluko 5pina → CODESYS-eko 1bita
    Moduluko 2pina → CODESYS-eko 2bita
    Moduluko 6pina → CODESYS-eko 3bita

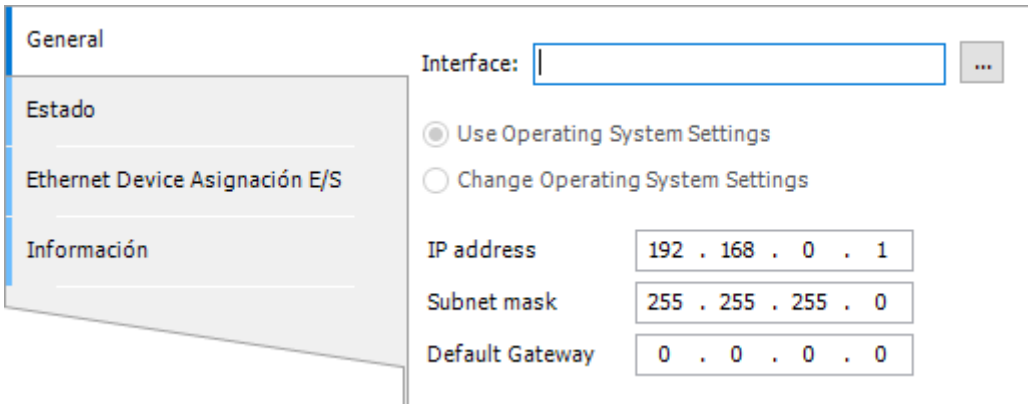
### 3.3.2.2. PROFINET-en funtzionamendua

Hasieran, EtherCAT-en berdina izango litzateke. Hau da, “Dispositivos”-eko “Device”-ren gainean eskuineko botoiarekin sakatu eta “Agregar el dispositivo” zapalduz leiho bat irekiko da; baina kasu honetan “PROFINET” aukeratuko dugu.



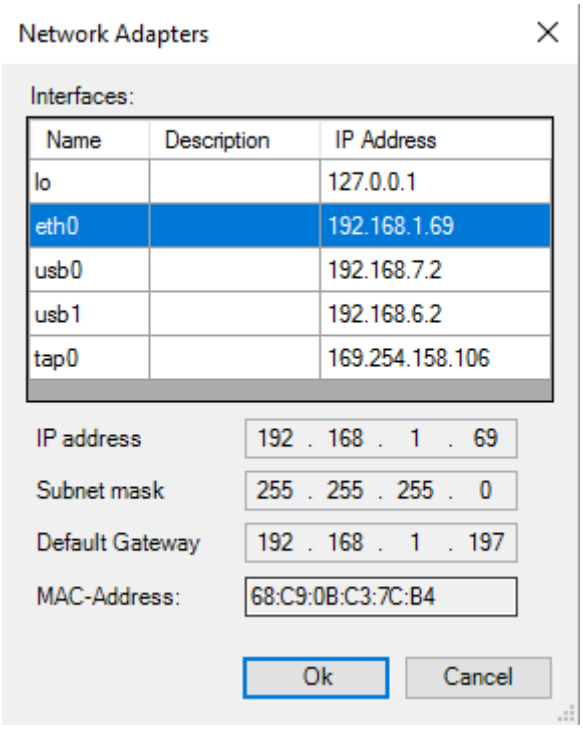

Irudia 68: “Agregar el dispositivo” PROFINETerako.

“Agregrar el dispositivo” zapaldu ondoren “Dispositivos”-en agertuko da. Orduan, hor sartuko gara IP-a aukeratzeko, PROFINET-eko modulua Etherneteko kable bitartez konektatzen delako.



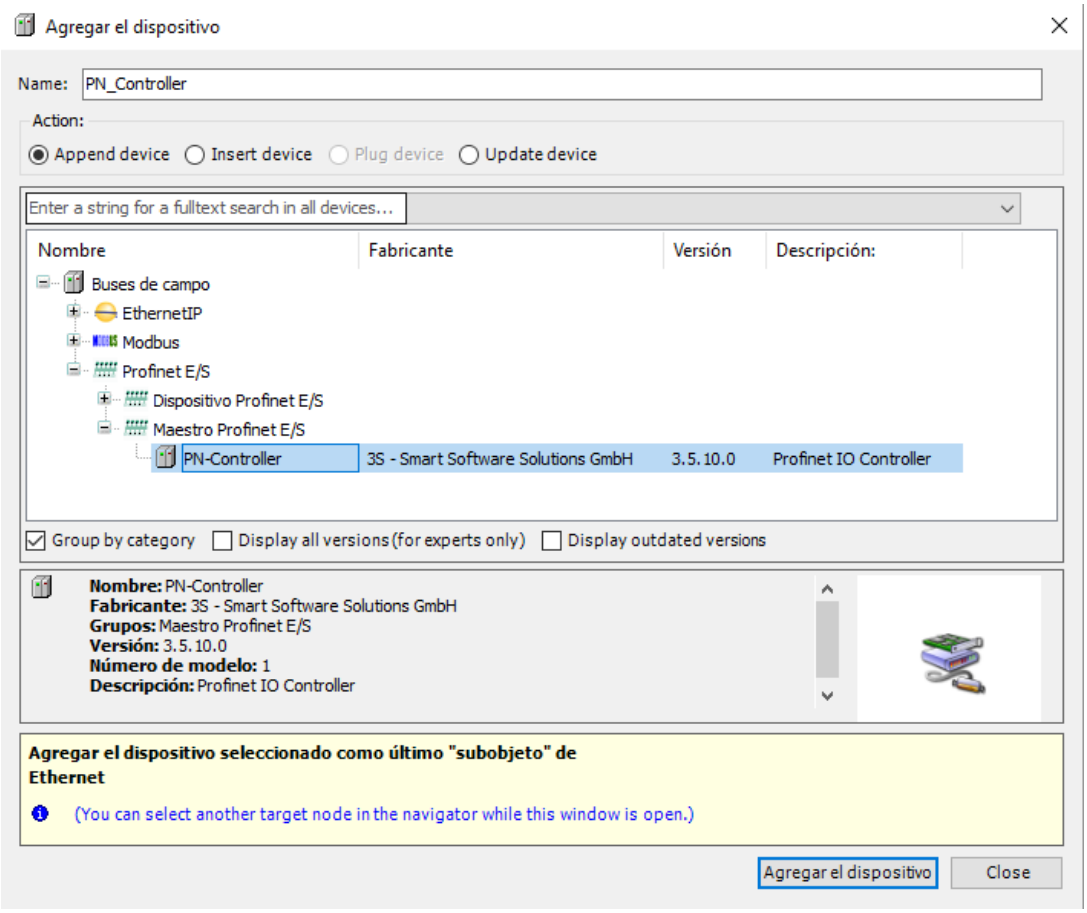

Irudia 69: PROFINETen leiho orokorra.

Leihoa irekitzean “Interface” ondoko hiru puntuetan zapalduz agertuko dira aukera posibleak hurrengo irudian ikusten den bezala. PROFINET Ethernet bidez konektatzen denez gure BBB-ko “eth0” aukeratu behar da.



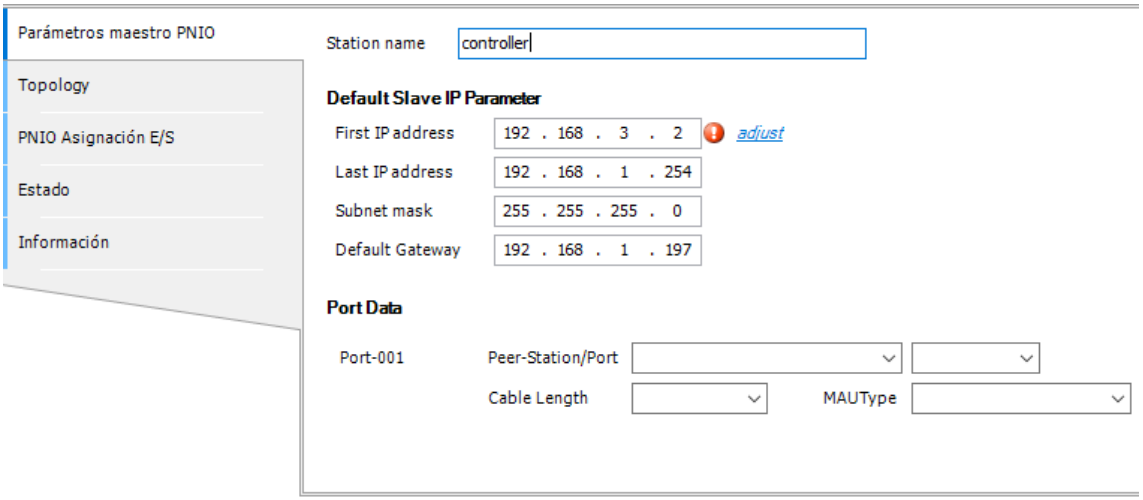

Irudia 70: “Interface”-rako aukerak, gurea “eth0”.

Konektatu ondoren, “Dispositivos”-eko “Ethernet”-en gainean eskuineko botoiarekin zapalduz “Agregar el dispositivo” sakatu behar da. Leiho horretan “PROFINET E/S”→”Maestro PROFINET E/S”→ “PN-Controller” aukeratu eta “Agregar dispositivo” sakatuz programara kopiatuko da.



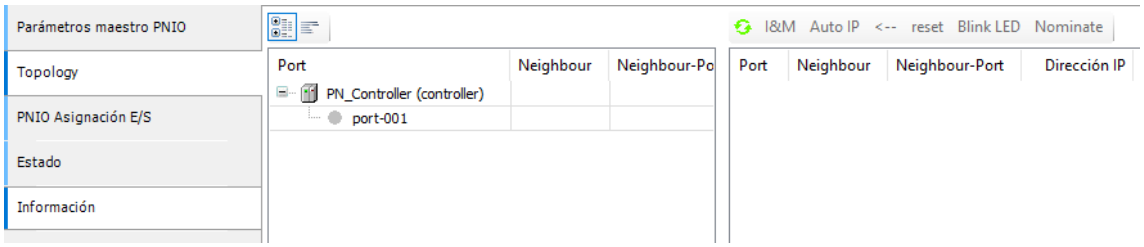

Irudia 71: “Agregar el dispositivo” “PN-Controller” instalatzeko.

“PN-Controller” agertutakoan bi aldiz klikatuko dugu bere gainean eta horrela bere leihoa irekiko da. Bertan, irudian agertzen den “adjust” zapalduko dugu lehen sartutako IP-a aktualiza dezan. Horrela, gure kontrolagailua konektatuko da.




Irudia 72: “PN-Controller”-eko “Parámetros maestro PNIO” leihoa.

Leiho honetako “Topology” menuan sartuz, ondorengoa irudia agertuko da.

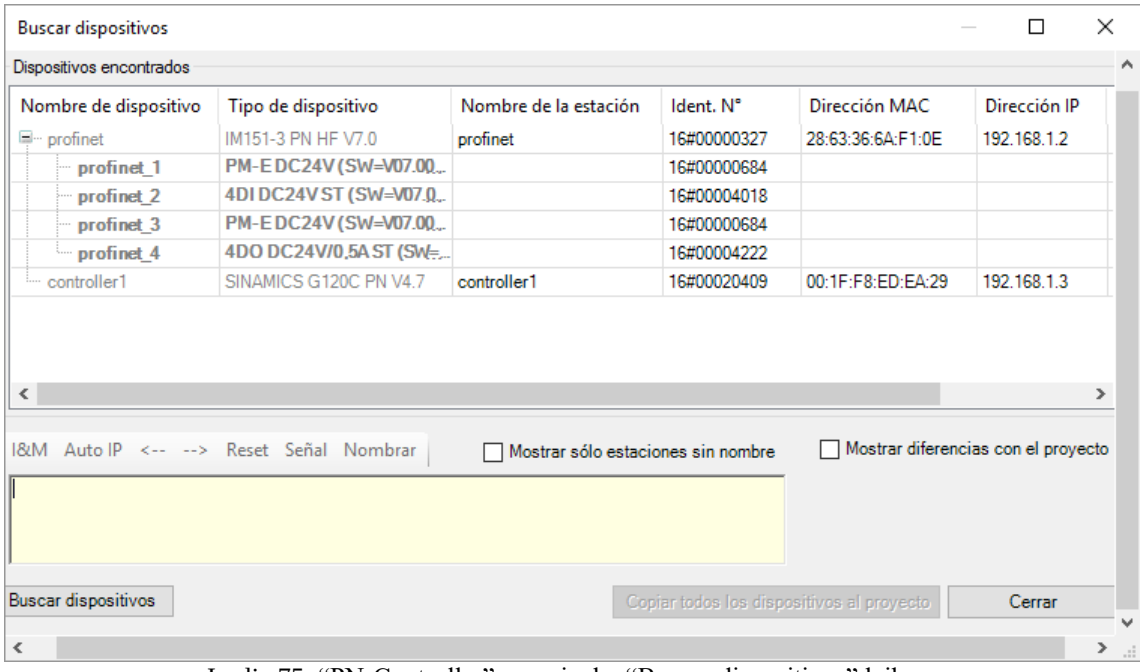



Irudia 73: "PN-Controller"-eko "Topology" leihoa.

Hemen, "Refresh" berdeari emango diogu, honela konektatuta dauden moduluak bilatzen ditu eta bertan agertuko dira. Irudian ikusten den bezala "no data!" agertzen da. Hau da, "EtherCAT"-en bezala PROFINET moduluetako .xml-ak bilatu behar dira baino kasu honetan SIEMENS-eko orrialdean, PROFINET-eko moduluak bertakoak direlako (Ikus erreferentzia [15]).


Irudia 74: "PN-Controller"-eko "Topology" leihoa aktualizatuta.

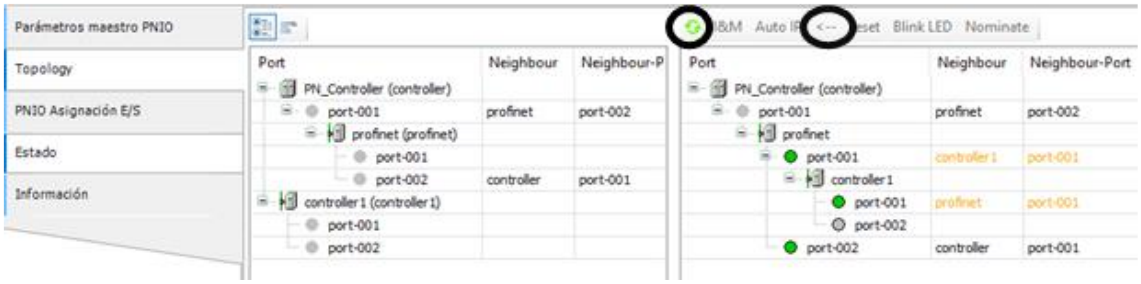
.xml-ak deskargatu eta gero, "Repositorio de dispositivos"-en bitartez instalatu behar dira aurreko eremu busarekin egin den bezala.
Instalatutakoan, "Iniciar sesión" sakatu eta "PN-Controller"-ean xaguaren eskuineko botoia zapalduz "Buscar el dispositivo" aukeratuko dugu. Ondorengo irudian ikusten den bezala gure PROFINET moduluetako gailuen izenak agertuko dira.




Irudia 75: "PN-Controller"-en gaineko "Buscar dispositivos" leihoa.

“PROFINET” izena duen gailua, PROFINET-eko modulua da. “controller1” jartzen duena ordea, PROFINET motako inbertsorea da.

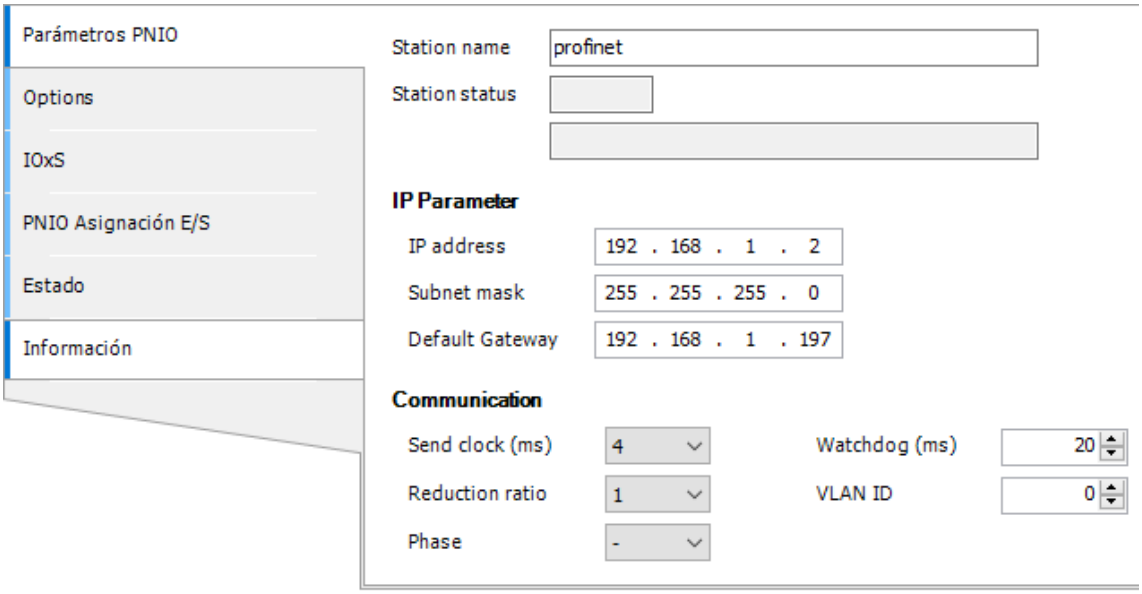
Moduluko gailuak aurkituta dituelarik, “Topology”-n sartuko gara berriz ere eta lehenengo “Refresh” egingo dugu. Bigarrenik, ezkerrerako norantza adierazten duen botoia zapalduko dugu (Irudia 76). Horrela bilatu dituen gailuak CODESYS-ean sartuko ditu, hurrengo irudian ikusten den bezala.




Irudia 76: “PN-Controller”-eko “Topology” leihoa aktualizatuta.

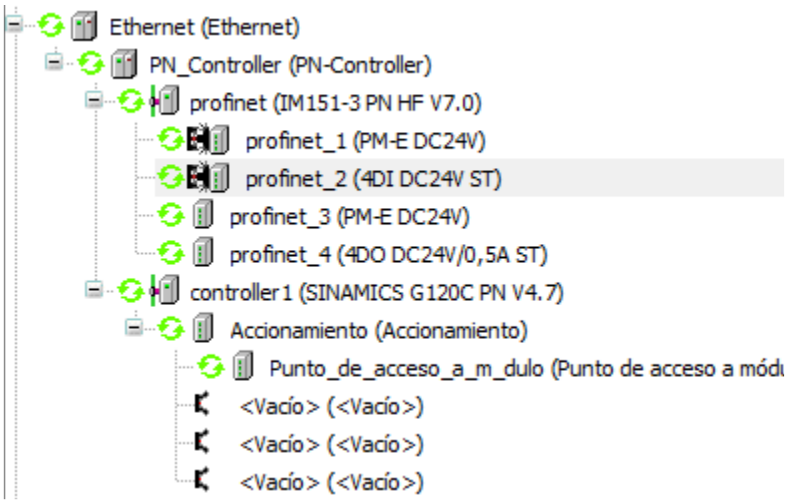
Jitter ugari dagoenez paketeak bidaltzen egoten den bitartean errorerik ez emateko bidaltze-denbora eta watchdog-aren balioak handitu behar dira. Orduan, “PROFINET” eta “controller1”-ko bidaltze denbora eta watchdog-a 4ms eta 20ms-tara jarri beharko dira.

Gailuak aurkitu ondoren IP berria eman behar zaie kontrolagailurekin bat egin dezan. Horretarako, IP-ko azkeneko zenbakia bakarrik aldatu behar da, 192.168.1. mantenduz biek konexio bera izateko.




Irudia 77: PROFINET izeneko gailuaren “Parámetros PNIO” leihoa.

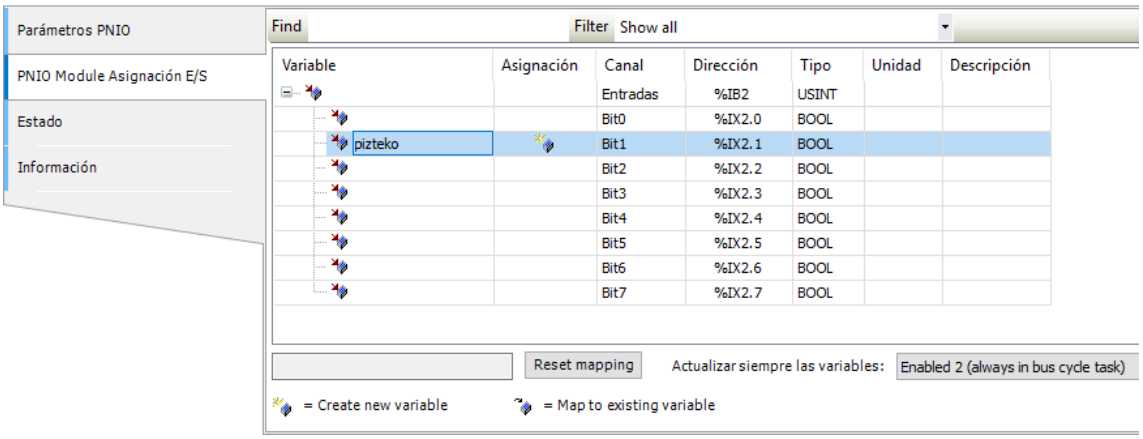
IP-ak konfiguratu ondoren, “Iniciar sesión” egin eta denak berdez badaude arazorik ez dagoela esan nahi du, hurrengo irudian agertzen den bezala.




Irudia 78: “Dispositivos”-eko PROFINET-eko zatia.

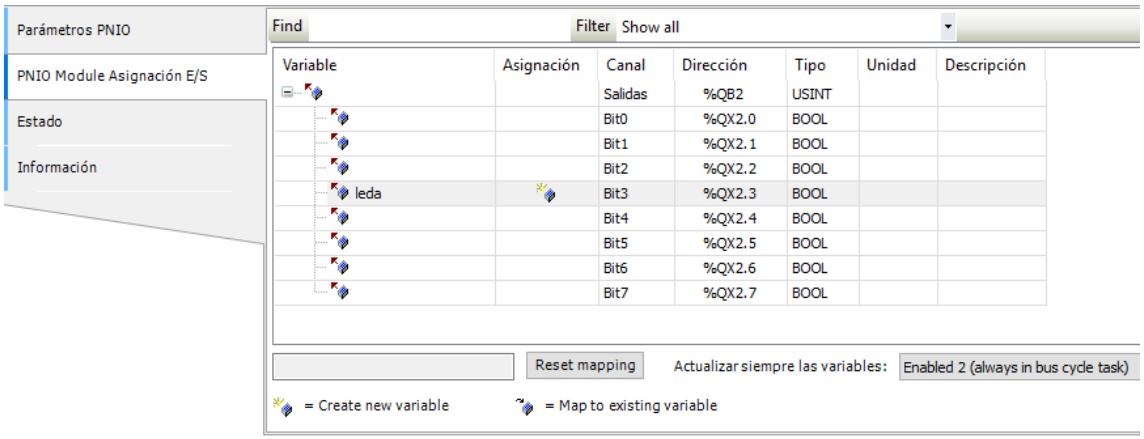
Orain, sarrerak eta irteerak konfiguratu behar dira ondoren egingo den programak funtziona dezan. Horretarako, “PROFINET_2” sarrerarena izango da, “4DI DC24V ST” sarreren gailua delako. “PROFINET_4” ordea, “4DO DC24V/0,5A V4.7” irteeraren gailua izango da.

Sarreraren pinarekin hasteko, “PROFINET_2”-an klikatuz “PNIO Module Asignación E/S”-n sartuko gara eta erabili nahi den sarrerako bita aukeratuko da. Nahiz eta hemen 8 BIT-eko aukera izan moduluak 4 BIT bakarrik onartzen ditu. Gure kasuan, BIT1 aukeratu dugu, hau da, gure moduluan “4DI DC24V ST” gailuko 5. pina aktibatuko da.




Irudia 79: “PROFINET_2”-ko “PNIO Module Asignación E/S” leihoa.

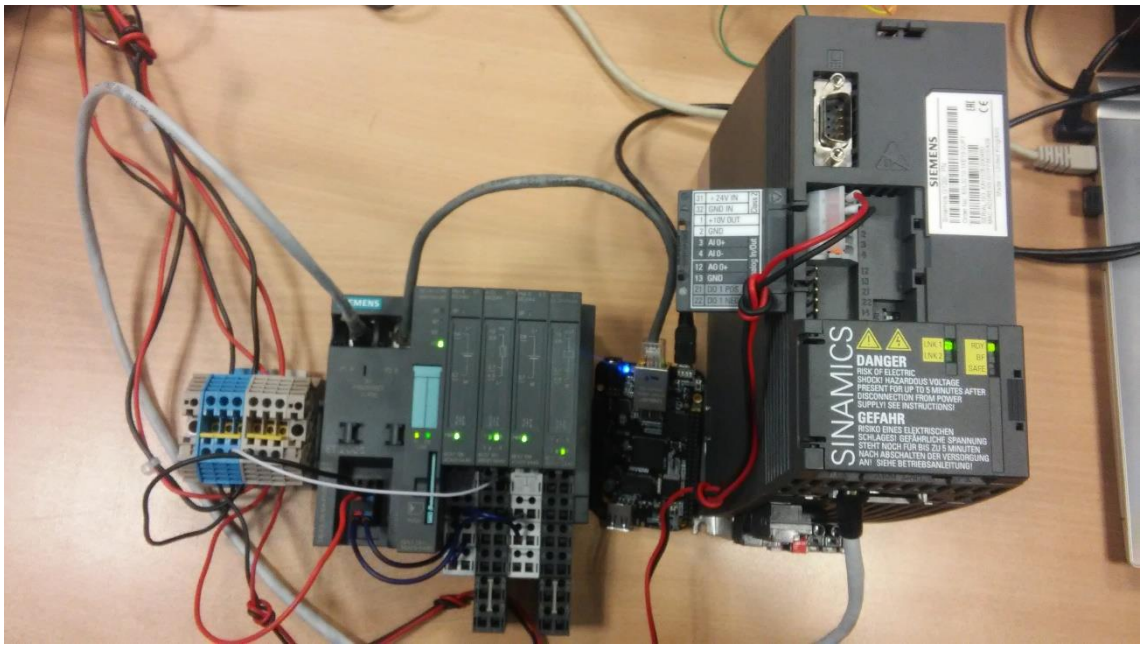
“PROFINET_4”-an klikatuz “PNIO Module Asignación E/S”-n sartuko gara eta erabili nahi den irteerako bita aukeratuko da. Kasu honetan, sarrerako gailuaren berdina gertatzen da; hau da, gailuak 4 BIT bakarrik ditu. Gure kasuan, BIT3 aukeratu dugu, hau da, gure moduluan “4DO DC24V/0,5A V4.7” gailuko 6. pina aktibatuko da.




Irudia 80: “PROFINET_4”-ko “PNIO Module Asignación E/S” leihoa.

I/O pinak konfiguratu ondoren, programa txiki bat egingo dugu egindako konfigurazioa eta moduluko pinak ea bat datozen ikusteko. Orduan, lehen sortutako sarrera eta irteerarekin programa bat sortu da.
Programa hori konpilatu eta “Iniciar sesión” sakatzean, moduluan konfiguratutako sarrerako pinari (pizteko) tentsioa emango diogu eta irteerako pina (leda) programan eta moduluan piztuko da. Ondorengo irudietan ikus daitekeen bezala.


Irudia 81: PROFINETeko proba egiteko programa.




Irudia 82: PROFINETeko modulua eta BBB-a martxan.

SINAMICS driverrarekin komunikazioa ere frogatu da. Parametro batzuk aldatu dira, adibidez abiadura begiztaren Kp-a, baina motorrik ez denez konektatu bakarrik egin da, komunikazioa zuzena dela frogatuz.

### 3.3.3. CANopen

#### 3.3.3.1. CANopen modulua

CANopen-eko moduluaren terminalen eta CODESYS-en arteko erlazioa:

- **Sarreren moduluak:**

  - **KL1804-1**
    Moduluko 1pina → CODESYS-eko 0bita
    Moduluko 3pina → CODESYS-eko 1bita
    Moduluko 5pina → CODESYS-eko 2bita
    Moduluko 7pina → CODESYS-eko 3bita

  - **KL1804-2**
    Moduluko 1pina → CODESYS-eko 4bita
    Moduluko 3pina → CODESYS-eko 5bita
    Moduluko 5pina → CODESYS-eko 6bita
    Moduluko 7pina → CODESYS-eko 7bita

- **Irteeren moduluak:**

  - **KL2404-1**
    Moduluko 1pina → CODESYS-eko 0bita
    Moduluko 5pina → CODESYS-eko 1bita
    Moduluko 4pina → CODESYS-eko 2bita
    Moduluko 8pina → CODESYS-eko 3bita

  - **KL2404-2**
    Moduluko 1pina → CODESYS-eko 4bita
    Moduluko 5pina → CODESYS-eko 5bita
    Moduluko 4pina → CODESYS-eko 6bita
    Moduluko 8pina → CODESYS-eko 7bita

- **KL9010 bus terminala**

#### 3.3.3.2. RS485 CAN CAPE-aren konfigurazioa

CANopen komunikazioa egiteko, lehenik eta behin “RS485 CAN CAPE”-a ezarri behar zaio BBB-ari. Kapa horrekin komunikazioa egiteko, BBB-aren terminalean “can0”-aren konfigurazioa egin behar da. Horretarako ondorengo pausoak jarraitu behar dira:

```
modprobe can

modprobe can-dev

modprobe can-raw
```

```
ip link set can0 up type can bitrate 125000

ifconfig can0 up
```

Azken bi komando hau erabiltzean “can0” ez dagoenez konfiguratuta arazoa emango du ondorengo mezua agertuz:

```
Cannot find device "can0"
```

Orduan, “can0” ager dadin ondorengo prozesua jarraitu behar da:

```
sh -c "echo 'BB-CAN1'> /sys/devices/platform/bone_capemgr/slots"
```

Honela, behar den “can0”-a terminalean dagoela ikus daiteke ondorengo komandoaren bitartez:

```
cat /sys/devices/platform/bone_capemgr/slots
```

Komando honekin ondorengo mezua agertuko litzateke:

```
0: PF----  -1
1: PF----  -1
2: PF----  -1
3: PF----  -1
4: P-O-L-   0 Override Board Name,00A0,Override Manuf,BB-CAN1
```

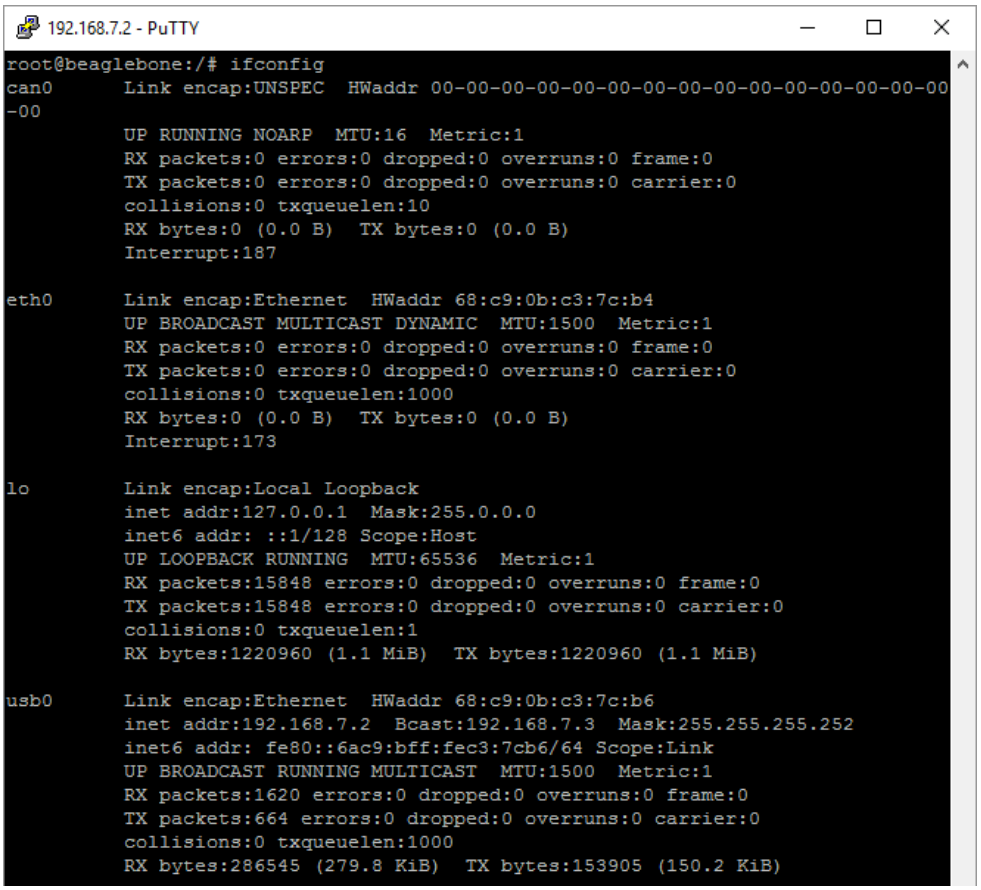
Ondoren aurretik funtzionatu ez duen komando berdinak sartuko ditugu baina kasu honetan ez du arazorik emango:

```
ip link set can0 up type can bitrate 125000

ifconfig can0 up
```

Konfiguratu dela ikusteko ondorengo komandoa erabiliko da:

```
Ifconfig
```

Orduan, hurrengo irudian ikusten mezua agertuko da eta ikus daiteke lehenengoa “can0” dela.




Irudia 83: "ifconfig" komandoaren mezua CAN konfigurazio aktiboa delarik.

Aurreko irudian ikus daitezke BBB-an konfiguratuta dauden gailu guztiak. Hasieran, "can0" ez zegoen instalatuta, baina konfiguratu ondoren ikus daiteke beste komunikazio konfigurazioarekin agertzen dela.

Gainera "can0"-a eskuz sartu daitekeela jarri daiteke horrela konektatzen garen bakoitzean bi komando hauekin nahikoa da:

```
ip link set can0 up type can bitrate 125000

ifconfig can0 up
```

Orduan, estatikoa izan dadin ondorengoa egin behar da gure terminalean:

```
cd /etc/network

nano interfaces
```

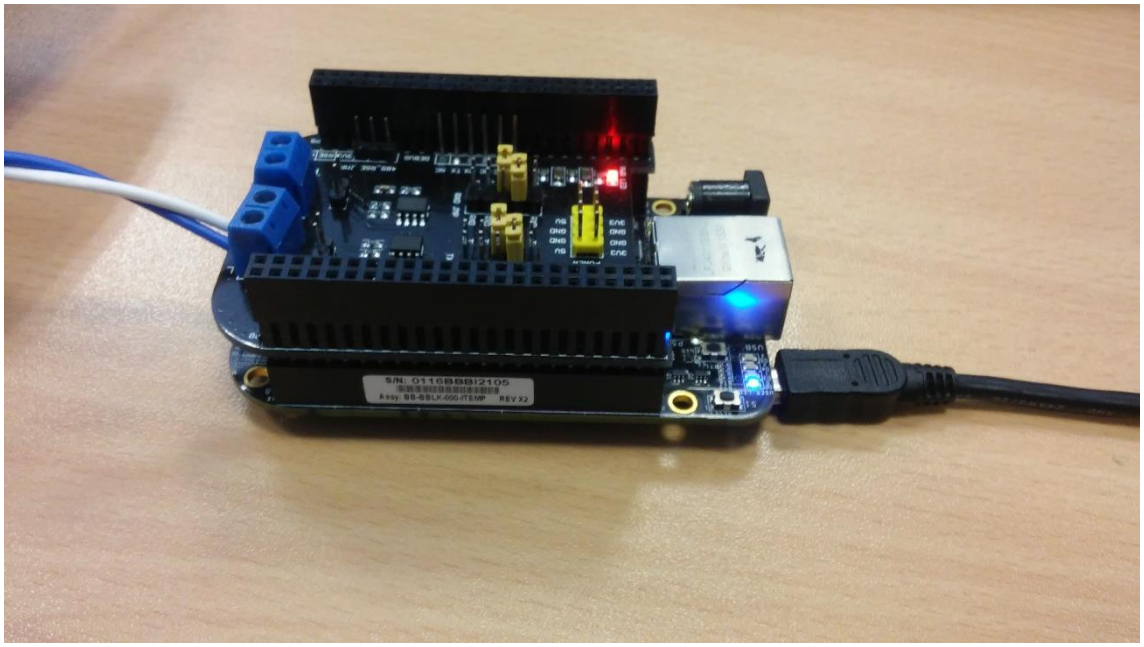
nano komandoarekin fitxategi horretan sartuz aldaketak egiteko aukerak daude. Kasu honetan, ondorengoa idatzi beharko litzateke:

```
auto can0
iface can0 inet manual
        #pre-up ip link set $IFACE type can bitrate 125000 listen-only off
        pre-up /sbin/ip link set $IFACE type can bitrate 125000 triple-sampling on
        up /sbin/ifconfig $IFACE up
        down /sbin/ifconfig $IFACE down
```

Orain Debian-eko “can-utils” paketea instalatuko dugu:

```
apt-get install can-utils
```

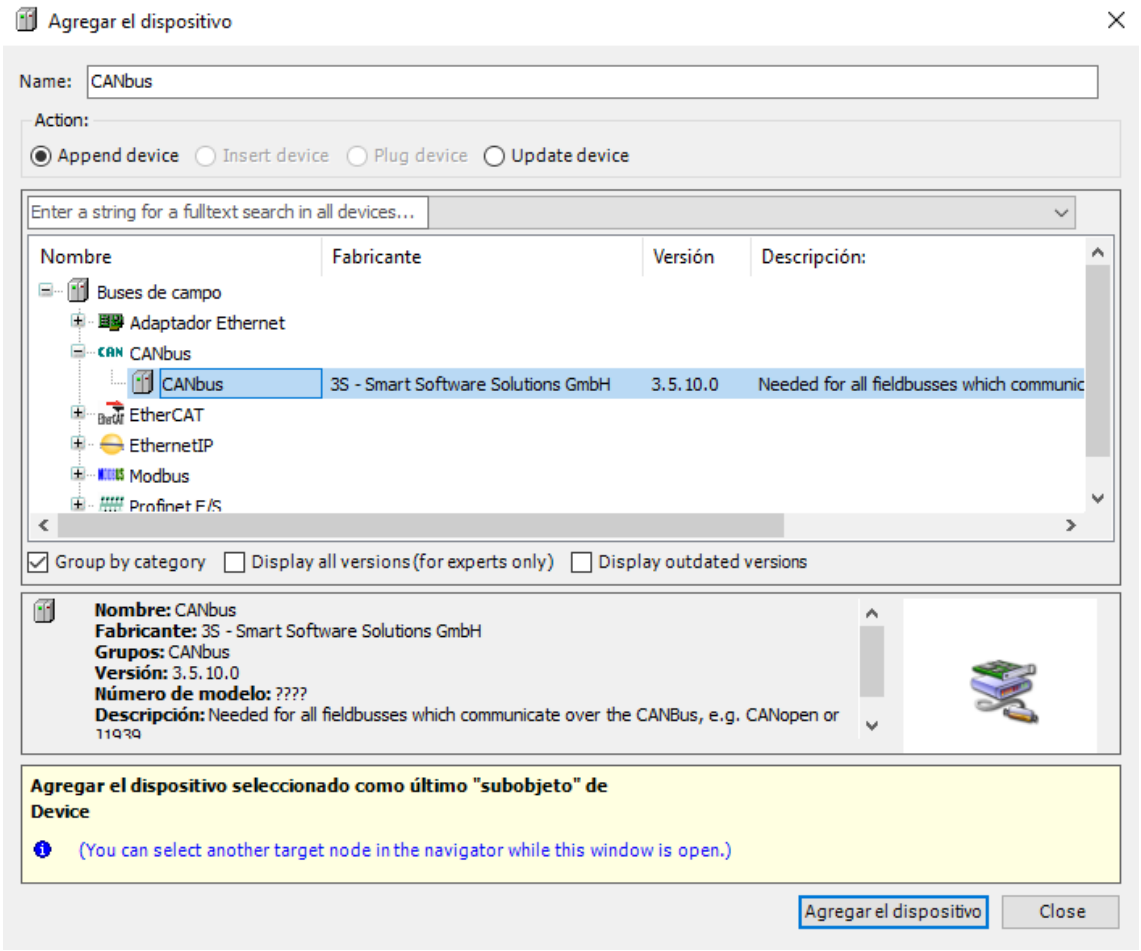
Honekin nahikoa izango litzateke CANopen-a konfiguratzeko. Orain, gure CAN CAPE-a BBB-ra konektatuko da dituzten pinen bitartez (Irudia 84).



Irudia 84: RS485 CAN CAPE-a BBB-an konektatuta.

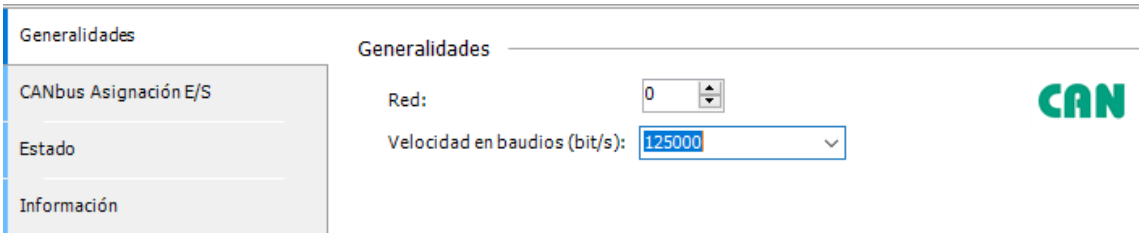
#### 3.3.3.3. CANopen funtzionamendua

Behar den kapa BBB-an konektatu ondoren, CODESYS-ekin jarriko gara martxan. Proiektuan “Dispositivos”-en “Device”-ren gainean eskuineko botoia sakatuz “Agregar el dispositivo” sakatuko da ondorengo leihoa irekiz.



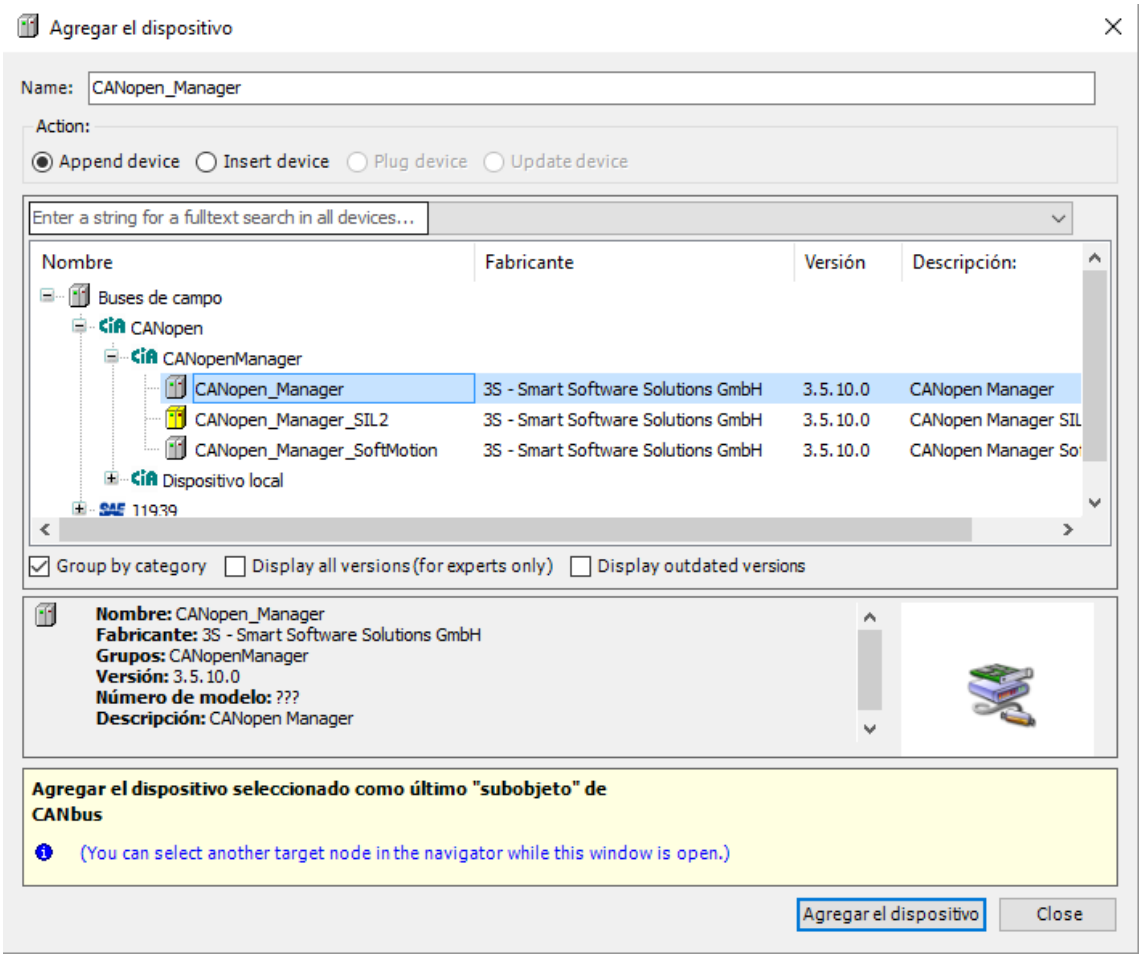

Irudia 85: CANbus-erako “Agregar el dispositivo” leihoa.

“Agregar el dispositivo” sakatu eta gero, “dispositivos”-eko “CANbus”-ean sartu eta “generalidades”-en “Velocidad en baudios”→125000bit/s jarriko da, lehen BBB-ko terminalean jarri dugun datu berdina izanik.



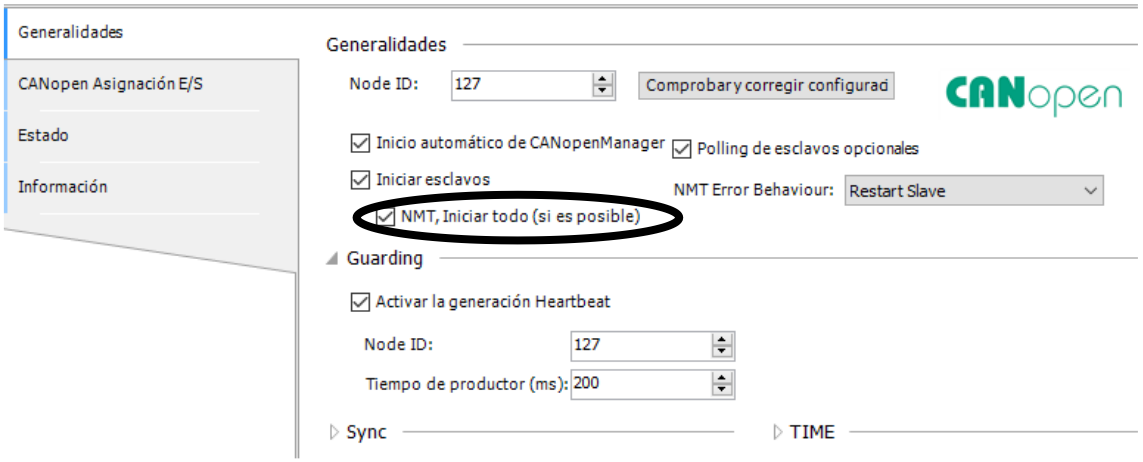

Irudia 86: CANbus-eko “Generalidades” leihoa.

Ondoren, “dispositivos”-eko “CANbus”-en gainean eskuineko botoiarekin sakatu eta “Agregar el dispositivo” klikatu behar da “CANopen_manager” aukeratuz ondorengo irudian ikusten dena.




Irudia 87: "CANopen_Manager" instalatzeko "Agregar el dispositivo".

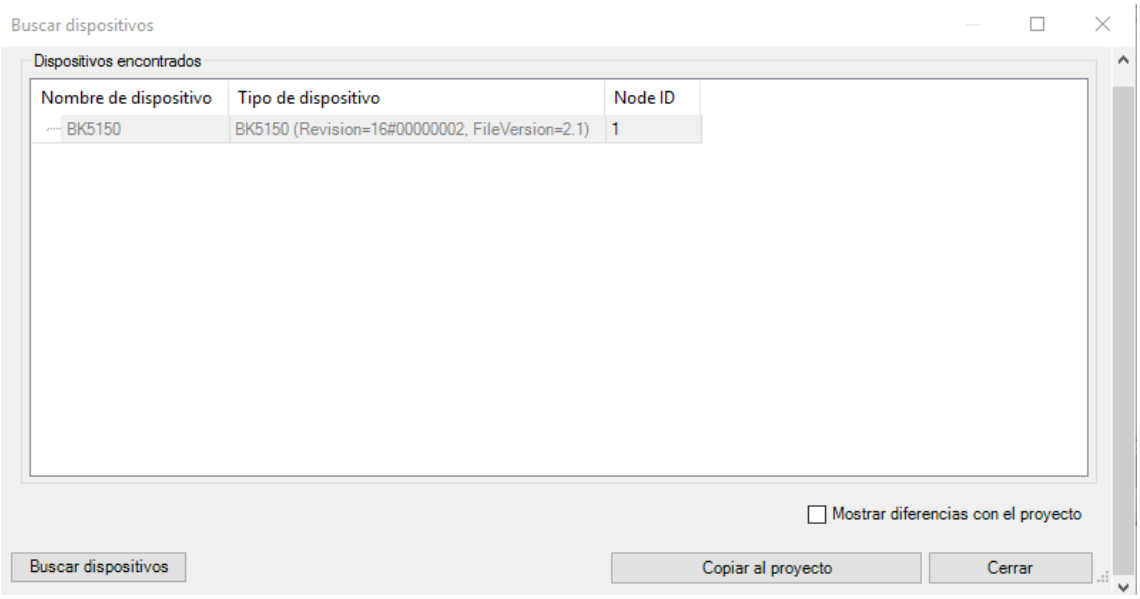
"CANopen_manager"-en gailua ezartzean, bertako "Generalidades"-en sartu eta aldaketa bat egin behar da. "NMT, Iniciar todo" hasieran baiezkorik gabe dago, orduan horri ere baiezkoa ezarriko diogu.




Irudia 88: "CANopen_Manager"-eko "Generalidades" leihoa.

Ondoren, "Iniciar sesión" egingo dugu eta "CANopen_Manager"-en gainean eskuineko botoiarekin sakatuz "Buscar dispositivo"-ri sakatuko da. Horretarako, BECKHOFF-etik gure moduluaren .eds fitxategia deskargatu beharko litzateke eta hori

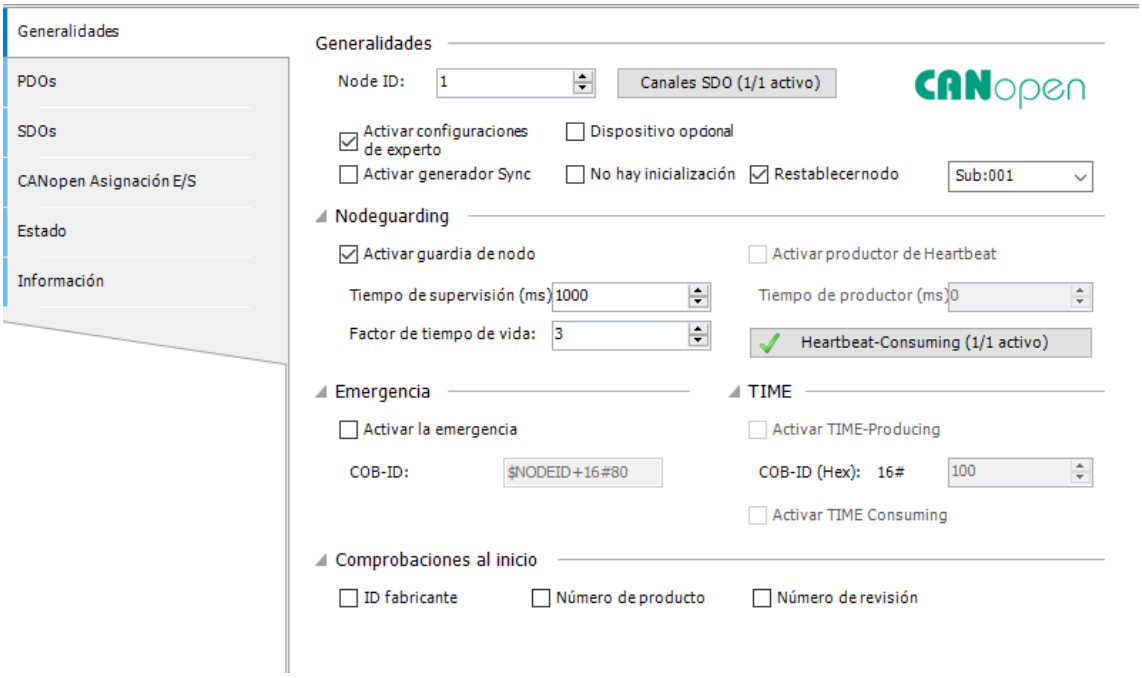
"Herramientas"→"Respositorio de dispositivos"→"Instalar" eginez gure .eds-a aukeratu eta CODESYS-en instalatuko da. Honela "Buscar dispositivo"-ren bitartez bilatzeko aukera izango genuke eta guk deskargatutako .eds-a agertuko litzaiguke hurrengo irudian ikusten den bezala (Ikus erreferentzia [20]).




Irudia 89: "CANopen_Manager"-eko "Buscar dispositivos" leihoa.

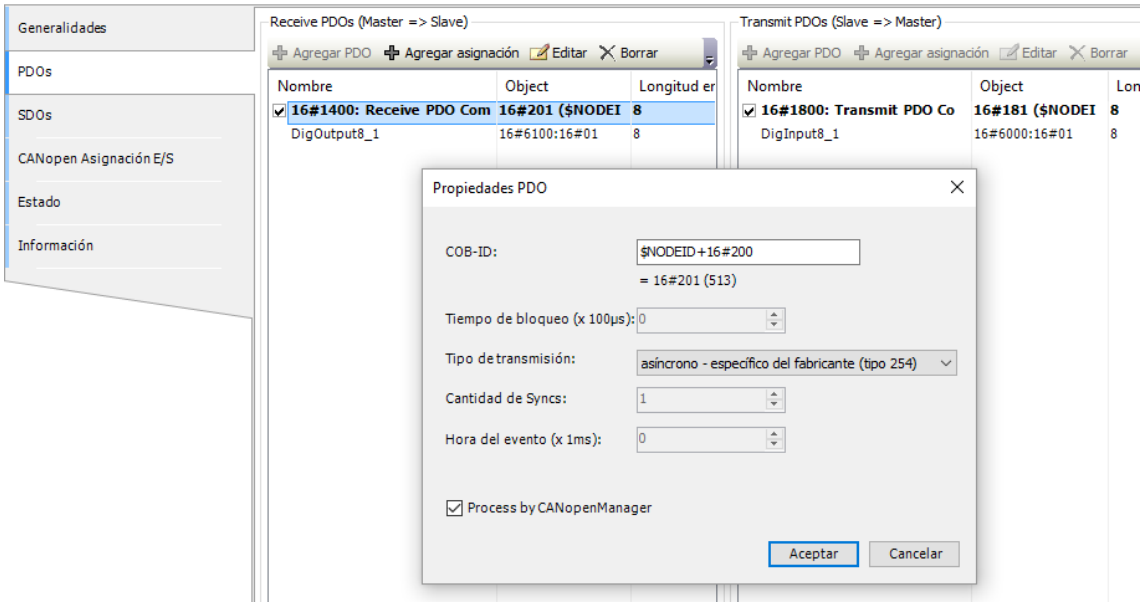
"CANbus"-ean modulu printzipalaren izena bakarrik agertzen da. Hau da, CODESIS-i konektatuta dauzkan S/I moduluak zeintzuk diren esan behar zaio. Gure kasuan, S/I digitalekin bakarrik egingo dugu lan.

BK5150-aren "Generalidades"-en barruan datuak aldatu behar dira, hau da, hurrengo irudian ikusten den bezala konfiguratu behar da.




Irudia 90: "BK5150"-ren "Generalidades" leihoa.

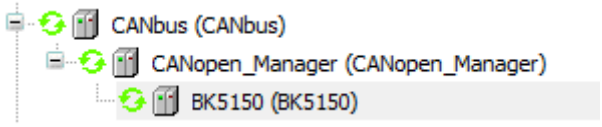
Leiho horretako hurrengo atalean, hau da “PDOs”-n, gure modulu printzipalak dauzkan modulu esklabuak izendatuko ditugu.




Irudia 91: “BK5150”-ren “PDOs”-eko “Propiedades PDO” leihoa.

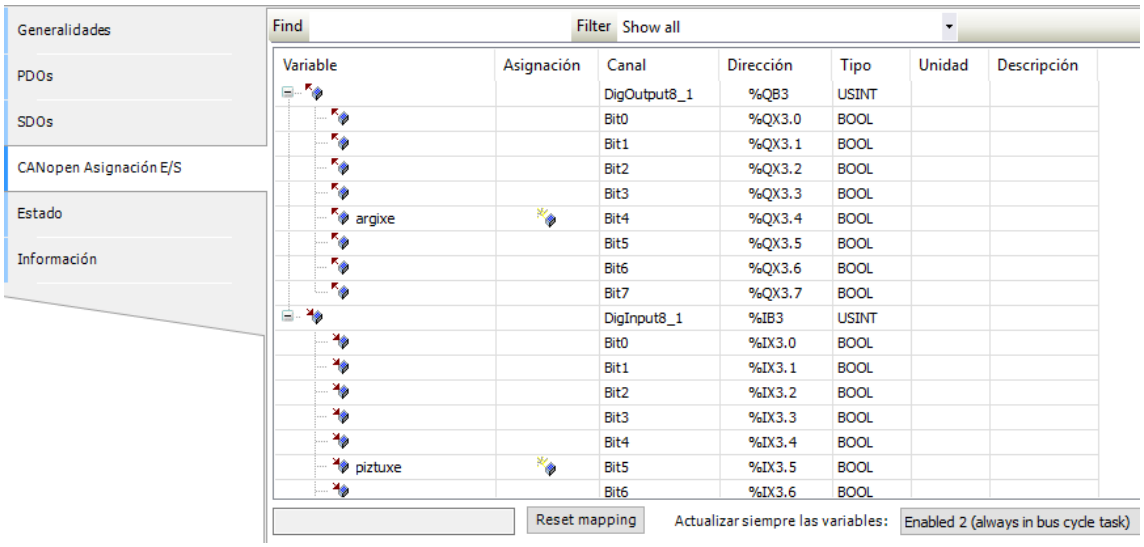
91. irudian agertzen den leihoan, “Receive PDOs”-ak irteerak dira, “Transmit PDOs”-ak ordea, sarrerak. 16#1400 eta 16#1800 aukeretan eskuineko botoiarekin editatzeko agindua eman behar zaio eta “Tipo de transmision”→”tipo 254” jarri behar da. Kasu honetan, datuak ezin dira sinkronoki bidali. Hau da, sarrerako pina aktibatzean bidaliko du datua. Pina aktibatuta geratzen bada ez du datu gehiagorik bidaliko berriz desaktibatu eta aktibatu arte.

Modulu esklabuak konfiguratu ondoren, “Iniciar sesión” egin eta denak berdez agertzen badira arazorik ez dagoela esan nahi du.




Irudia 92: “Dispositivos”-eko CANopen atala konektatuta.

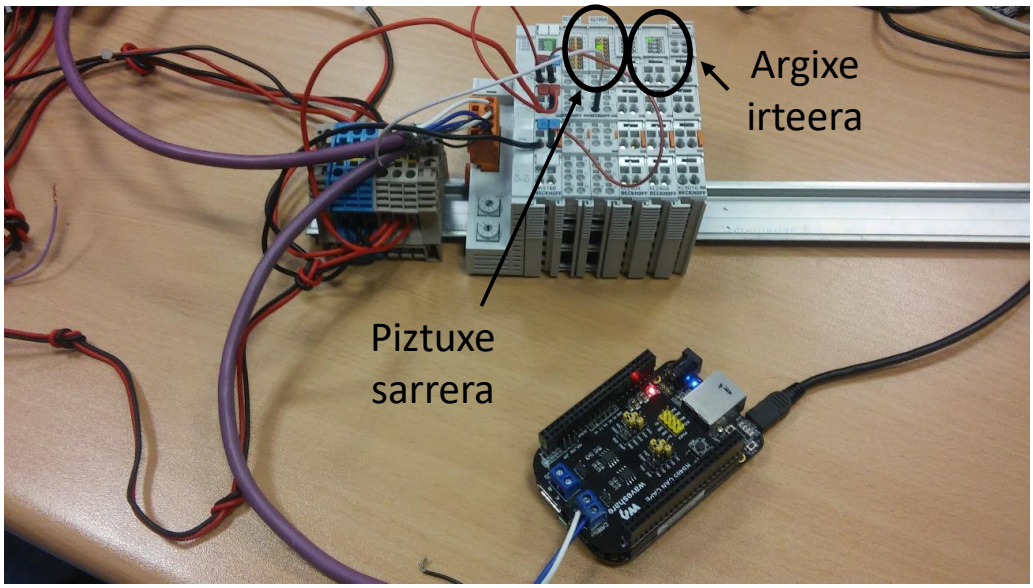
Orain, pinen asignazioa egingo da eta horretarako “CANopen Asignación E/S”-en sarrera eta irteerari izena jarriko digu bakoitzaren bita aukeratuz. Gure kasuan, irteerari dagokionez, Bit4 kanala aukeratu da eta “argixe” izena ezarri zaio, non moduluko “KL2404-2” gailuan lehenengo pina aktibatuko den. Sarrerari dagokionez, Bit5 kanala aukeratu da eta “piztuxe” izena ezarri zaio, non moduluko “KL1804-2” gailuan hirugarren pina aktibatuko den.




Irudia 93: “BK5150”-ko “CANopen Asignación E/S” leihoa.

Azkenik, moduluan konfiguratutako sarrerako pina (piztuxe) aktibatzean, irteerako pina (argixe) programan eta moduluan piztuko litzateke.


Irudia 94: CANopen-a frogatzeko egindako programa.




Irudia 95: CANopen-eko modulu digitalak eta BBB-a martxan egindako programarekin.

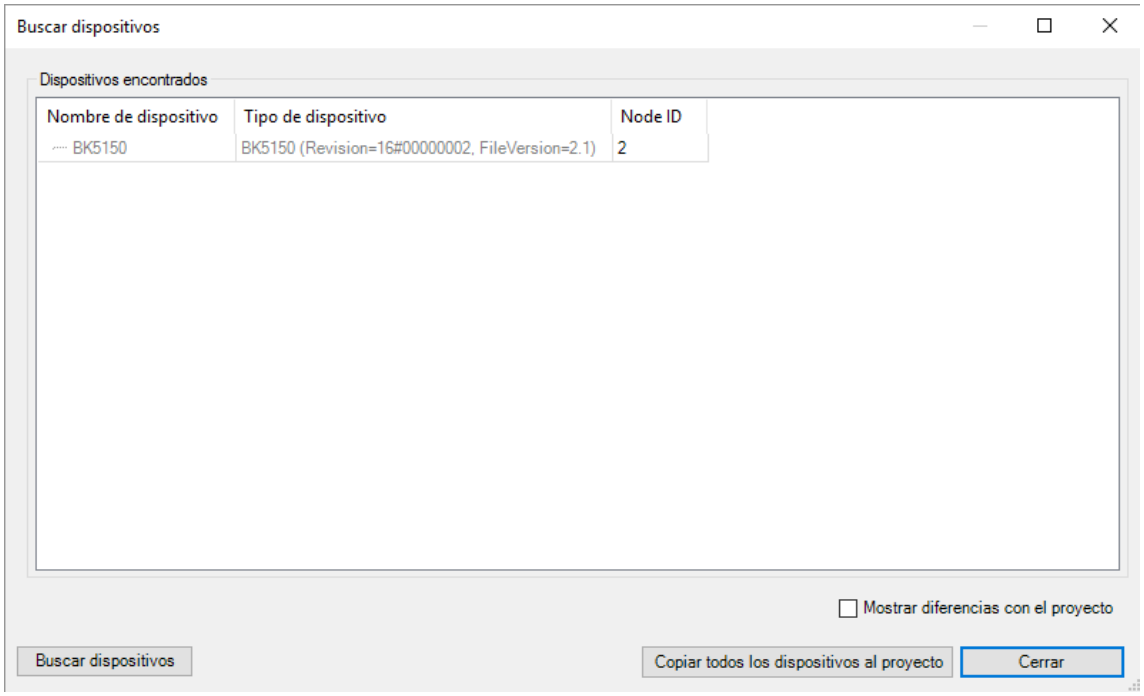
S/I digitalekin ondo doala ikusi ondoren, S/I analogikoekin probatuko dugu. Ondorengoak dira gure S/I analogikoak:

KL3162 Sarrera analogikoak
KL4004 Irteera analogikoak

Horretarako, lehendik konfiguratutako “can0”-a aktibatzeko ondorengo bi komandoak erabiliko ditugu:

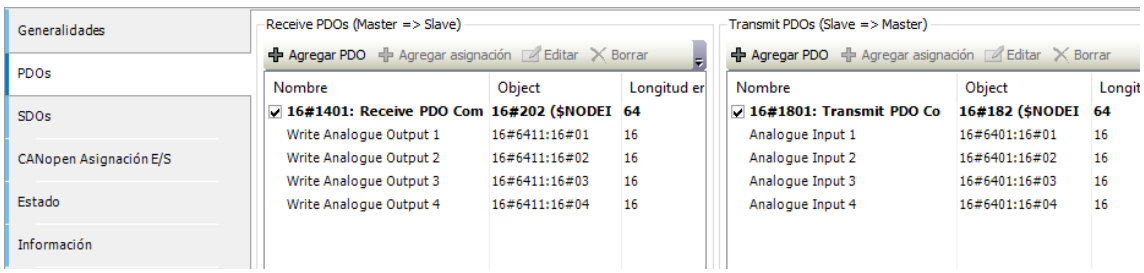
```
ip link set can0 up type can bitrate 125000

ifconfig can0 up
```

Honela “RS485 CAN CAPE”-a jarriko diogu eta S/I digitalen moduluak S/I analogikoen moduluengatik ordezkatuko ditugu. Orain, “CODESYS”-en aurreko pausu berdinak egin behar ditugu; baina, “Node ID”-a 2 jarriko dugu S/I digitalean 1 erabili dugulako. Orduan, “Buscar dispositivos” egitean ondorengo moduan agertuko da:




Irudia 96: CANopen-eko “Buscar el dispositivo” leihoa.

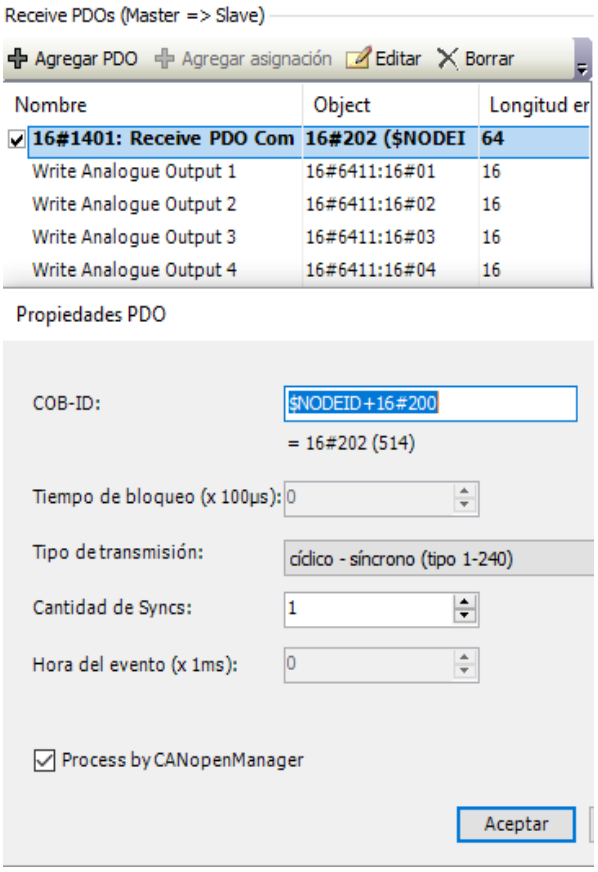
Ondoren, S/I digitalarekin ezberdinak izango diren datu bakarrak “PDOs” eta “CANopen asignación E/S” izango dira. Hau da, “BK5150”-ko “Generalidades” leihoa bi kasuetarako berdina izango da.
Kasu honetan, sarrera eta irteera analogiko bakarrak izango ditu moduluak, orduan “Receive PDOs” eta “Transmit PDOs” bakarrak egongo dira; hau da, 1401 eta 1801.



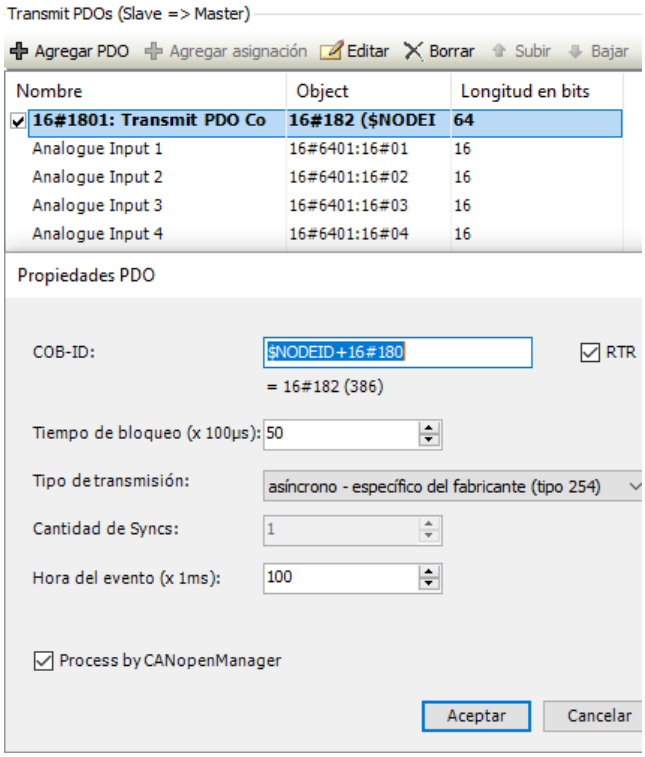

Irudia 97: “BK5150”-ko “PDOs” leihoa.

Orain, 1401 eta 1801-n gainean eskuineko botoiarekin klikatuz, “editar” egingo dugu. Irteeran (1401), “cíclico (tipo 1-240)” jarriko dugu. Sarreran (1801) ordea, “asíncrono (tipo 254)” jarriko da. Gainera, “hora del evento→100” jarriko da. Honek, zenbat

milisegunduro bidaliko duen datua esan nahi du. Ondorengo irudietan ikus daitezke beraien propietateak.

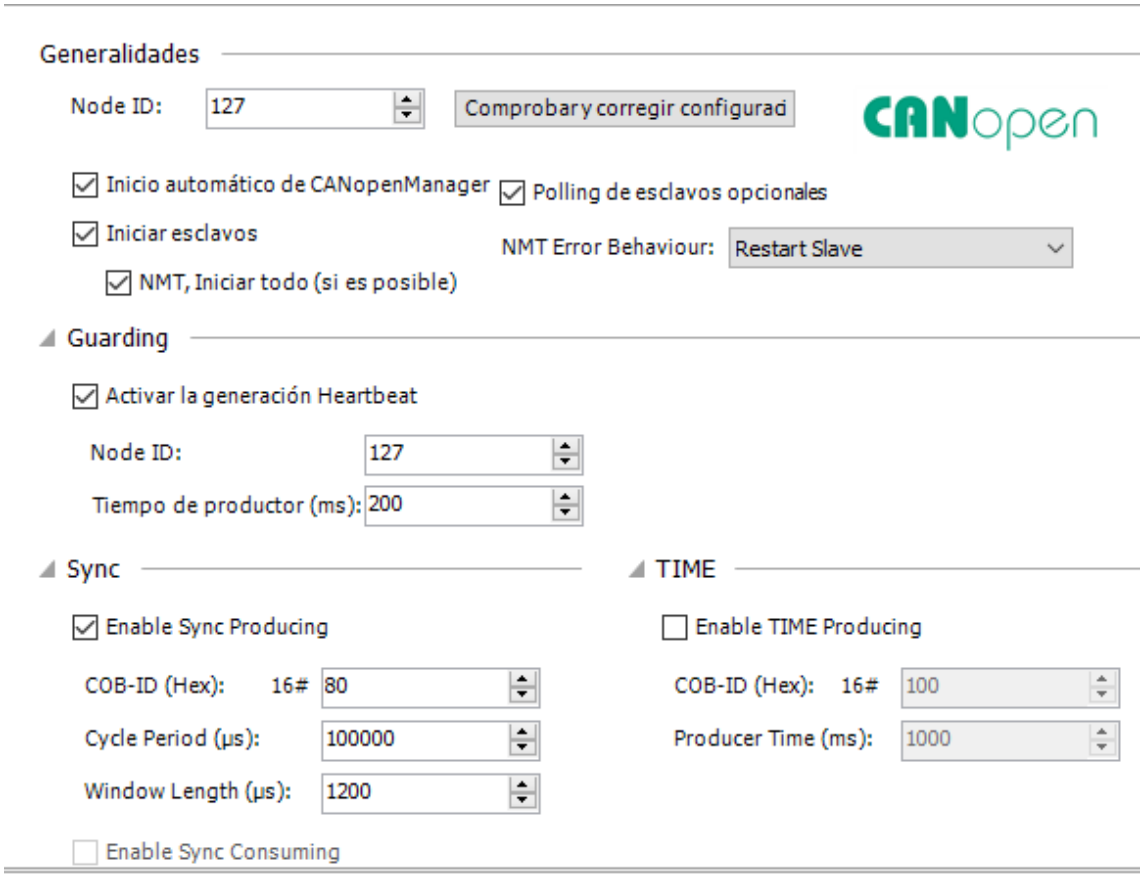



Irudia 98: “Receive PDOs”-ren “Propiedades PDO” leihoa.



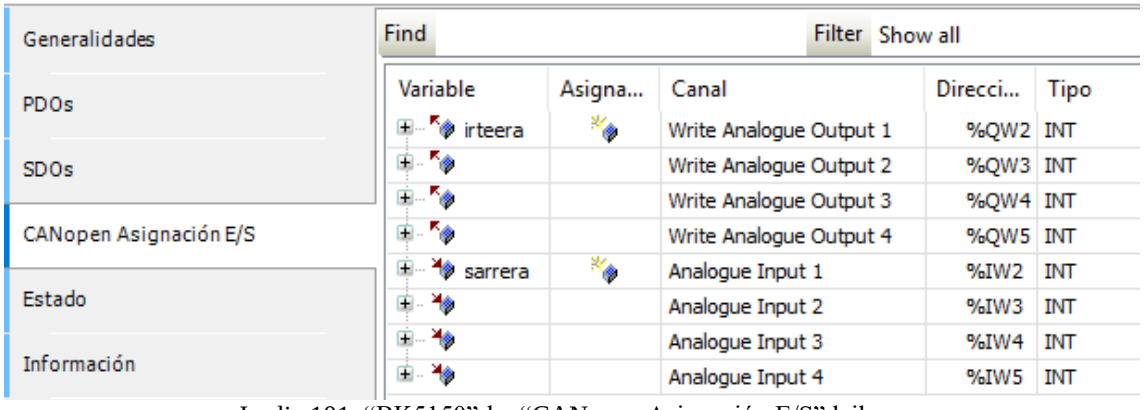

Irudia 99: “Transmit PDOs”-ren “Propiedades PDO” leihoa.

Sinkronoki egin dezan, “CANopen_manager”-eko “Generalidades”-en sartu behar da. Bertan “Cycle Period→100000” jarriko da. Honen bitartez, datuak ez ditu azkar bidaliko eta honela, bere kapazitatea ez gaindituko eta errazago ikusiko da sinkronoa. Ondorengo da konfigurazioa:




Irudia 100: “CANopen_Manager”-eko “Generalidades” leihoa.

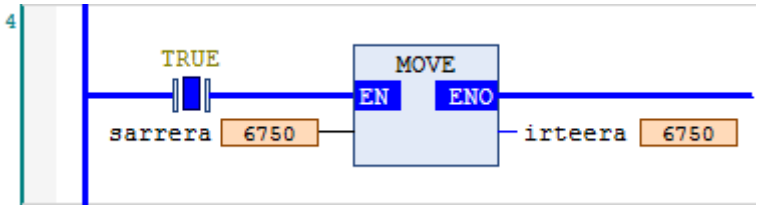
Dena konfiguratu ondoren, sarrera eta irteera izendatu behar dira BK5150-ren barruan.



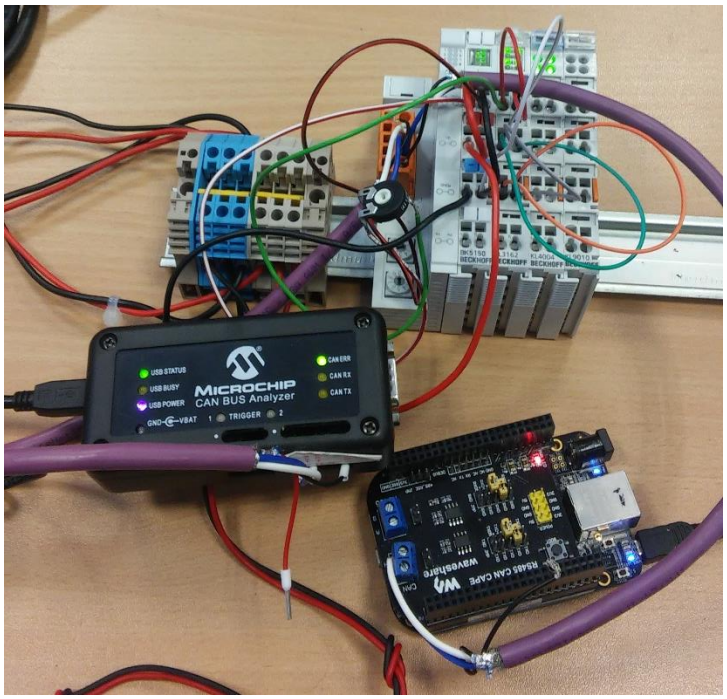

Irudia 101: “BK5150”-ko “CANopen Asignación E/S” leihoa.

Ondoren, potentziometro bat moduluari gehituko zaio, horrela sarrera analogikoa irakurtzean balio hori irteera analogikoan ikusiko da ondo funtzionatzen badu.

Horretarako, MOVE agindua duen blokea erabili dugu sarrerako balioa, hau da potentziometroak ematen duena, irteeran ikusteko. Ondorengo irudian ikus daiteke ondo funtzionatzen duela modulu analogikoarekin.




Irudia 102: Sarrera eta irteera analogikoak frogatzeko egin den programa.




Irudia 103. CANopen modulua potentziometroarekin eta BBB-a martxan.

Orduan, potentziometroa mugitu ahala, sarrerako balioa aldatuko litzateke eta honen ondorioz, irteerakoa ere aldatuko da. Hau da, beti datuak bidaltzen egongo litzateke.

Analizatzeko ondo doan edo ez CAN BUS Analyzer bat erabiliko da:

CAN BUS Analyzer

File View Tools Setup Help

Fixed Trace

| TRACE | ID | DLC | DATA | DATA | DATA | DATA | DATA | DATA | DATA | DATA | TIME STAMP (sec) | TIME DELTA (sec) | COUNTER |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RX | 0x80 | 0 | | | | | | | | | 197,6943 | 0,100 | 2121 |
| RX | 0x202 | 8 | 0x34 | 0x26 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 0x00 | 197,6953 | 0,100 | 2121 |
| RX | 0x182 | 8 | 0x34 | 0x26 | 0x3E | 0x26 | 0x3E | 0x26 | 0x3E | 0x26 | 197,7454 | 0,090 | 2291 |
| RX | 0x77F | 1 | 0x05 | | | | | | | | 197,5954 | 0,200 | 1061 |
| RX | 0x702 | 1 | 0x85 | | | | | | | | 197,7104 | 0,000 | 426 |

Irudia 104: CAN BUS Analyzer-en ikusitako emaitzak.

Bertan agertzen diren ID-en azalpen txiki bat hurrengoa da:

0x80: sinkronismoaren denbora da eta CODESYS-ean jarritakoarekin bat egon behar du.

0x202 PDO-a: irteera analogikoak aktibatzeko PDO-a da. Maisuak, esklabuari bidaltzen dio.
0x182 PDO-a: Maisuari esklabuak bidaltzen dion irakurketa analogikoa.
0x702: nodeguarding-a
0x77F: Heartbeat

Esklabu txartelak 0x182 PDO-a bidaltzen du. 4 irakurketa analogikoen emaitza bidaltzen du. Lehenengo 2 bytetan potentziometroaren irakurketaren emaitza dago, hau da 0x3426 irakur daiteke (IRUDIA 104). Informazio hau PLC-ra heltzen da eta programaren arabera irteera analogiko batera bidaltzen da. Hau 0x202 PDO-an ikusten da. Bakarrik lehenengo 2 byte-ak bidaltzen dira programan eskatu bezala, alegia, 0x3426 bidaltzen da.

# 4. ONDORIOAK

Azken atal honetan, proiektu guztian zehar lorturiko ondorio nagusiak zerrendatuko dira. Orrialde eta atal desberdinetan zehar BeagleBone Black eta CODESYS-en arteko prozesua aurkeztu da. Bertan, denbora errealean funtzionatzen duen ikusi da eta gainera hiru eremu busak frogatu dira.

Lehenik eta behin, BBB-an sistema eragileen hainbat irudi ezberdin frogatu dira. 3.2.4. atalean ikus daitekeen bezala, irteera digital bat erabiliz Jitterra neurtu da sistema eragile desberdinekin. Kasu guztietan ikusi da enpresaren beharrak ez dituela betetzen, eta denbora errealean enpresak behar duen bezala arazo ugari ekar ditzake. Linux-aren kodea ezin dugunez aldatu horrela geratu da sistema. Arazo hau konpontzeko, Linux-a programatu eta sistema eragile espezifikoa egiteko gai den programadore batek aldaketak egin beharko lituzke helburu bezala enpresaren beharrak kontuan hartuz, nagusiki denbora errealeko eskakizunak betez Jitter onargarria lortzeko.

Bestetik, enpresak EtherCAT eta PROFINET eremu busekin bakarrik egin nahi zituzten frogak. Dutt empresan ordea, bus eremu nagusi bezala CANopen erabiltzen dutenez eremu bus hau konfiguratzea ere eskatu zidaten. Beraiek ez zekiten BeagleBone Black-arekin funtzionatzen zuen. Orduan, informazioa bilatu eta gero, RS485 CAN CAPE-a konektatuz BBB-ari frogak egin eta gero sistema martxan jarri nuen. Hasieran, kontaktu fisikoarekin bakarrik konektatuz ikusi nuen ez zuela funtzionatzen. Orduan, BBB-aren terminalean konfigurazioak egin behar zirela konturatu nintzen eta horrela, CANopen eremu busa ere konektatzea posible zela frogatu nuen.

Froga guztiak egiteko CODESYS softwarea erabili da. Software honek IEC 61131-3 estandarra betetzen du eta PLC industrialak programatzeko erabiltzen da. Programazioa lantzea baino sistemaren konfigurazioan lana handia egin da eremu busak frogatzen. Horrela CODESYS-en bidez eta eremu busak erabiliz hainbat dispositibo kontrolatu dira. Hala nola, sarrera/irteera digital eta analogikoak eta SINAMICS driverra.

# 5. ERREFERENTZIAK

## 5.1. Web orrialdeak

Enpresa

[1]
http://www.duttelectronics.com/

CODESYS

[2]
https://www.codesys.com/the-system.html

[3]
https://en.wikipedia.org/wiki/CODESYS

[4]
https://store.codesys.com/codesys.html?___store=default

[5]
https://store.codesys.com/codesys-control-for-beaglebone-sl.html

Denbora erreala

[6]
http://isa.uniovi.es/docencia/TiempoReal/Recursos/temas/sotr.pdf

[7]
https://forum.codesys.com/viewtopic.php?f=26&t=5652&start=30

BeagleBone Black

[8]
https://elinux.org/Beagleboard:BeagleBoneBlack#BeagleBone_Black_Description

[9]
http://beagleboard.org/getting-started

[10]
http://www.farnell.com/datasheets/1819364.pdf?_ga=2.187153256.1198084811.1494233619-1646089628.1479322053

[11]
http://derekmolloy.ie/set-ip-address-to-be-static-on-the-beaglebone-black/

EtherCAT

[12]

https://www.ethercat.org/en/technology.html#1.8.4

[13]
https://en.wikipedia.org/wiki/EtherCAT

[14]
https://www.beckhoff.com/english.asp?ethercat/el1004.htm7

PROFINET

[15]
https://support.industry.siemens.com/cs/document/19699080/archivos-gsd-para-profinet%3A-i-o-et-200s?dti=0&lc=es-WW

[16]
https://cache.industry.siemens.com/dl/files/131/30598131/att_80298/v1/et200s_im151_3_pn_manual_es-ES_es-ES.pdf

[17]
https://www.industry.siemens.com/automation/aan/es/industrial-communications/profinet/pages/default.aspx

[18]
http://www.infoplc.net/documentacion/7-comunicaciones-industriales/74-informe-ethernet-industrial

CANopen

[19]
https://e-archivo.uc3m.es/bitstream/handle/10016/14680/MEMORIA%20PFC%20CONRADO%20GRANDAL.pdf?sequence=1

[20]
https://www.beckhoff.com/english.asp?bus_terminal/bk5150_bk5151.htm

[21]
https://www.waveshare.com/wiki/RS485_CAN_CAPE#RS485_CAN_CAPE

[22]
https://developer.ridgerun.com/wiki/index.php/How_to_configure_and_use_CAN_bus

## 5.2. Liburuak

[23] Mark A. Yoder & Jason Kridner: BeagleBone Cookbook. O´Reilly. Apirilak 2015.