Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Fakultatea
Informatika Ingeniaritza

▪ **K**arrera **A**maierako **P**roiektua ▪

NXT eta ROS BIDEZKO ZABOR BILTZAILEA

Ziortza Gallo Ipiña

2015 - ekaina

informatika fakultatea
facultad de informática

Eskertzeak

Oso luzea egin da proiektua amaitzeko bidea, jende asko pasa da nire bizitzatik eta egoera ezberdin asko bizi ditut urte hauetan. Robotika ikasi ez ezik, nire bizitzako arlo askotan eragina izan du proiektuak.

Nire familia eta lagunak oso garrantzitsuak izan dira karrera eta proiektua amaitzeko garaian. Pazientzia handia izan dute eta, gainera, buruhausteak izan ditudanean niri animatzeko lehenak izan dira.

Familiagatik beti espero duzu holako portaera, horregatik proiektu honetan nire eskertzerik garrantzitsuena, Elenarentzat da. Nahiz eta desastre hutsa izan naizen instalazioak egiten, Elena beti laguntzeko prest egon da. Bere laguntzarik gabe ez nuke proiektu hau inoiz amaituko.

Azkenik, eskertze txiki bat eman behar diot CampTecnologikoari. Materiala utzi didalako eta, gainera, beraiekin lan egiteko aukerari esker, robotaren portaerak ezagutu nituelako.

Laburpena

Proiektu honetan, LEGOko NXT plaka eta ROS sistemaren arteko konektagarritasuna landu da.

ROS (*Robot Operating System*) sistema, egun, robotikan ikertzeko tresna nagusia da. Tresna honekin, robotaren programazioa garatuko da.

LEGO NXT Mindstorm robotika irakasteko oinarrizko teknologia da. LEGO pieza ezberdineko kit bat da, non sentsoreak eta robotaren plaka eskaintzen dituen. Honek, proiektuaren robota eraikitzeko erabiliko da.

Teknologia hauen bateragarritasuna frogatzeko, “zabor biltzaile” baten portaera inplementatu da. Alde batetik, robotaren morfologia diseinatuz eta eraikiz, eta beste aldetik robotaren mugimenduak garatu dira.

\- -

The aim of this project was to try and test the connectivity between the LEGO NXT brick and the ROS framework.

Nowadays, ROS (*Robotic Operating System*) is one of the most widely used tools for robotic research. That’s why It was chosen for developing the software of the robot.

LEGO NXT Mindstorm, the main technology is used for introducing robotics to students. That kit provides us the brick, the sensors and different pieces of LEGO. And so far, it has been selected to build the robotic platform used during the project.

In order to test the compatibility of those tools, the robot has been assigned a task: It must show a rubbish collector behaviour.

HITZ GAKOAK: NXT, ROS, Robotika, zabor biltzailea

Gaien Aurkibidea

Irudi, Taula eta Kode Aurkibidea

Irudiak

1. KAPITULUKO IRUDIAK:

2. KAPITULUKO IRUDIAK:

3. KAPITULUKO IRUDIAK:

4. KAPITULUKO IRUDIAK:

6. KAPITULUKO IRUDIAK:

Taulak

2. KAPITULUKO IRUDIAK:

3. KAPITULUKO IRUDIAK:

4. KAPITULUKO IRUDIAK:

Kodea

3. KAPITULUKO IRUDIAK:

4. KAPITULUKO IRUDIAK:

1. zatia: Sarrera

1. kapitulua:

Sarrera

1. kapitulua: Sarrera

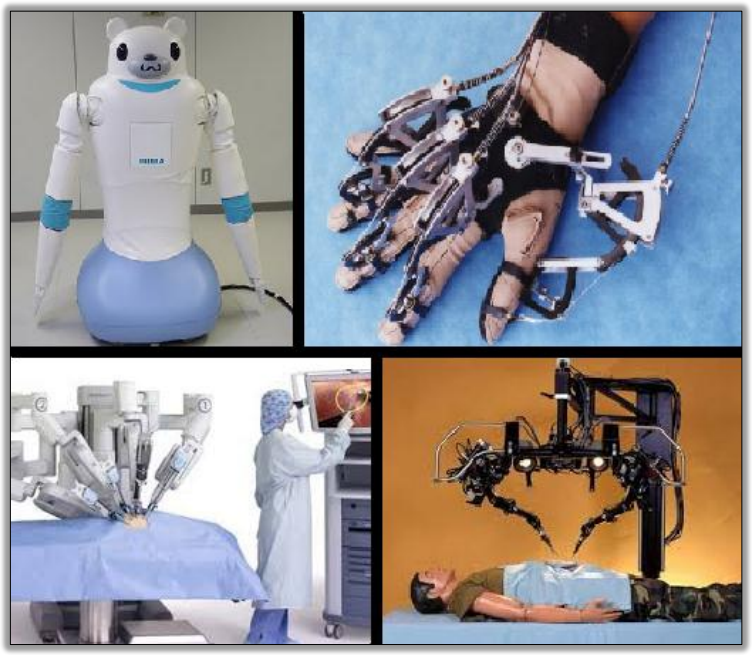
Robotak automatikoki atazak burutzen dituen gailu elektroniko eta mekanikoak dira. Robotek morfologia ezberdinak izan ditzakete eta era askotan programatu daitezke. Bete behar duten portaeraren arabera, analisi bat egiten da eta ondoren, robota diseinatu eta garatzen da.

Robotek gizakiok bete ezin ditugun atazak garatzen dituzte. Batzuetan atazaren batean zehaztasun handia behar delako, beste batzuetan maneiatu behar den piezaren pisuagatik, baita ere, ataza arriskutsuak izan daitezkeelako, bai tenperatura handiekin edo bai txinpartarekin lan egin behar delako. Robotak arlo askotan erabili daitezke, hona hemen azterketa txiki bat:

- Robot industrialak: Robot hauen funtzio nagusienak: soldadura, piezak leku batetik bestera mugitzeko eta piezak galdaketetan sartzeko dira. Hau da, industriak, produktu bat ekoizteko prozesua automatizatzeko erabiltzen dira. 1.1. irudian ikus dezakegu robot hauen morfologia ohikoa. Sentsoreak ia ez dira erabiltzen, posizio ezberdinak definitzen dira eta robota aurredefinitutako posizioetara mugitzen da.

1.1. Irudia: Robot industrialak

- Urruneko robotak: Robot mota hauek ez dira era bat autonomoak, telekomando bitartez mugitzen dira. Industrialak moduan, gizakiok bete ezin ditugun atazak betetzen dituzte normalean. Material bereziez eginda daude, bonba baten aurrean ezer gerta ez dadin edo urperatzerako garaian, urpeko presioa jasan dezaten. Arlo ezberdinetan erabiltzen dira, adibidez medikuntzan, aplikazio militarretan, urruneko planeten esplorazioan edo erreskate lanetan.



1.2. Irudia: Medikuntza arloko robotak

1.2. irudiko robotak medikuntza arloan erabiltzen dira. Urruneko ebakuntzak egiteko, zein erizaintzan gaixoak leku batetik bestera garraiatzeko. Urruneko robotak dira eta normalean telekomandoan interfaze grafiko bat izaten dute robota bideratzeko. Teknologia haptikoak ere erabiltzen dira.

1.3. Irudia: Robot militarrak

1.3. irudian robot militar mota ezberdinak ikus ditzakegu. Robot militarrak material bereziez eginda daude, kamerak izaten dituzte robotaren inguruneko irudiak jaso ahal izateko, horrela gizakiak aurrean zer duen jakingo du. Urruneko telekomandoa izaten dute eta telekomandoan pantaila txiki bat izan dezakete robotaren ingurunea bistaratzeko.

Eta, azkenik, 1.4. irudian espazioan edo urpean erabiltzen diren robotak dira. Espazioko roboten kasuan, laborategiak robotaren barnealdean kokatu daitezke. Robotak ingurunea aztertu behar duenez, material ezberdinak aztertu behar izaten dira.

1.4. Irudia: Espazio eta Urpeko robotak

- Etxeko robotak: Badaude gizakion eguneroko eginbeharrak egiten dituzten robotak. Beste arloetan ez bezala, robot hauek ez dute betetzen ataza oso arriskutsuak, gizakiok bete ditzakegun atazak betetzen dituzte, baina diseinatuta daude guri bizitza errazteko.

1.5. Irudia: Etxeko Robotak (Asimo, Roomba, Asibo)

Badaude robot mota ezberdinak arlo honetan, humanoideak, jostailuak edo xurgagailuak esaterako. Ohikoa bihurtu da gure etxeetan, 1.5. irudiko eskuin aldeko goialdeko *Roomba* xurgagailua izatea. Baina beste motako robotak, dituzten prezioengatik, oso zaila da etxeetan topatzea. Adibidez, humanoideak, ikerkuntza prozesuan daude gehienak, behar bada urte batzuetan errazagoak izango dira robot hauek ikustea.

- Hezkuntzan: Unibertsitateetan eraikitzen diren robotek ez dute produktu bat sortzen edo zerbitzu zehatz bat eskaintzen. Ikasleek, robotika munduan aurkitu ditzaketen baliabideak ikertzeko erabiltzen dira. Topa daitezke mihiztadura lengoaian programatzen diren robotak, baita goi mailako programazio lengoaia edo baita ere, C++ edo Pythonen programatzen direnak. Azken kasua izango da proiektu honen programazio eremua.

1.6. Irudia: Hezkuntzarako robotak

(Cebek, MoWay, Arduino (3, 4), Lego NXT)

Egun, nahiko eskuragarri bihurtu dira robotak. Robotak ez dira bakarrik unibertsitateetan erabiltzen, eskoletan edo udako kanpamentuetan ere irakasten dira. Umeentzako deigarri bihurtzeko, eskolen arteko jokoak izaten dira.

Umeei zuzenduta dauden programazio-lengoaiak erabiltzen dituzte robot hauek. Oso intuitiboak dira eta ia-ia puzzle baten modukoak dira. Gainera, LEGO NXT Mindstorm kit-ak, morfologia ezberdinak muntatzeko aukera ematen digu, betiko jostailuetan agertzen diren LEGO piezekin. Honek, umeak asko erakartzen ditu eta motibazio handia bihurtzen da beraientzat.

Proiektu honetan, unibertsitate eremuan erabiltzen den ROS sistema eragilea eta LEGO NXT Mindstorm robotaren arteko konektagarritasuna landu da. Baliabide hauek erabiliz, zabor biltzaile baten portaera garatu da, esperimentu gisa.

1.1. Motibazioa eta helburuak

Proiektuaren helburu nagusia NXT eta ROSen konektagagarritasuna aztertzea da, horretarako zabor biltzaile bat eraiki eta programatuko da. Robotak, hurrengo kapituluetan azalduko den moduan, latak bilatu identifikatu eta bere lekuan uzteko helburua du, bere ingurugiroan egon daitezkeen oztopoak saihestuz. Hau lortzeko, Lego NXT-ak eskaintzen dituen piezak eta sentsoreak erabiliko dira.

Software-aren aldetik, ROS erabiltzea izango da gakoa. Inoiz erabili ez dudan softwarea ikastea eta erabiltzea izango da motibaziorik handiena.

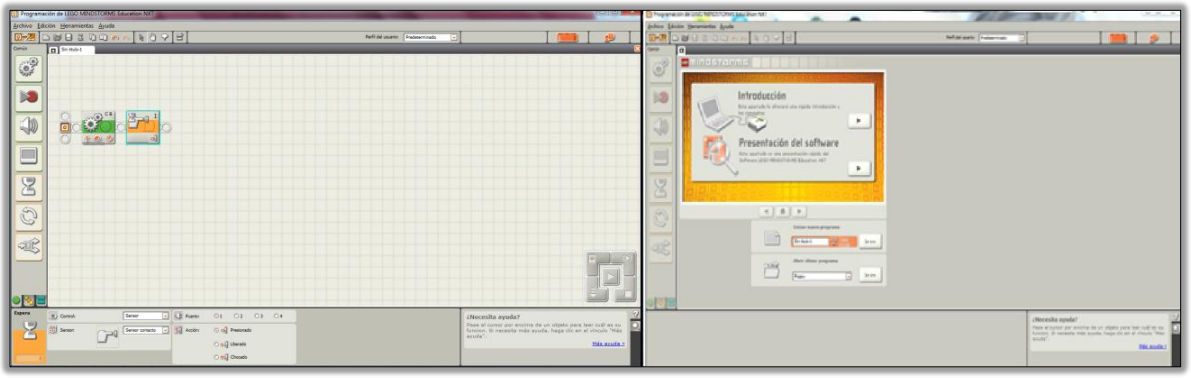
Proiektuaren antolakuntza eta zuzenketari begira, agertzen diren zalantzak edo ez jakinak ondo maneiatzea izango da helburuetariko bat. Honek, proiektua gehiago ez atzeratzea erraztuko du.

1.2. Aurrekarien analisia

Lego NXT hezkuntzan erabiltzen den robota da. Nire kasuan, CampTecnologiko udako ikastaroetan erabiltzen dut. Umeek robotika mundua ikertzeko baliabide moduan erabiltzen dugu. NXT robota eraikitzeko LEGO piezak erabiltzen dira, horrek robot eramangarria eta sendoa bihurtzen du. Lego NXT-a era askotan muntatu daiteke, 1.7. irudian ikus dezakegun moduan.

1.7. Irudia: NXT muntaketa ezberdinak

NXT adreilua programatzeko, aplikazio ezberdinak daude. 1.8. irudian ikus dezakegun ingurune grafikoa, umeekin aritzeko NXT 2.1 Programming softwarea da. Umeek erraz erabiltzeko programa da, intuizio apur batekin, robotaren mugimenduak puzzle batean itsasten dira. Software hau 8 eta 16 urte dituzten umeentzako da egokia. Programatzeko gaitasunak ez dutenek programa hau erabil dezakete, ez baita programazio lengoaia.



1.8. Irudia: NXT 2.1 Programing Softwarea

Normala den moduan, NXT robota beste lengoaiatan ere programatu daiteke, baina noski, programatzeko esperientzia apur bat edukitzea beharrezkoa da. Adibidez, *ROBOTC*-k C-z programatzeko lengoaia oinarritzat hartzen du, *leJOS NXJ* goi mailako *Open Source*-a da eta *Java*n oinarritzen da.

ROSek *nxt_ros* paketea eskaintzen du horretarako eta *python*-en dago garatua. Horregatik, nahiz eta ROS-en C++ eta *Python*-en programatzea dagoen, *Python* aukeratu da proiektua garatzeko oinarrizko lengoaia gisa.

2. kapitulua:

Proiektuaren kudeaketa

2. kapitulua: Proiektuaren kudeaketa

2.1. Proiektuaren helburu dokumentua

2.1.1. Deskribapena eta helburua

Proiektuaren helburu nagusia, ROS eta NXTren arteko konektagarritasuna aztertzea da. ROSek eskaintzen duen *nxt_ros* paketea aztertu eta funtzionamendua probatzea izango da lehengo urratsa.

Konektagarritasuna frogatua eta aztertua dagoela, robotak bete beharreko portaera bat diseinatu eta inplementatuko da. Lehendabizi portaeraren eszenatokia definitu beharko da, gero robotak izango duen morfologia diseinatu eta eraiki. Robota eraiki ondoren izango dituen kasu egoerak pentsatu eta inplementatzen hasi behar da.

Baina hau egin aurretik proiektuan emango diren kudeaketa prozesuak deskribatuko dira atal honetan. Proiektuaren aurrekontua egin behar da, proiektuaren estimazioa eta planifikazioa egin eta proiektuan zehar gerta daitezkeen arriskuak identifikatu etorkizun batean hauek saihesteko.

2.1.2. Norainokoak

Hurrengo atalean proiektuak izango dituen atazak eta dokumentuak deskribatzen dira. Emangarrien zerrenda eta atazak azpi-atazetan zerrendatuko dira, proiektuaren estimazioa egin ahal izateko eta hobeto kudeatzeko.

Emangarrien Zerrenda:

- Proiektuaren Helburu Dokumentua (PHD)

    - Deskribapena eta helburua: Proiektuaren kokapena.

    - Norainokoak: Proiektuan zehar zer lan egingo den eta zer lan ez den egingo definitzen da.

        1) *Emangarrien zerrend*a: Proiektuaren atal bat amaitutzat emateko entregatu behar den dokumentuen zerrenda.

        2) *Azpiatalen zerrenda:* Proiektuaren ataza guztien zerrenda.

        3) *LDE diagrama:* Proiektuaren norainokoa definitzen eta multzokatzen duen adierazpen egituratua.

    -Planifikazioa: Ataza bakoitza bukatzeko estimaturiko denbora.

        1) *Gantt Diagrama:* Planifikaturiko ekintzen egutegiaren adierazpen grafikoa.

    - Arriskuak: Arriskuen identifikazioa eta beraien aurrezaintza eta kontingentzia planak deskribatuko dira.

    - Lan metodologia: Lana eta artxiboak nola banatzen, kudeatzen eta antolatzen diren deskribatzeaz gain, erabakiak nola hartuko diren ere azaltzen da.

- Memoria: Proiektu guztia deskribatzen duen dokumentua.

Azpi-atazen zerrenda:

- *Formakuntza:*

    -F – Formakuntza

        + F1 ROS
        + F2 *ros_nxt*
        + F3 Linux-en trebatu

- Kudeaketa prozesuak:

  -K – Kudeaketa

  + K1 Bilerak

  + K2 Artxiboen kudeaketa

  * K21 Segurtasun kopiak burutu

  -P – Planifikazioa

  + P1 PHD-a egin

  *P11 Deskribapena eta helburua zehaztu

  * P12 Norainokoak

  **P121 LDE-a egin

  **P122 Azpi-atazen zerrenda egin

  **P123 Emangarrien zerrenda egin

  *P13 Denbora plangintza

  **P131 Gantt diagrama egin

  **P132 Denbora estimazioak egin

  *P14 Arriskuen identifikazioa

  **P141 Zerrenda egin

  **P142 Kontingentzia plana egin

  *P15 Lan metodologia zehaztu

  *P16 Aurrekontua

  +P2 Birplanifikatu

- Garapen prozesuak:

  -G – Garapena

  +G1 Eskakizunen Bilketa egin

  +G2 Analisia egin

  +G3 Robot Muntaketa

  +G4 Inplementazioa burutu

  +G5 Probak egin

- Dokumentazioa:

  +D1 Memoria Idatzi

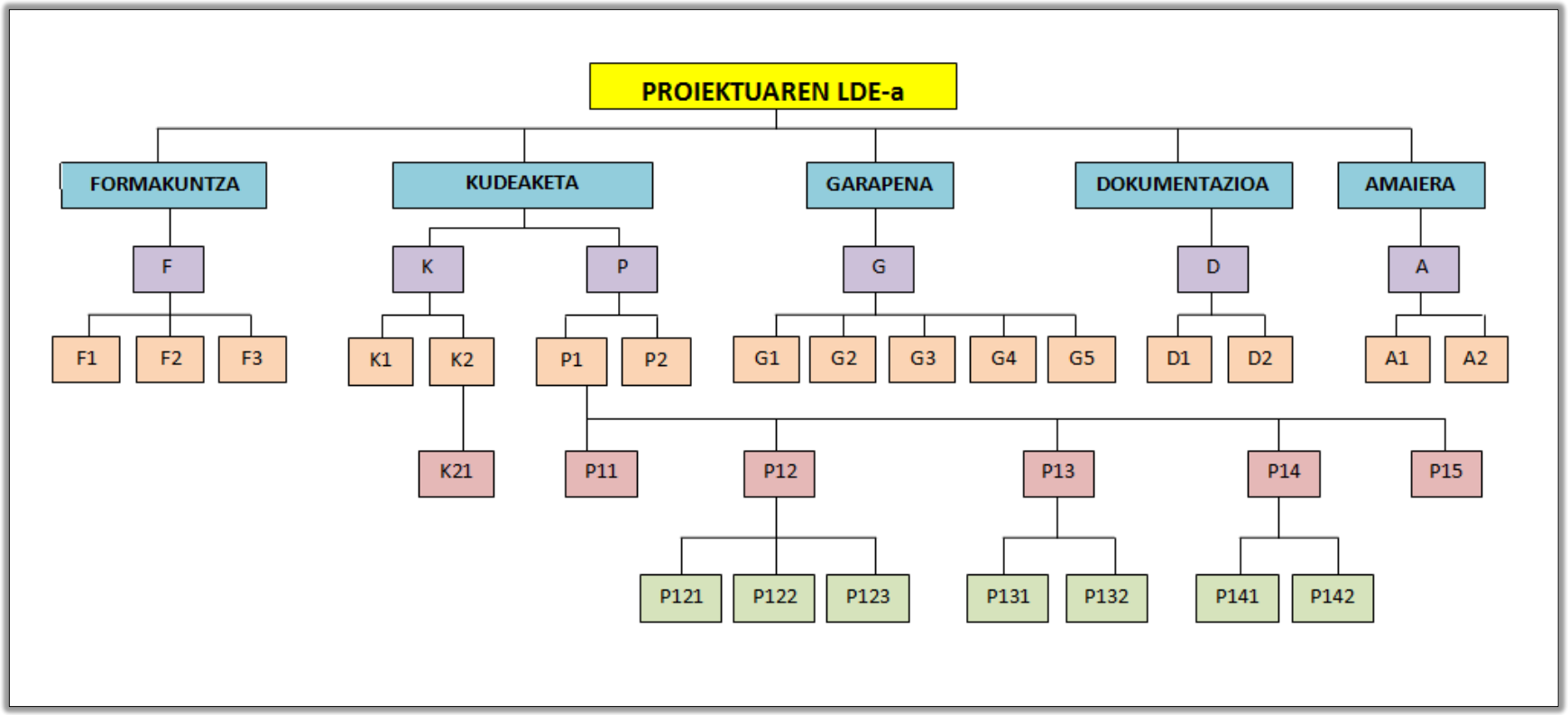
  +D2 Memoriaren Zuzenketak

- Amaiera:

  +A1 Aurkezpena prestatu

+A2 Aurkezpena egin

Atazen zerrenda aztertu ostean, 2.9. irudian ikus dezakegu LDE diagrama antolatu da. Diagraman agertzen diren ataza guztiak burutuko dira proiektuan zehar.



2.9. Irudia: LDE Diagrama

**LDE diagraman agertzen diren etiketak atazen zerrendan deskribatutako etiketak dira.

2.1.3. Arriskuak

Proiektu bat burutzeko garaian, hainbat faktorek izan dezakete eragina honen inguruan jarritako entrega-epeak betetzeko garaian, egindako planifikazioa ez zapuzteko. Horietako batzuk, aldez aurretik identifika daitezke eta hauen aurrean eman beharreko erantzuna planifika daiteke. Jarraian, aipaturiko arriskuak azalduko dira eta baita hauen aurrean eman beharreko erantzuna. Hala ere, kontuan izan behar da proiektua burutzerakoan dagoen ez-jakintza eta honek sor ditzakeen aurreikusi gabeko arazoak eman daitezkeela.

Arriskuen Zerrenda:

➔A1 Teknologiarekin lotutakoak

→A11 Datuen galera orokorra

Deskribapena *Software*ak, *hardware*ak edo pertsona batek eraginda proiektuaren datuak gal daitezke. Adibidez, bateria amaitzean ordenagailua amatatu egiten da eta berrabiaraztean gal dezakezu lana, ordenagailuaren plaka zapuztu daiteke... Arrisku maila honek larritasun maila oso handia dauka, egindako lana berriz egitera behartzen digulako eta planifikazioan atzerapen handiak sor ditzakeelako.

Aurrezaintza plana

I) Kanpoko disko gogor batean segurtasun kopiak egitea.
II) *Hardware*-a ondo zaintzen saiatu. Esaterako, tenperatura egokian mantendu, dagokien estalkiekin babestu, konexio puntuak garbi mantendu...
III) *Unix*-en oinarritutako sistema eragileak erabili, birusak ia ez daude eta arriskua txikiagotzen du.

Kontingentzia plana Disko gogorrean egindako *backup*-etatik informazioa errekuperatu.

➔ A2 Pertsonekin lorturikoak

→A21 Formazio falta dela eta blokeatzea

Deskribapena Formazio falta dela eta programatzerakoan blokeatzea, planifikazioan atzerapenak suertatuz eta lan erritmoa motelduz. ROS era bat ezezaguna denez proiektuaren egilearentzako, eta gainera dagoen dokumentazio guztia ingelesez dagoenez, arrisku handia sor daiteke, erritmoa motelduz.

Aurrezaintza plana

I) ROS *wiki*-an erabiltzaile kontu bat egin.
II) Ingelesa ikasi.

Kontingentzia plana Foroetan galdetu zalantzak edo tutorearengana jo zalantzak argitzeko.

→A22 Osasun arazoak

Deskribapena Epe luzeko proiektua denez, gaixotasun edo osasun arazo bat edukitzea posible da, bai egileak izatea edo egilearen familiakoren bat. Honek atzerapenak sor ditzake proiektuan. Larritasun maila baxua du, denbora luzez osasun arazo bat izateko probabilitatea oso baxua baita.

Aurrezaintza plana Ez dago, proiektuaren kanpo geratzen da.

Kontingentzia plana Sendatutakoan proiektuari lehentasun handiagoa eman beste zereginekiko.

➔ A3 Planifikazioari lotutakoak

→A31 Lan karga handiegia dela eta, epeak ez betetzea

Deskribapena Beste zereginak eta betebeharrak direla eta, epe batetara garaiz ez iristea. Larritasun maila nahikoa da.

Aurrezaintza plana Planifikazio zuzen bat egiten saiatu eta astero proiektuan sar daiteken ordu kopurua estimatu eta betetzen ahalegindu.

Kontingentzia plana Betebehar eta zereginak amaitutakoan proiektuari lehentasun handiagoa atzitu.

→A32 Proiektuaren planifikazio txarra

Deskribapena Egilearen esperientzia faltarengatik egindako planifikazioa okerra izan daiteke. Eragin aldakorreko arrisku bat da, zaila baita kalkulatzea zenbaterainoko atzerapenak suertatu daitekeen arazo honegatik.

Aurrezaintza plana Planifikazioan denbora bat gorde birplanifikaziorako.

Kontingentzia plana Atazaren edo proiektuaren birplanifikazio bat egin. Bestalde, arazoak minimizatzen saiatu.

2.1.4. Lan metodologia

Kudeaketa eta antolaketa

Egileak bere kabuz burutuko du proiektua, hau da, ez du enpresa batean burutuko, ondorioz berak erabaki beharko du zenbat ordu eskainiko dizkio proiektuari, lan erritmoa mantentzeko eta finkatzen duen epean amaitzeko. Egileak tutore bat izango du, Elena Lazkano, eta berarekin hartuko dira erabaki garrantzitsu guztiak.

Egilearen eta irakaslearen arteko komunikazioa e-posta bitartez egitea dago edo egileak irakaslearen denbora librean bilerak egingo dira.

Artxiboaren kudeaketa

Astero artxiboaren kopia bat egingo da. Bai disko gogor batean eta bai USB batean. Artxiboa bi zatitan banatua egongo da, dokumentazioa alde batetik eta proiektuaren kodea beste aldetik. Proiektuaren lan ingurunea (workspace) gordeko da proiektuaren kode giza eta gordetzen den fitxategiaren izenean lan ingurunearen izena, bertsio zenbakia eta data agertu behar dira (rosbuild_ws(1, 03_12_2014)). Beste alde batetik memoriarekin erlazionatuta dauden dokumentu guztiak agertuko dira fitxategi ezberdin batean. Eta gordetzeko prozesua berdina izango da, fitxategiaren izena, bertsio zenbakia eta azkenik fitxategia noiz kopiatu den data (memoria_dokumentazioa(3, 04_03_2015)).

2.1.5. Planifikazioa

Tamaina honetako proiektu batekin hasi aurretik oso garrantzitsua da planifikazio on bat egitea eta gero hau kontrolatzea. Alde batetik ataza bakoitzaren iraupena estimatu behar da eta, beste aldetik, proiektuarentzako garapen prozesu zikloa hautatuko da. Ideia nagusia iterazio bakoitzak proiektuaren hedapena suposatuko duela da, hau da, elementuren bat zehatzago adieraztea edo funtzionaltasun bat gehitzea.

Denbora estimazioak

2.1. Taulan denbora estimazioak agertzen dira. Ataza bakoitzari denbora estimazioa bat atzitu zaioa. Denbora, zenbat ordu erabiliko diren ataza horretarako adierazten du.

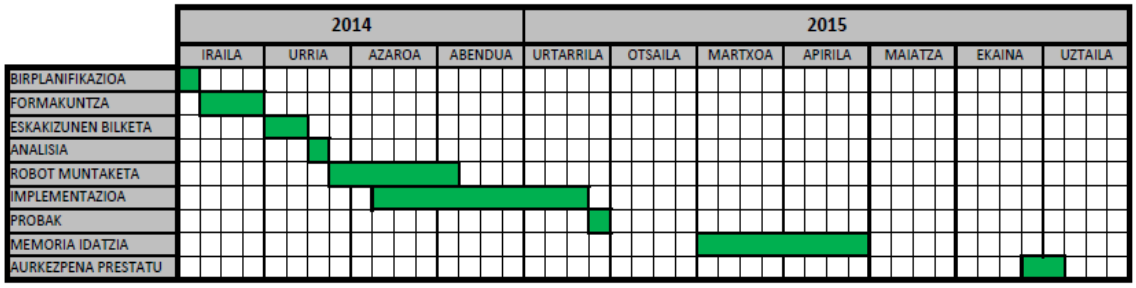
ATAZA	**ESTIMAZIOA**
Formakuntza	
ROS	50
Ros_nxt	25
Linux	10
Kudeaketa Prozesua	
Kudeaketa	
Bilerak	15
Artxibo kudeaketa	5
Planifikazioa	
PHD-a egin	10
Birplanifikatu	5
Garapen Prozesua	
Eskakizunen bilketa	10
Analisia	10
Robot muntaketa	25
Inplementazioa Burutu	100
Probak egin	20

Dokumentazioa	
Memoria idatzi	50
Memoria zuzendu	10
Amaiera	
Aurkezpena prestatu	7
GUZTIRA	**352**

2.1. Taula: Denbora Estimazioak

Gantt Diagrama

2.10. irudian agertzen den diagrama azken birplanifikazioan eginda dago. Proiektua 2012an hasi zen, baina, 2015eko irailean proiektuaren helburua murriztu eta birdiseinatzea erabaki zen.



2.10. Irudia: Gantt Diagrama

2.1.6. Aurrekontua

Proiektu hau zenbat kostatzen den jakiteko 2.2. taulan agertzen den aurrekontua kalkulatu da. Kontuan hartu da, erabilitako hardwarea, programatzaileak sartu dituen orduak. *Hardware*aren kostua kalkulatzeko www.ro-botika.com web orriko prezioak kontuan hartu dira. Eta lanean ibili diren pertsonen soldatak kalkulatzeko ordu kopuruaren arabera kontuan hartu da. Junior programatzailea: 10€/orduko.

HARDWAREA:

-Lego NXT Mindstorm ……..…… 345.00€

-Lego pieza gehigarriak ……..... 99.95€

-Iparrorratza ……..…….. 75.00€

LANGILEEN SOLDATAK:

-Junior programatzailea (350 Ordu) ……..……. 3500€

PROIEKTUAREN KOSTUA: _____4019.95€

2.2. Taula: Proiektuaren Aurrekontua

2.2. Estimazioak

2.2.1. Estimatutako denbora eta denbora erreala

Ez da erreza egileak duen planifikatze esperientziarekin, planifikazio egokia egitea. Honek hainbat arazo ekar ditzake proiektua aurrera eramateko garaian, aurretik planifikaturikoa zehatz-mehatz betetzea zaila egiten delako. Hau jakinda, planifikazioan birplanifikatzeko tartea utzi da.

Atal honetan hasieran proiektu honentzako egindako planifikazioaren eta benetan sartu diren ordu kopuruen arteko konparaketa egingo da, eman diren desberdintasunen balorazioa emanez. 2.3. taulan ikus dezakegu eman den diferentzia.

ATAZA	ESTIMAZIOA	ERREALA
Formakuntza		
ROS	50	65
Ros_nxt	25	30
Linux	10	10
Kudeaketa Prozesua		
Kudeaketa		
Bilerak	15	15
Artxibo kudeaketa	5	6
Planifikazioa		
PHD-a egin	10	13
Birplanifikatu	5	10
Garapen Prozesua		
Eskakizunen bilketa	10	15
Analisia	10	10
Robot muntaketa	25	30
Inplementazioa Burutu	100	100
Probak egin	20	25
Dokumentazioa		
Memoria idatzi	50	50
Memoria zuzendu	10	10
Amaiera		
Aurkezpena prestatu	7	7
GUZTIRA	**352**	**391**

2.3. Taula: Estimatutako denbora eta denbora errealaren konparaketa

2.2.2. Balorazioa

Planifikazioa ez da ona izan eta horregatik proiektua hasi zenetik denbora asko pasa da. Proiektua 2012ean hasi zen eta izan ditudan egoerengatik ez naiz amaitzeko gogoekin jarri 2014 irail arte. Ordenagailua puskatu zitzaidan, dokumentazioa ulertzeko nuen arazoengatik gehiago atzeratu zen. Konturatu nintzen ingelesa ikastea behar nuela dokumentazioa hobeto ulertzeko eta Irlandara joan nintzen ingelesa ikastera. Han amaitzea nuen helburu baina, nahiz eta ordu asko pasa ordenagailuaren aurrean, NXTCam *driver*a saiatzen ulertzen, irailean proiektua birmoldatu genuen eta planifikazioa hobeto bete zen.

Laburbilduz, gehien atzeratu didana ROS nola zebilen ulertzea izan da. Ordu asko pasa ditut ROS-en dokumentazioa irakurtzen eta tutorialak jarraitzen. Gainera hasieran kriston arazoak izan nituen Ubuntu instalatzeko garaian, ez nintzen oso trebatua linux-ekin.

2. zatia: Garapena

3. kapitulua:

Erabilitako tresnak

3. kapitulua: Erabilitako tresnak

3.1. Hardwarea

Proiektuaren ibilgailua eraikitzeko Legok NXT Mindstorm kit-a erabili da. Robot hau muntatzeko kit-ak, adreilu bat, 3 motor eta 4 sentsore ekartzen ditu. Adreilua kontroladoreari esaten zaio eta sentsoreak irakurri, motorrei aginduak bidali eta programak exekutatzeko da beharrezkoa.

- NXT Adreilua:

Atmel-eko bi mikrokontroladore ditu, lehengoak 32-biteko Atmel AT91SAM7S256 prozesadore du kontroladore nagusi bezala, 256KB-eko flash memoria eta 64KB-eko RAMa. Eta bigarrenak, 8-biteko Atmel ATmega48 prozesadorea, 4KB-eko flash memoria eta 512 Byteko RAMa.

3.11. Irudia: Lego NXT Adreilua

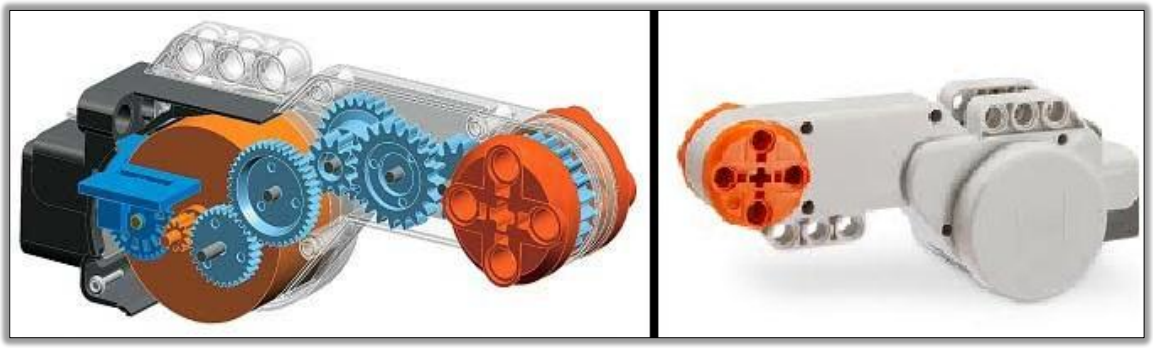
3.11. irudian ikus dezakegu LEGO NXT adreilua. Sentsoreekin komunikatzeko sarrerako 4 portu ditu eta, motorrentzat berriz, irteerako 3 portu, guztiak 6 pinekoak. Sarrerako portuak 1-4 zenbakiak dauzkatenak dira eta hemen sentsoreak konektatzen dira. Bestalde, irteerako portuak, A-C letra daukatenak dira eta motorrak konektatzeko

balio dute. Horrez gain, USB portu bat eta bluetooth-a aukerak ere baditu, ordenagailuari konektatzeko. Honek, ibilgailuaren eta aplikazioaren arteko konexioa lagatzen du.

- Motorrak:

NXT motorrek 9V potentzia eskatzen dute. Potentzia hau dutenean 117 rpm errotazio abiadura, 16.7 N.zm-ko indar parea eta 550 mA-ko korronte intentsitatea lor dezakete. Gainera, 1:48ko erredukzioa dute, 3.12. irudian zabalduta dagoen motorrean ikus dezakegu. Behar izanez gero, erredukzioa handitu daiteke, LEGOk eskaintzen dituen engranajeekin.

Motorrak erabil ahal izateko, kable bidez, adreiluak dituen 6 pineko output-eko portuetara konektatu behar dira.



3.12. Irudia: Motorrak

- Sensoreak:

LEGO NXT *Minstorm*eko oinarrizko kit-ak, 4 sentsore eskaintzen dizkigu, argi-sentsorea, ultrasoinu-sentsorea, mikrofonoa eta talka-sentsorea. Baina beste sentsore batzuk ere badaude, adibidez, iparrorratza edo kolore-sentsorea.

  - Argi-sentsorea: 2 funtzionamendu modu ditu: Aktiboa eta inaktiboa. Inaktiboan, argi-intentsitatearen araberako balioa itzultzen du. Balio altuak jasotzen duenean, argitasun handia dagoela adierazten du, eta bestalde, balio baxuekin, argitasun txikia dagoela adierazten du. Aktiboan berriz, argi-infragorria igorri eta jasotako islaren araberako balioa ematen digu.

Horregatik, zuria eta beltza desberdintzeko da egokia. 3.13. irudian ikus daiteke argi-sentsorea eta bi led ditu, bata igorlea da eta bestea hartzailea.

3.13. Irudia: Argi-sentsorea

- Ultrasoinu-sentsorea: Ultrasoinu-sentsoreak helgaldi denboraren menpeko sentsoreak dira (TOF, Time of flight). Seinale baten hedapen-abiadura ezagututa, seinalea igorri eta oihartzuna jaso arteko denbora neurtuz, oihartzuna eragin duen oztoporainoko distantzia kalkula daiteke. Ultrasoinuen kasuan, igorritako seinalea soinua da. 3.14. irudian agertzen den sentsorea, ultrasoinu-sentsorea da.

3.14. Irudia: Ultrasoinu-sentsorea

- Talka-sentsorea: Sentsore inaktibo bitarra da hau, 0 eta 1ak irakurtzen ditu. 0ak sentsorearen botoia sakatuta ez dagoela adierazten du, eta 1ak, berriz, kontrakoa, botoia sakatuta dagoela. Sentsorearen botoia 3.15. irudian ikusten den laranja zatia da.

3.15. Irudia: Talka-sentsorea

- Mikrofonoa: Soinuaren intentsitatea neurtzeaz arduratzen da, db-etan neurtzen du soinua. Honek ere 2 funtzionamendu modu dauzka, soinu gordina jaso edo filtratuta jaso. 3.16. irudian ikus daiteke NXT mikrofono bat.

3.16. Irudia: Mikrofono-sentsorea

- Kolore-sentsorea: 6 kolore diferente detektatzen ditu. Urdina, berdea, gorria, horia, zuria eta beltza. Horretarako rgb (gorria, berdea eta urdina) argiak igortzen ditu. 3.17. irudian ikus dezakegu kolore-sentsore bat.

3.17. Irudia: Kolore sentsorea

- Iparrorratza: Iparrorratz sentsorea, lurraren ipar magnetikoarekiko desbideratzea neurtzen duen gailua da. Robotaren norabideak detektatzeko balio du. Sentsoreak beste gailu magnetikoren interferentziak saihesteko kalibrazioa behar du.

3.18. Irudia: Iparrorratza

3.2. Softwarea

Proiektuan erabili den *software* guztia, GNU lizentziadun *software*a izan da. Ubuntu sistema eragilea erabili da, hasieran 10.04 bertsioan eta geroago, dependentzia arazoengatik, 12.04 bertsiora eguneratu zen.

Erabili den sistema eragileaz aparte, robota komunikatzeko ROS banaketa eta horren barruan, robotak behar duen, nxt_ros *driver*-a erabili dira. Hurrengo azpiataletan, zer eta zertarako erabiltzen den ROS azaltzen da eta nxt_ros *driver*-aren deskribapen bat agertzen da.

3.2.1. ROS

ROS (*Robot Operating System, Robot Open Source*) robotekin lan egiteko liburutegi, eta tresnak eskaintzen dituen softwarea da. ROSek aukera ematen du robotentzako *softwarea* garatzeko. ROSek eskaintzen dizkigun zerbitzuak, sistema eragile batek eskaini ditzakeenak dira, *hardware* abstrakzioa, behe mailako gailuen kontrola, prozesuen arteko mezu komunikazioa edo paketeen mantenua. Grafo arkitektura batean oinarrituta dagoen sistema da, prozesaketa nodoek egiten dute, eta mezuak bidali edo jaso ditzakete. Mezuak sentsoreenak, kontrolekoak, egoera-aldagaiak edo planifikatzailearenak izan daiteke.

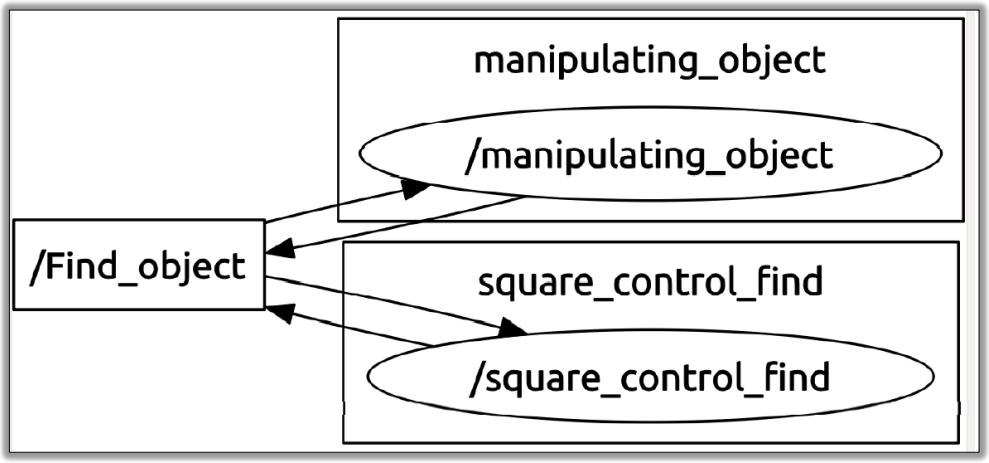
ROSek bi API nagusi ditu, bata C++-en programatzeko (*roscpp*) eta, bestea, Python-en programatzeko (*rospy*). Bi API hauek nodoen inplementazioa eta prozesamendua egiteko tresnak dira. Bestetik, komunikazioak ez dauka lengoaiarekiko menpekotasunik. Mezuak ez dira lengoaia zehatz batean programatzen, formatu bereziko testu fitxategietan datu egitura bat definitzen da eta konpiladorea gai da fitxategi hori konpilatzeko eta lengoaia horretara egokitzeko. ROSek egiten duen prozesu honegatik, beste nodoekiko komunikazioa errazten du.

Esan bezala, ROSek grafo bidezko nodo egitura antolatzen du. ROSeko aplikazio bat exekutatu nahi duzunean, nodoen arteko komunikazioa kudeatzen duen nodoa martxan jarri behar da. Nodo hau, terminalean `roscore` komandoa exekutatzen martxan jartzen da. `roscore` nodoak, ROSeko MASTERa, *parameter server*-a eta rosout *logging* nodoa hasieratzen ditu.

ROS sistema banatua da izaeraz. Hau da, nodoak (bezero-prozesuak) makina desberdinetan exekuta daitezke baina maisua non exekutatzen den jakin behar dute. Horretarako, ROS_MASTER_URI egoera-aldagaia finkatu behar da makina guztietan. Nodo guztiak makina bakarrean exekutatuko badira, aldagai horren balioa http://localhost:11311 izaten da. *Localhost* makinaren helbidea da eta 11311 zenbakia, komunikazioa ematen den portua da. Beste makinekin komunikatu beharra izanez gero, *localhost* makinaren IPaz ordezkatu behar da.

3.19. irudian, ROSeko aplikazio baten egoera-diagrama ikus daiteke. `/manipulating_object` eta `/square_control_find` nodoak dira. Eta hauek, `/Find_object` *topic*-ean publikatzen eta suskribatzen dira. `/Find_object` objektua

gutunontzia da, mezuak gordetzen diren lekua, honi *topic* esaten zaio. Irudiak grafo egitura ROSek eskaintzen duen `rqt_graph` aginduaren bidez automatikoki lortzen da.



3.19. Irudia: rqt_graph diagrama

Edozein garatzailek inplementatutako nodo bat exekutatzeko `rosrun` komandoa erabiltzen da. Programazio lengoaiaren arabera, kodea garatzen da eta konpilatzen da. Hurrengo pausua, *manifest.xml* eraldatzea izango zen, baina bakarrik C++ APIa erabili bada. Eta, azkenik, komandoa exekutatzen da. Exekutatzeko, lehenik eta behin `roscore` komandoa exekutatu behar da, eta gero, `rosrun`. Komando honi, exekutatu nahi den nodoaren paketea eta nodoaren izena adierazi behar zaizkio. Adibidez: `rosrun pakete_izena nodo_izena` .

Gainera, ROSek, terminalean exekutatzeko komando propio gehiago ditu. Adibidez, `roscd` paketeen zuhaitz egituran nabigatzeko, `rostopic` bidaltzen ari diren mezuak aurreikusteko, `rosparam` bidaltzen diren parametroak aurreikusteko eta zenbait gehiago. Tutorialak eta informazio gehiago www.ros.org/wiki web orrian aurkitu daiteke.

3.2.2. NXT paketea

ROS eta NXTren arteko konexioa, NXT *driver*-aren bidez egiten da. *Driver* hau ROS-entzako dagoen paketea da, eta NXT duen hardware guztiarentzako inplementazioa aurkitzen da.

Driver hau, ROS Groovy bertsiorako dago eskuragarri. ROS bertsio ezberdinak daude, eta aplikazio bat, ROS bertsio konkretu batentzako inplementatzen denean, ezin da zuzenean beste bertsio batera pasatu. ROS bertsio horretarako konpilatu behar da eta ROS horrek izan dituen aldaketen arabera egiten da.

Kontuan izan behar da ze Ubuntu bertsio instalatzen den, ROS bertsio bakoitza Ubuntu bertsio ezberdinentzako baitago eskuragarri. Kasu honetan, ROS Groovy, Ubuntu 12.04 bertsiorako konpilatua dago.

NXT_ROS *driver*-a ROSeko *stack* bat da, bere barnean hainbat pakete ezberdin aurkitu daitezke. Alde batetik, NXT robota RVIZ simulatzailean aritzeko *pluging*-a eta bistaratzeko eredu grafikoak daude. Eta, bestetik, NXT martxan jartzeko behar diren *driver* espezifikazioak aurkitzen dira. Ikusi 3.4. taula, nxt paketeak dituen paketeak agertzen dira.

Nxt paketea:
- nxt_controllers
- nxt_description
- nxt_msgs
- nxt_lxt2urdf
- nxt_python
- nxt_ros
- nxt_rviz_plugin

3. 4. Taula: nxt paketea

Proiektu honen helburua, robotak mundu errealean portaera konkretu bat erakustea denez, NXT martxan jartzeko behar den atala aztertu da. Hau inplementatzen duen paketeak *nxt_python* du izena. Pakete honetan, adreilua, motorra, sentsoreak eta usb-rekiko konexioaren espezifikazioak aurkitzen dira. *nxt_python* paketea, *python* programazio lengoaian dago garatua, hau dela eta, proiektuan garatuko den kode guztia *python*-en idatziko da.

Hurrengo ataletan, *nxt_python-en* agertzen diren fitxategi nagusienak azalduko dira. Fitxategi bakoitza objektu bat izango da eta barruan klaseak eta metodoak egongo dira. Objektu bakoitzaren deskribapenean, bi sinbolo bereiz ditzakegu: + sinboloak, klasea dela adierazten du, eta, – sinboloak, aldiz, metodoa dela adierazten du:

- `usbsock.py`: Klase honek, ROS eta NXTren arteko konexioa eskaintzen du. Konexioa, USB bitartez izango da. Bluetooth-eko aukera era badago (beste objektu batean inplementatuta), baina arazoak ematen ditu eta motela da. 3.4. taulan, `usbsock.py` objektuaren deskribapena agertzen da.

Usbsock.py
+USBSock(object): -_init_(self, device) -connect(self) -close(self) -send(self, data) -recv(self) -find_bricks(host, name)

3.5. Taula: usbsock klasea

- `locator.py`: Honen bidez, NXT adreilua konektatuta dagoen begiratzen da, gero `usbsock.py`-ren bitartez NXTren konexioa egiteko. 3.5. taulan ikus daitezke objektu horren klaseak.

locator.py
+BrickNotFound(Exception) +NoBackendError(Exception) -find_bricks(host, name) -find_one_brick(host, name)

3.6. Taula: locator klasea

- `brick.py`: adreiluaren erazagupena egiten da klase honetan. Adreiluaren metadatoak lortzen dira eta honen hasieratzea egiten da. 3.6. taulan `brick.py` objektuak dituen klaseak ikus ditzakegu.

brick.py
+_Meta(Type) +Brick(Object) +FileFinder(Object) +FileReader(Object) +FileWriter(Object) +ModuleFinder(Object)

3.7. Taula: brick klasea

- `motor.py`: Motorren definizioak egiten dira eta hauek exekuta ditzaketen funtzioak adierazten dira. Ikus 3.7. taulan agertzen den definizioa.

motor.py
+Motor(Object) -_init_(self, brick, port) -set_output_state(self) -get_output_state(self) -reset_position(self, relative) -run(self, power, regulated) -stop(self, braking) -update(self, power, tacho_limit, braking, max_retries)

3.8. Taula: motor klasea

- `sensor.py`: Hemen, sentsoreak definitzen dira eta bi taldetan banatuta daude. Alde batetik, sentsore analogikoak eta, bestetik, sentsore digitalak. Ikus 3.8. taulako definizioa.

sensor.py
+Type(object) +Mode(object) +digitalSensor(Sensor) +CommandState(object) +AnalogSensor(Sensor) +TouchSensor(AnalogSensor) +LigthSensor(AnalogSensor) +ColorSensor(AnalogSensor) +SoundSensor(AnalogSensor) +UltrasonicSensor(DigitalSensor) +AccelerometerSensor(DigitalSensor) +GyroSensor(AnalogSensor)

3.9. Taula: sensor klasea

- `compass.py`: Klase honek, `sensor.py` objektua hartzen du erreferentzia gisa, iparrorratz sentsorea definitzeko. Ikus 3.9. taula, objektu honen definizioa agertzen da eta `sensor.py` objektuari erreferentziatzen zaio.

compass.py
+Command(object) +MetaCMPS_Nx(sensor ._Meta) +CompassSensor(sensor .DigitalSensor)

3.10. Taula: compass klasea

3.2.3. Adibidea

Driver honen analisia amaitzeko, eta ROS eta NXTren arteko konexioa frogatzeko, sentsoreak erabiltzen eta *topic* bat bidaltzen duen pakete sinple bat garatu da. Bi nodo sortu dira hau frogatzeko. Lehenengo nodoak robota aurrerantz gidatuko du oztopoa aurrean topatu ezean. Bigarren nodoak, oztopo hori lata bat delakoan, objektua jasoko du. Lehenengo nodoak 2.ari esango dio robotak prest dagoela objektua jasotzeko.Bi nodoak komunikatzeko, mezu bat definitu da eta *topic* baten bitartez mezu hori bidaliko da. 3.1. kode zatian, mezuaren definizioa dago.

```
Header header

Boolean find_it
```

3.1. Kodea: Mezu erazagupena

Bi nodoak komunikatzeko *publisher* eta *subscriber*-ak definitu behar dira. 3.2. kodean lehengo nodoaren *publisher*-a definitzen da.

```
Self.pub=rospy.Publisher('Find_object', Find_object)
```

3.2. Kodea: publisher-aren erazagupena

Bidaliko den *topic*-ean izena eta mezu mota adierazten da. Hau, klasearen hasieraketan jartzen da eta mezua bidali nahi den unean funtzio horri deitzen zaio, mezua gehituz.

Subscriber-a, *publish*-ean egiten den moduan, klasearen hasieraketan definitzen da. Zein *topic*-ari harpidetuko den eta honek zein funtzioari pasatuko dion mezua adierazi behar da. Funtzio honi *callback* funtzioa deitzen zaio. Mezua funtzio horretara heltzen da eta hor mezua prozesatu behar da. 3.3. kode zatian, froga hau gauzatzeko, erabili den harpidetzaren definizioa agertzen da.

```
Rospy.Subscriber('Find_object', Find_object, self.findit)
```

3.3. Kodea : Subscriber-aren erazagupena

Behin ROS-era konektatua dagoela, NXT-ko adreilua bilatu eta hasieratzen da. Adreilua bilatzen da, eta, honen arabera, suertatu den konexioa, nodoari pasatzen zaio, sentsoreak eta motorrak hasieratu dezakeen. 3.4. kode zatian adierazten da, nola egiten den.

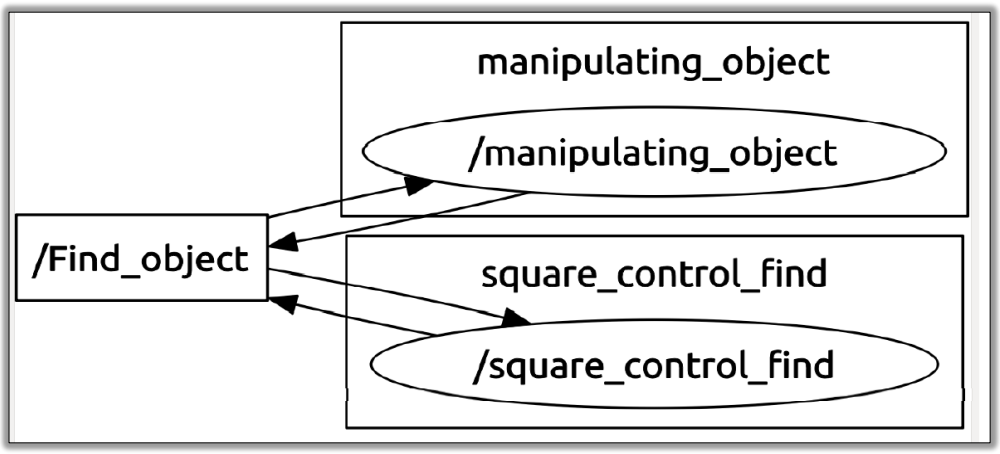
```
Sock = nxt.locator.find_one_brick()
if sock:
     b=sock.connect()
     m1= Motor(b, PORT_B)
else
     print "No NXT brick found"
```

3.4. Kodea : Adreiluaren hasieraketa

Froga martxan jartzeko hurrengo 3.5. kode zatiko komandoak exekutatu dira terminalean:

1. Terminalean:
   ```
   $ roscore
   ```
2. Terminalean:
   ```
   $ rosrun nxt_kap find_lata
   ```
3. Terminalean:
   ```
   $ rosrun nxt_kap take_lata
   ```

3.5. Kodea: Frogaren nodoak exekutatzeko komandoak



3.20. Irudia: Adibidearen rqt_graph

3.20. irudian `rqt_graph` komandoaren bitartez atera den diagrama azaltzen da. Diagrama honetan, froga martxan zegoenean atera da. Ikus dezakegu, nola bi nodoak `/Find_object` *topic-a* publikatzen eta susbribatzen duten.

Egin den frogarekin, konprobatu da ROS eta NXT-ren arteko konektagarritasuna posible dela. Orduan, hurrengo kapituluetan planteatuko den portaera inplementatzeko prestatua dugula sistema.

4. kapitulua:

Zabor biltzailea

4. Kapitulua: Zabor biltzailea

ROS eta NXT-ren arteko konektagarritasuna aztertu ondoren, portaera bat inplementatzea izango da hurrengo helburua. Lehenik eta behin, NXT-ek eskaintzen digun funtzionalitateekin, portaera bat diseinatu behar da. Gero, robotaren morfologia analizatu da, robotak bete behar dituen eskakizun guztiak bete ditzan. Eta azkenik, portaera hori inplementatu da.

Portaera inplementatua dagoenean, esperimentu edo froga ezberdinak egingo dira, robotaren erantzukizuna ebaluatzeko.

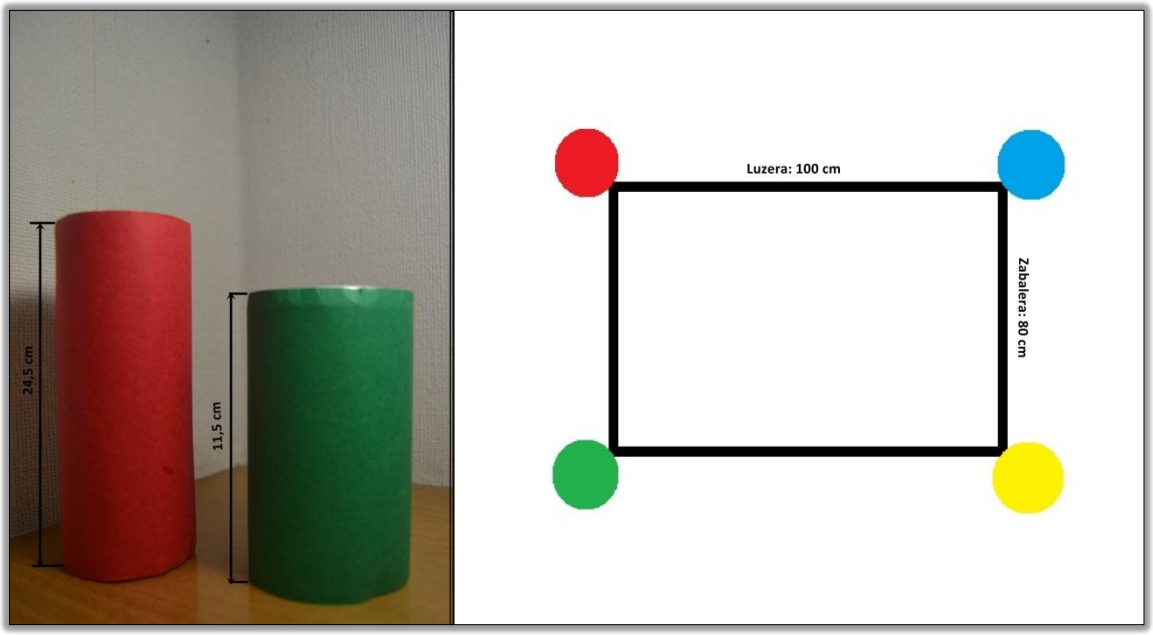
4.1. Portaeraren deskribapena. Ingurunea.

Robota ibiliko den ingurunea zaborrez beteta egongo da eta robotak, zabor hori sailkatu behar izango du. Zaborra latak izango dira. 4.21. irudian agertzen diren modukoak, 24,5 edo 11,5 cm-koak izan daitezke. Ingurunean izango den zabor mota bakarra, zabor-latak izango dira.

Lata bakoitzak kolore bat izango du eta kolore horren arabera sailkatu beharko du robotak. Koloreak gorria, berde, urdina eta horia izango dira, kolore-sentsoreak detektatzen dituen koloreak direlako.

Robotaren lan ingurunea mugatzeko karratu bat markatuko da zoruan, 100 cm-ko luzerarekin eta 80 cm-ko zabalerarekin. Karratu hori zinta beltz batekin egingo da. Zinta beltz horrek, argi izpiak ez errebotatzea eragiten du.

4.21. irudian agertzen den bezala, eszenatokiak lau txoko izango ditu, kolore bakoitzeko bat. Karratuaren izkina bakoitza kolore bat izango du, beti kolore berarekin. Robotak lata bat hartzen duenean, kolorea detektatu behar izango du eta gai izan behar da, izkina horretara eramateko.



4.21. Irudia : Zaborra eta porteraren eszenatokia

4.2. Robotaren eraikuntza

Robota eraikitzeko garaian, zenbait faktore kontuan hartu behar dira. Oso garrantzitsua da morfologia egokia izatea, etorkizunean portaera onargarria izan dadin.

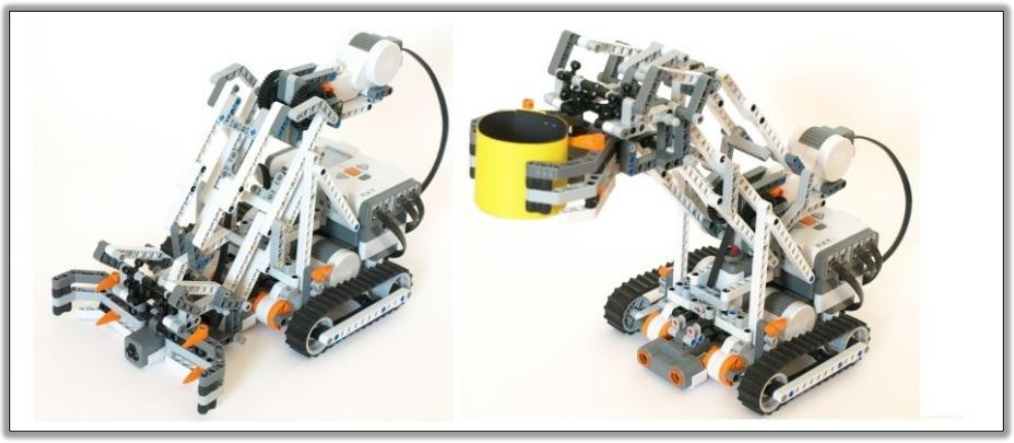
Deskribatutako portaera inplementatzeko, beso bat beharko du. Beraz, robotak bi atal nabari izango ditu, robotaren oinarria batetik, eta bestetik, robotaren besoa. Besoaren kokapenak oinarrien oreka ziurtatu beharko du, robota irauli ez dadin, pisua dela eta, mugimendua eragotzi ez dezan. Gainera besoaren atzipen-eremua ziurra dela ziurtatu behar da, besoak robotarekin talka egin ez dezan.

Robotaren mugimendu-sistema diferentziala izango da. Gurpilen konfigurazioari dagokionez, aukera desberdinak frogatu dira. Hainbat proba egin ostean, 3. gurpil eroaren aukera arbuiatu eta beldar-kateak erabiltzea erabaki da.

4.2.1. Oinarriaren eraikuntza

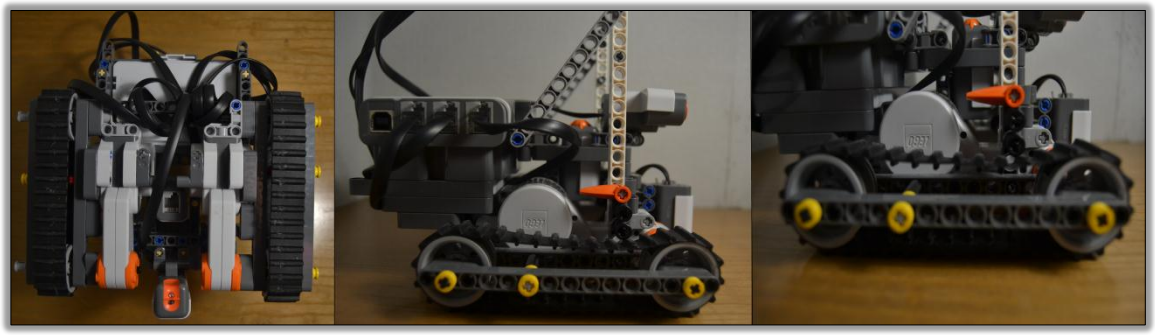
Oinarria muntatzeko garaian, ingurunea hautemateko sentsoreak ezarriko dira eta, besoan, latak identifikatzeko sentsoreak. Hau buruan izanda Google-n sartu eta prototipo ezberdinak aztertu dira.

Snatcher estiloko robot eredua aukeratu da [3]. *Snatcher* estiloko robotak (ikusi *4.22. irudia*) zabor biltzailearentzako beharrezkoak diren ezaugarri guztiak betetzen ditu.



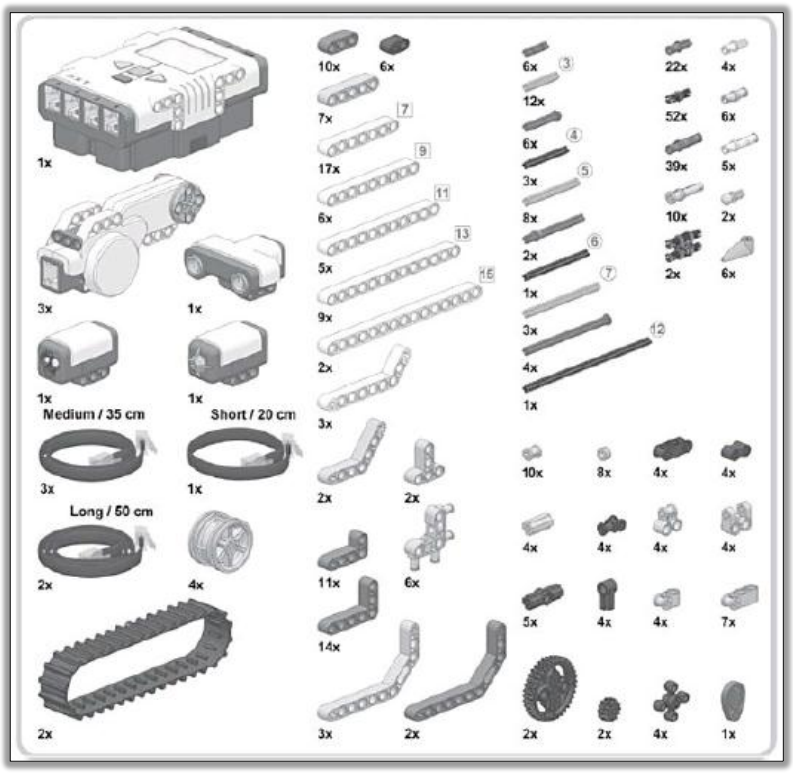
4.22. Irudia : Snatcher robota

Oinarria eta besoaren oreka nabaria da, adreilua oinarriaren atzealdean kokatzen da eta besoaren pisu handiena aurrealdean dago. Horrela, gurpilen marruskadura orekatuta egongo da. Gainera, ez dauka gurpil erorik, beldar-kate gurpilak erabiltzen ditu. Aurreko gurpilak beldar-katearen indar eragileak dira.



4.23. Irudia : Gurpilen egitura

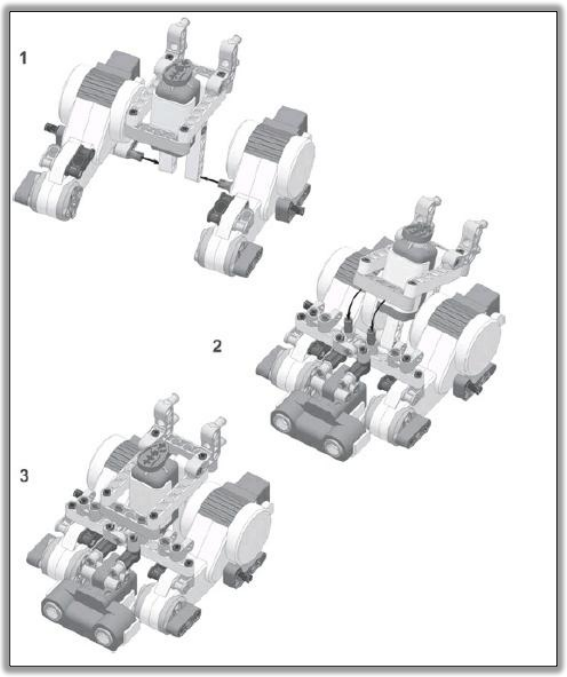
Eraikuntzan 356 LEGO pieza erabili dira, adreilu bat, 5 sentsore, 3 motor, 7 kable, 4 hagun eta 2 beldar-kate. 4.24. irudian ikus daiteke erabilitako pieza guztien irudia.



4.24. Irudia : Erabili diren piezak

Oinarriaren eraikuntzan urrats nagusienak hurrengoak izan dira:

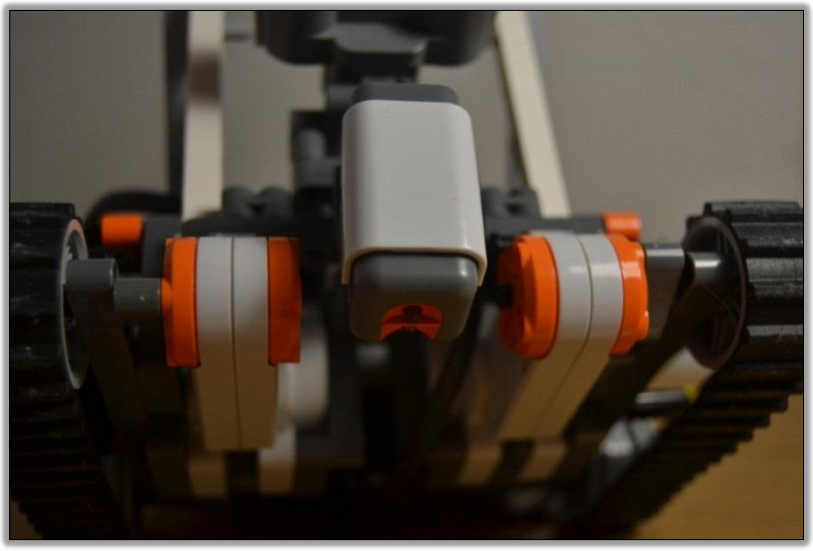
1→ Gurpilek, mugimendu egokia izateko, garrantzitsua da, gurpilen eragingailuak izango diren bi motorrak sendo kokatzea. 4.25. irudian ikus daiteke motorrak oinarriaren muina izango direla. Gurpilak mugitzeko ez da erredukzio gehiago gehitu, ez baita beharrezkoa suertatu.



4.25. Irudia: Oinarri muntaketa. 1. urratsa

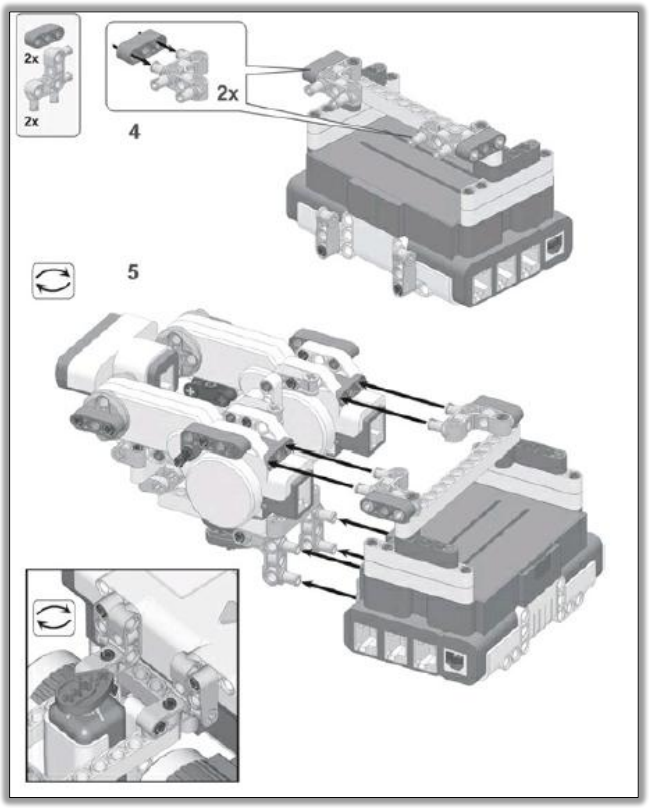
Robotak izango duen portaeran, argi-sentsorea, iparrorratza, ultrasoinu-sentsorea, kolore-sentsorea, hiru motorrak eta adreilua erabiliko dira.

Tutorialean ultrasoinu-sentsorea zoruaren parean ezartzen dute. Baina robotean beste leku batean kokatu da, gure portaerarako egokiago izango delakoan. Pistaren marra beltzaz ohartu behar da robota eta horregatik 4.26. irudian agertzen den bezala kokatu da.

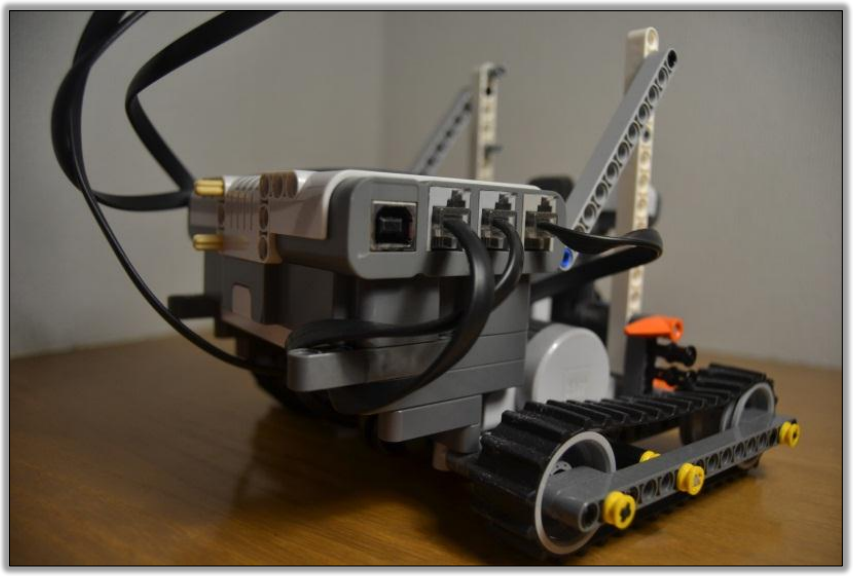


4.26. Irudia: Argi-sentsorearen kokalekua

2→ Lehen azaldu den moduan, adreilua atzealdean kokatzen da, besoa gehitzerakoan pisua orekatzeko. 4.27. irudian ikus daiteke nola muntatzen den, eta 4.28. irudian, bere kokalekua.

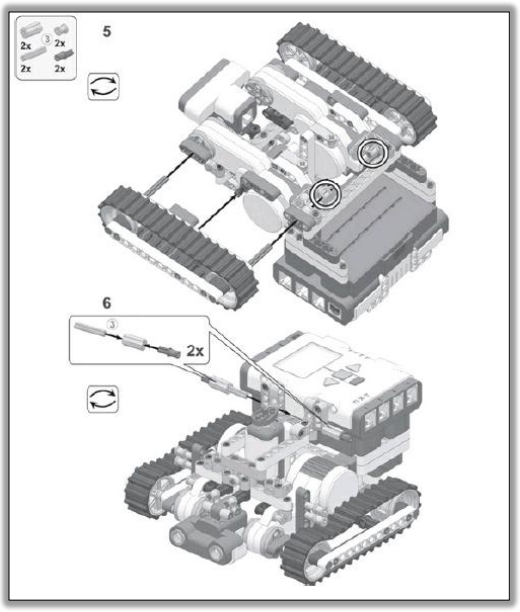


4.27. Irudia: Oinarri muntaketa. 2. urratsa

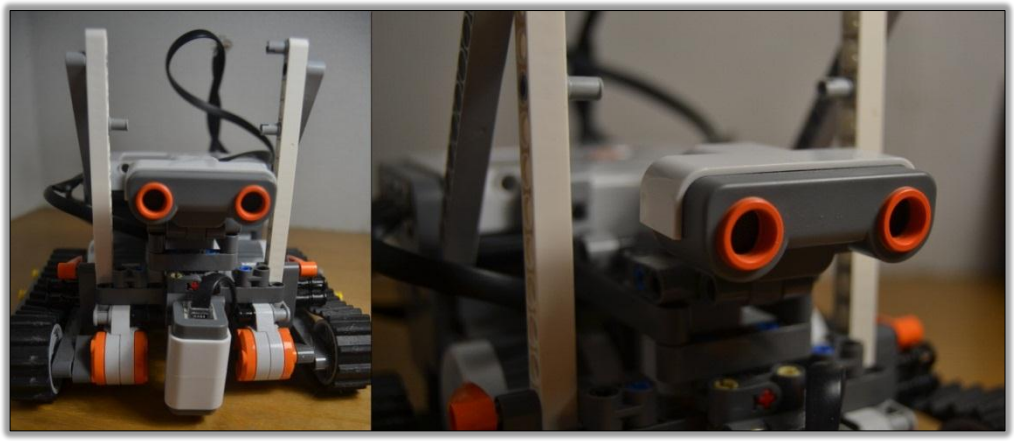


4.28. Irudia: Adreiluaren kokalekua

3→ Oinarriaren eraikuntza amaitzeko, beldar-kateak muntatzea falta da. 4.30. irudian agertzen den bezala, oinarrian ezartzen dira. Gure portaerarako argi-sentsoreaz gain, ultrasoinu-sentsorea ere behar dugu. Eta ikusita besoaren atzipen-eremutik kanpoko gune bat zegoela aurrealdean, ultrasoinu-sentsorea bertan jartzea erabaki da, 4.31. irudian agertzen denenez. Leku horretan kokatzeak, arazo txiki bat dakar, kontuan hartu beharrekoa programatzeko garaian. Objektuak detektatzean besoa beti goialdean egon behar da. Hau da, robotak bere bilaketa prozesuan besoa igota eraman behar du oklusiorik ez gertatzeko.



4.29. Irudia : Oinarri muntaketa. 3. urratsa

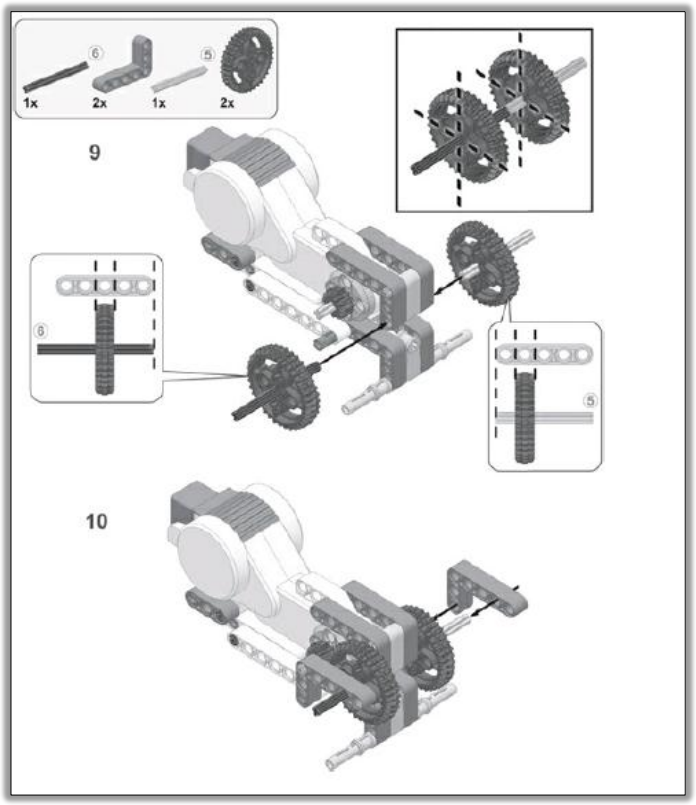


4.30. Irudia : Ultrasoinu-sentsorearen kokalekua

4.2.2. Besoaren eraikuntza

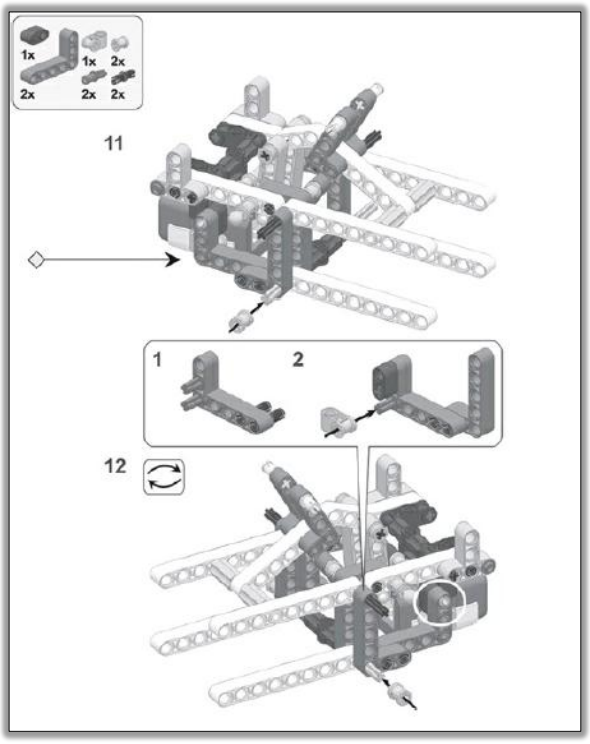
Oinarria mutatuta dagoela, besoa muntatu behar da. Tutorial berdina jarraitu da besoa eraikitzeko. Besoak matxarda behar du muturrean latak hartu ahal izateko. Motor bakarra daukagu besoarentzat, hortaz, motor horren bidez bi eragileak (besoa + matxarda) mugitu behar dira. Diseinu horren berezitasun nagusiena hori da, besoa altxatu ahala matxarda itxi eta, besoa jaitsi ahala matxarda ireki egiten dela; sistema-mekaniko batek ahal bidetzen du hau. Hona hemen, eman diren pausorik garrantzitsuenak:

1→ Lehengo urratsean, erredukzio-kutxa bat gehitzen da, besoak gehituko duen pisuari aurre egin ahal izateko. Erredukzio hori, kasu honetan 1:3koa izan da. 4.31. irudian besoak daraman motorrean, behar izango den erredukzioa nola muntatu agertzen da.



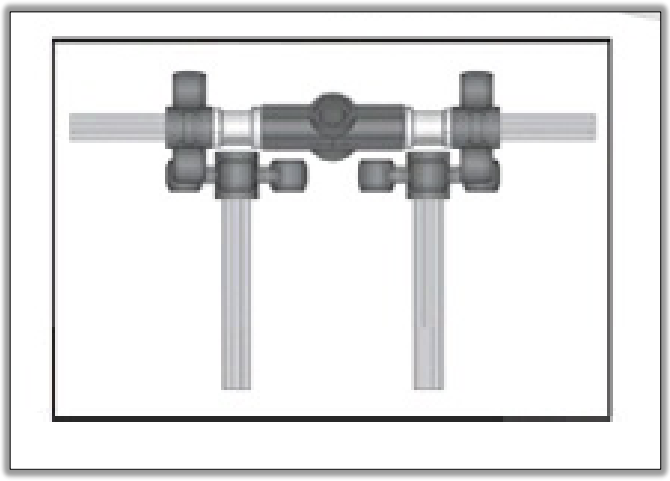
4.31. Irudia: Beso muntaketa. 1. urratsa

2→ Urrats honetan, kolore-sentsoreari kokaleku bat ematen zaio, matxardaren puntan. Kolore-sentsoreak lataren kolorea identifikatuko du, lata hartu ondoren kolore-sentsoreak oso distantzia txikira funtzionatzen du eta. 4.32. irudian ikus daiteke nola geratuko den sentsorea.



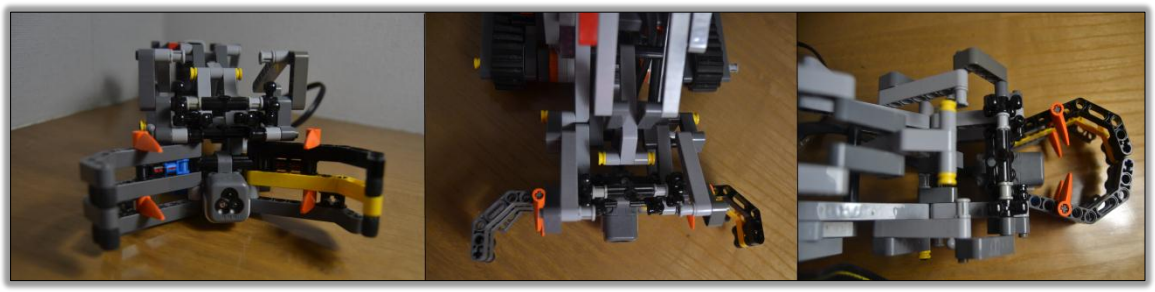
4.32. Irudia : Beso muntaketa. 2. urratsa

3→ Matxardaren zabaldu-itxi mugimendua egiteko, motorretik datorren mugimenduaren noranzkoa eta errotazio zentzua aldatu behar da. Aldaketa hauek 4.33. irudian agertzen den engranaje sistemarekin lortzen da.



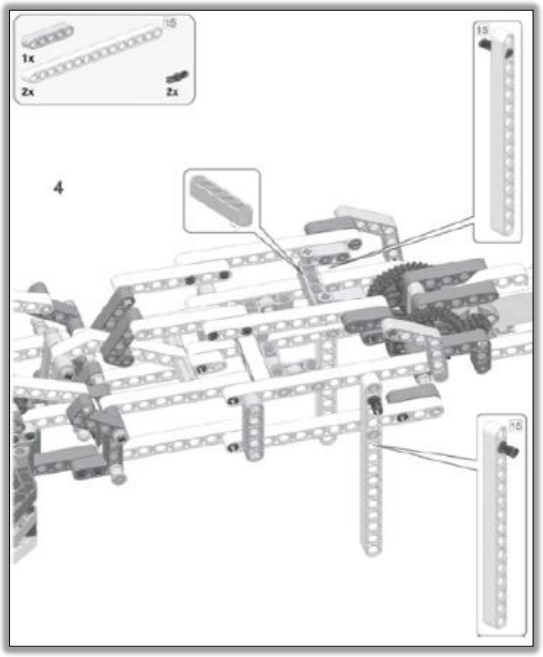
4.33. Irudia: Beso muntaketa. 3. urratsa

4→ Besoaren eraikuntzaren azkeneko urratsean, matxarda eraikitzen da. 4.34. irudiko argazkietan, ikus dezakegu nola geratu den besoaren muntaketa. Pieza laranjak, matxardak latak ondo hel ditzan kokatu dira.

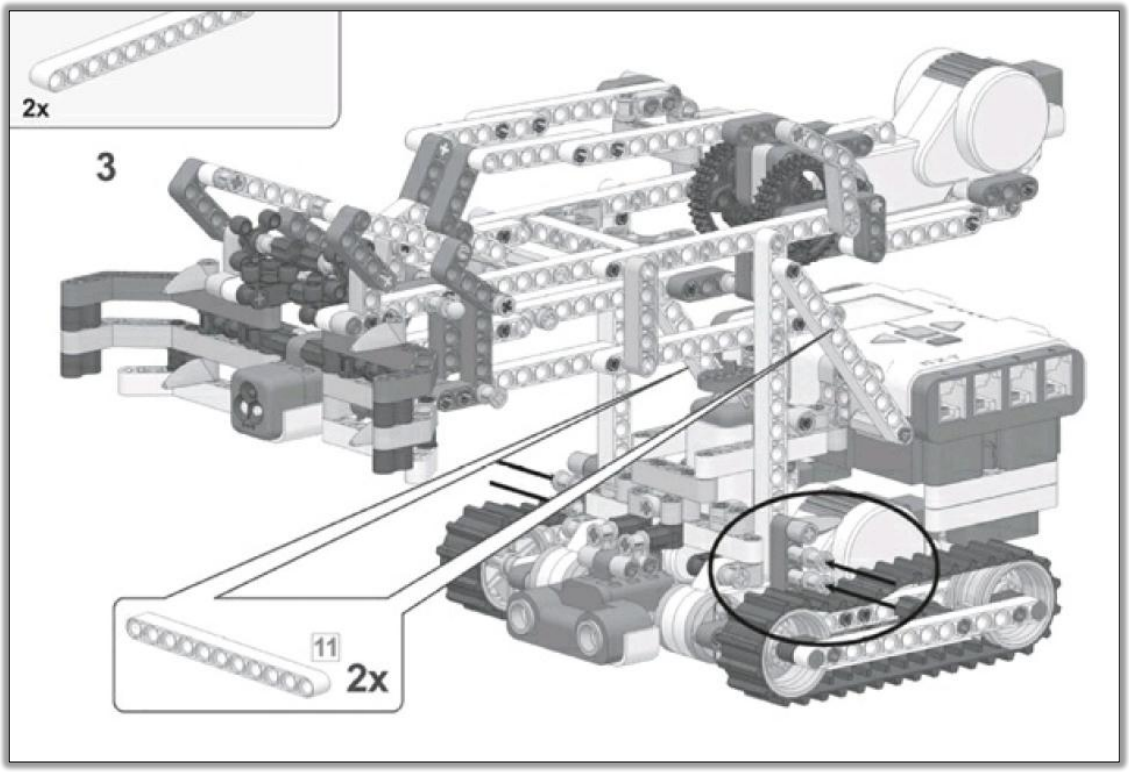


4.34. Irudia: Beso muntaketa. 4. pausua

Besoa eta oinarria eraikita daudela, bien arteko lotura finkatu behar da. Kontu handia izan behar da, etorkizunean besoa ez desmuntatzeko eta robotak latak ondo har ditzan. Lau piezekin lotzen dira oinarria eta besoa. 4.35. irudian, lehenengo bi pieza, besoaren zein ataletan kokatzen diren ikus daiteke. Pieza bakoitza besoaren alde batean kokatzen da eta besoaren bi piezetan finkatuta. Eta, azkenik, 4.36. irudian, azkeneko bi piezak, hiruki bat eginez, nola oinarriari finkatzen diren ikus daiteke.



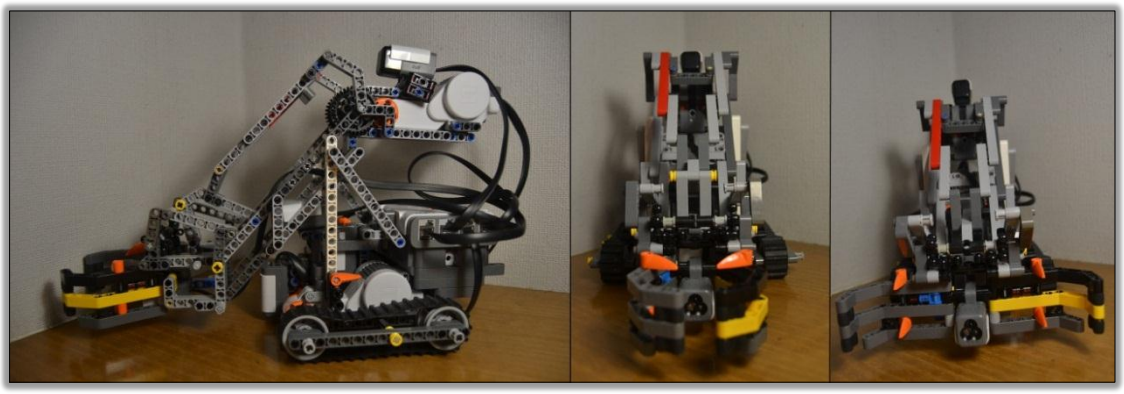
4.35. Irudia: Oinarria eta besoa lotzen 1



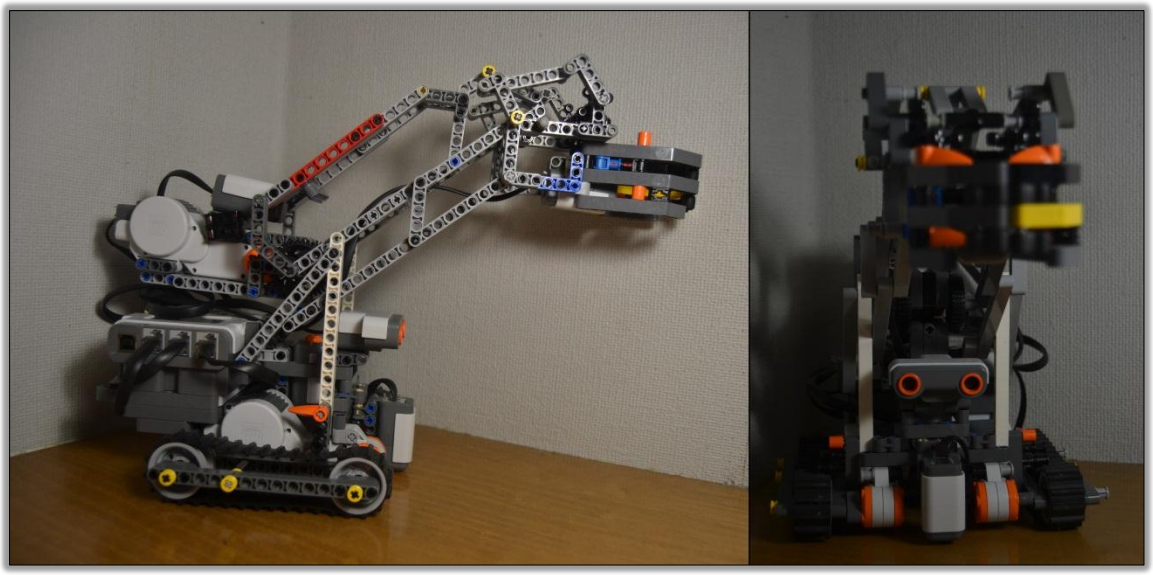
4.36. Irudia: Oinarria eta besoa lotzen 2

Robota eraikita dagoela, iparrorratza kokatzea falta da. 4.37. irudian, eskuinaldeko argazkian, besoaren goialdean ikusten den sentsorea iparrorratza da. Iparrorratza kokatzerako garaian leku ezberdinetan frogatu zen, eta errore magnetismorik ez zuela frogatutakoan, leku horretan kokatzea aukeratu zen.

4.37. eta 4.38. irudietan robotaren amaierako diseinua ikus daiteke.



4.37. Irudia: Robotaren morfología 1



4.38. Irudia: Robotaren morfologia 2

4.3. Portaeraren Garapena

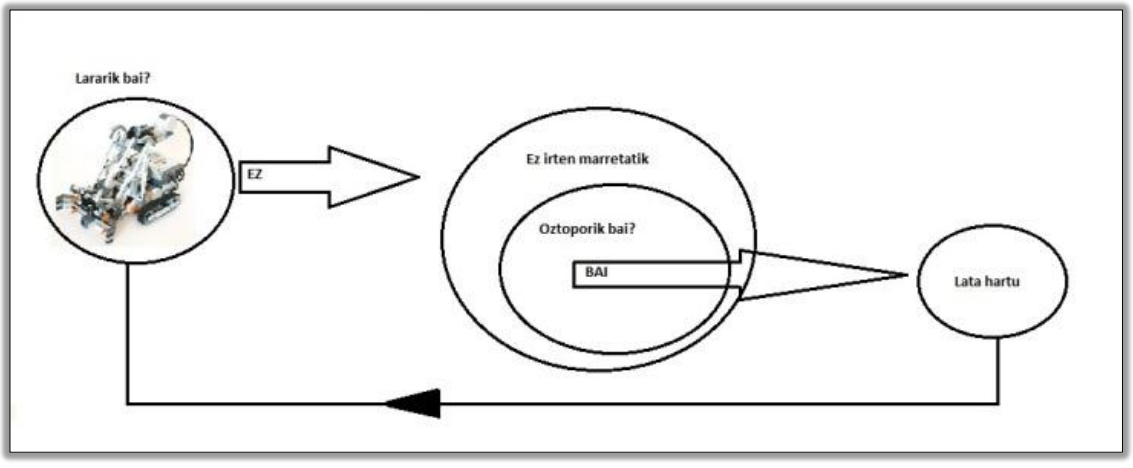
Robota muntatuta eta ROS eta NXTren arteko konektagarritasuna frogatuta dagoela, portaera inplementatzeko garaia heldu da. Lehengo urratsa, egoera kasuak definitzea izango da, egoera kasuak definitzerako garaian, emango diren egoera guztiak kontuan hartu behar izango da.

4.3.1. Egoera kasuak

Portaeraren deskripzioan, bi egoera kasu nagusi identifikatu daitezke. Lehengo egoera kasua, lataren bilaketa izango da. Eta bigarrena, behin robotak lata aurkitu duela lata hori sailkatzeko prozesua izango da.

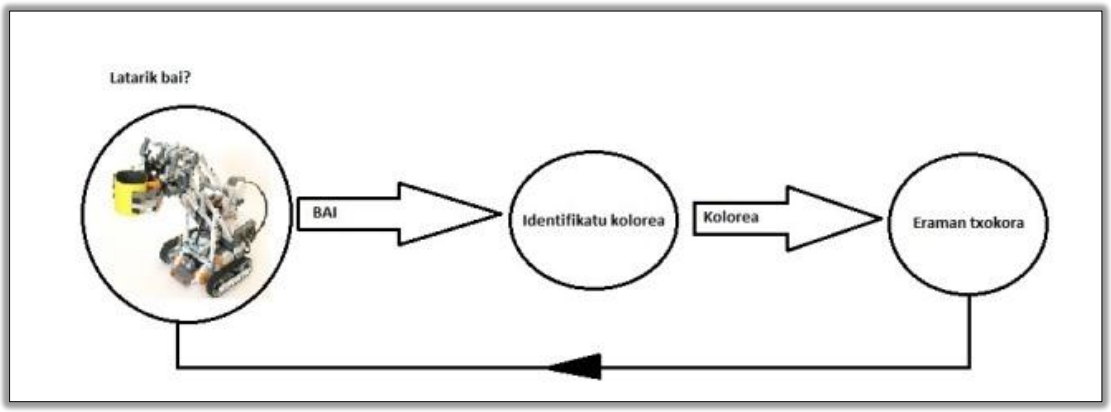
Latarik? EZ: Robotak latarik ez duenean aurrean, latak bilatzen ibili behar da eremuan, oztopo bat aurkitu arte.

Robotak eremutik ez irtetea izango du helburu nagusi moduan. Marra beltza aurkitzen duen momentuan buelta eman eta berriro lata bilatzen jarriko da. Baina lata bat bilatzen duenean lata hori hartu eta hurrengo egoera kasura pasatuko da. 4.39. irudian ikus dezakegu egoera kasu honen diagrama.



4.39. Irudia : Egoera kasua: Latarik? ez

Latarik? BAI: Robotak lata identifikatu eta hartu duenean, lata hori dagokion izkinara eramango du. Hau betetzeko, urrats batzuk jarraituko ditu. Lehendabizi lataren kolorea identifikatuko du, eta gero, identifikatu duen kolorearen arabera, kolore horri dagokion izkinara joango da. Azkenik, lata utzi eta beste lata berri bat bilatzera joango da. 4.40. irudian ikus dezakegu portaera orokor honen diagrama.



4.40. Irudia : Egoera kasua: Latarik? Bai

4.3.2. Inplementazioa

Egoera kasuak definitu direnean, robotak zer egin behar duen deskribatu da. Zabor biltzaile baten portaera izateko robota, hurrengo zerrendaren portaerak betetzeko gai izan behar da.

- Eremutik ez irten
- Oztopoak lokalizatu eta gerturatu
- Latak hartu
- Lataren kolorea identifikatu
- Lata kolorearen arabera sailkatu

Ataza guzti hauek bi nodo ezberdinetan garatu dira. Nodo bakoitzak, ataza ezberdinak exekutatuko ditu. Lehenengo nodoak, `Latarik_ez` egoera kasua garatuko du. 4.6. kode, nodo honen algoritmoa definitzen da.

1. NODOA (Algoritmoa)

```
1: brick_hasieraketa
2: Egoera: Lata ez
3:        Besoa_gora
4:        Oztoporik aurkitu ez duen bitartean:
5:        Zinta beltza?:
6:              Ez: AURRERA
7:              Bai: ZINTA_SAHIESTU
8:        Lata ikusten badu:
9:                Latatik_distantzia_jakinera_kokatu
10:              Mezua_Bidali: LATA AURKITU DUT
```

4.6. Kodea: 1. Nodoaren algoritmoa

Eta bigarren nodoak, aldiz, `Latarik_bai` egoera kasua garatzen du. 4.7. kodean nodo honen algoritmoa definitzen da.

2. NODOA (Algoritmoa)

```
11: Egoera: Lata bai
12:        Besoa_gora
13:        Kolorea_identifikatu
14:        Noranzkoa_bilatu
15:        Marra beltza aurkitu arte: AURRERA
16:        Noranzkoa_birkalkulatu
17:        Marra beltza aurkitu arte: AURRERA
18:        Besoa_behera
19:        Mezua_Bidali: LATA UTZI DUT
```

4.7. Kodea: 2. Nodoaren algoritmoa

Aplikazioaren diseinua zehaztu ondoren, sentsoreen portaera aztertu da. Ultrasoinu eta kolore-sentsoreetan kalibraketa beharrik ez dago, beti balio berak ematen dituzte. Baina, argi-sentsorean eta iparrorratzean bai. Kalibraketak uneoro ez egiteko *.launch* fitxategi bat eratzea erabaki da. Fitxategi honetan, ingurune zehatz bateko parametroak zehaztuko dira. Adibidez, unibertsitatean martxan jartzen bada, unibertsitateko balioentzako fitxategi bat eratu beharko da, baina beste leku batean martxan jartzea nahi bada, beste fitxategi berri bat eratu beharko da. Fitxategi honetan, ingurune aldagaien hasieratzeaz aparte, `roscore` eta nodoak exekutatuko dira.

Argi sentsorearen kasuan, robotak duen argi-sentsorea, zinta beltzaren gainean jarriko da, eta gero, zinta ez daukan lurzoruan. Hau frogatzeko, nxt_python paketean dagoen `sensor-test` programa exekutatuko da. Programa honek, sentsorearen balioak pantailaratzen ditu. Esandako bi balioak eskuratu direnean, bien arteko batazbestekoa ateratzen da eta hau izango da sentsorearen erreferentzia moduan jarriko den balioa *.launch* fitxategian.

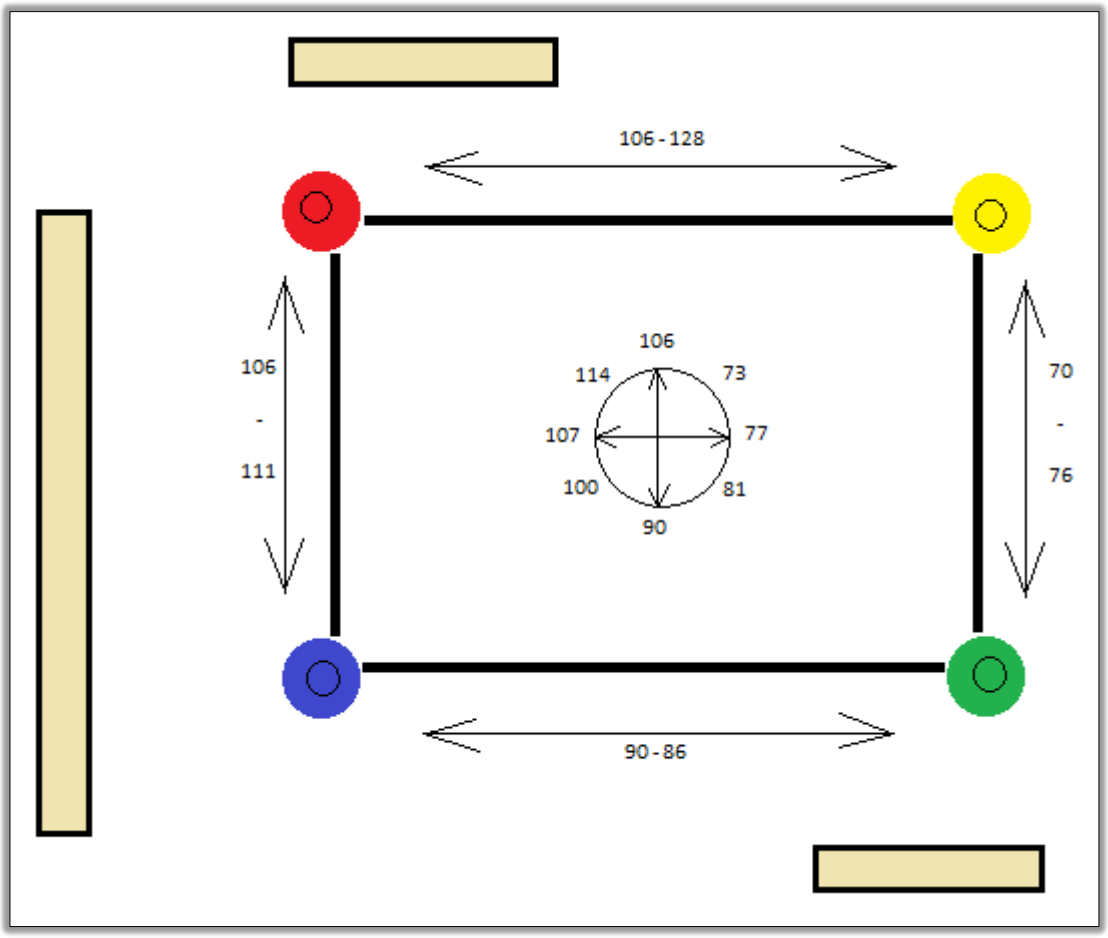
Iparrorratzaren balioak kalibratzeko, argi-sentsorearen prozesu bera jarraitu da. `nxt_python`-en dagoen fitxategi bat erabili da, iparrorratzaren balioak eskuratzeko. Baina kasu honetan, azterketa prozesua luzeagoa izan da, iparrorratza interpretatzeak duen konplexutasunagatik. 4.41. irudian, robotaren lan-eremua irudikatu da. Robota 5 puntu ezberdinetan jarri da, hauek borobil beltzekin marraztu dira. Puntu bakoitzeko, noranzko guztien balioak hartu dira, erdiko borobila bezala. Ondorio gisa, bilgune batetik bestera joateko batezbesteko noranzkoaren balioak finkatu dira, 4.10. taulan agertzen direnak.

```
Compass 1 blue-green= 90

Compass 2 blue-red = 107

Compass 3 red-yellow = 106

Compass 4 yellow-green = 77
```

4.11. Taula: Iparrorratzaren batezbesteko balioak

Robotak latak sailka ditzan, bi koloreen arteko noranzkora begira jarriko da eta aurrera egingo du marra beltza aurkitu arte. Marra aurkitu duenean, robotak beste erpinaren arteko balioaren noranzkora begira jarriko da, marra beltza aurkitu arte.

Behin marra beltza hau aurkitu duela, robota prest dago azkeneko biraketa bat egiteko eta lata bere kolorean uzteko.



4.41. Irudia: Iparrorratza kalibratzeko azterketa

Garatu den aplikazioak, lehen esan bezala, bi nodo ditu. Nodoak elkarren artean komunikatzeko ROSeko mezuak bidaltzen dituzte, *publisher* eta *subscriber*-en bitartez. Bi nodoak denbora guztian martxan egongo dira, txanda duenak, atazak exekutatzen, eta bestea, nodo honen mezua itxaroten egongo da.

Erabili den mezua, ROS eta NXT-ren arteko konektagarritasuna frogatzeko proban erabili dena izan da. Bai *subscriber* eta *publisher*-a ere bai. Eta jarraian, 4.6 eta 4.7 kodeko azpi-portaera nagusienak azalduko dira.

- Aurrera→
    - Robotak aurrera egingo du V= 6 cm/s abiaduraz, motorren %100 potentziaz.
    - Noiz arte? Sentsore irakurketa bateko baldintza bete arte.

- Zinta saihestu→
    - Atzera joko du 0,3 segundoz eta biratu egingo da 0,7 segundoz V= 6 cm/s-ko

- Latatik distantzia jakinean kokatu→
    - Atzera bota
    - Noiz arte? Robota latarekiko 27 cm-ra heldu denean
    - Zergatik? Ultrasoinu-sentsorea oinarrian dagoelako eta besoa oinarritik 25 cm ateratzen delako.

- Besoa gora/behera→
    - Besoa gora edo behera egiten denean, lata hartzeko edo uzteko da.
    - Eszepzioa: Lehengo aldian, robotaren besoa ultrasoinu-sentsoreari oklusioa egiten diolako, orduan zabor bilaketa prozesua hasten denean besoa goian jarri behar da.
    - V= 4cm/s-ko, motorren %100 potentziarekin. Besoa igotzeko erredukzio kutxa jarri da, orduan potentzia berberarekin, abiadura motelago lortzen da.

- Kolorea identifikatu→
    - Kolore sentsoreak lataren kolorea irakurtzen du.
    - Zertarako? Robota lata zein izkinara joan behar duen jakiteko.

- Noranzkoa bilatu→
    - Robota biratu, Kolorearen arabera biraketa eskuinera edo ezkerrera izango da.
    - Robotaren buruaren ardatzean. Motor bat aurrera, bestea atzera.
    - Noiz arte? Kolore horren lehenengo noranzkoa aurkitu arte.

- Noranzkoa birkalkulatu→
    - Robota biratu, kolorearen arabera biraketa eskuinera edo ezkerrera izango da.
    - Robotaren buruaren ardatzean. Motor bat aurrera, bestea atzera.
    - Robota sailkapen prozesuaren lehenengo marra beltzean dago eta kolorearen azkeneko noranzkoa bilatuko du biratuz.

- Mezua bidali→
    - Robotaren egoera bidaltzen da.
    - *Susbcriber* klase baten bidez *topic* batean idazten da.

4.4. Esperimentuak

Robotak, diseinatu den portaeraren aurrean izan dezakeen egoeren frogak egin dira. Atal honetan, egin diren esperimentuak azaltzen dira eta gertatutako aztertzen da. Gainera, frogen bideoak eransten dira dokumentu honekin.

1. **ESZENATOKIA GARBI:** Esperimentu honetan, eszenatokiak latarik ez du izango. Orduan, robota eszenatokia mugatzen duen karratutik ezin izango da irten.

Esperimentuak_001 bideoan ikus dezakegu, robota karratutik ez dela irteten. Robotak zinta beltza aurkitzen duenean eszenatokitik irtengo dela suposatzen du eta buelta ematen du.

Esperimentuaren emaitza: Arrakastatsua.

2. **LATA ROBOTAREN AURREAN:** Esperimentu honetan, robotari aurre-aurrez jarriko zaio lata bat. Robotak ez du inolako arazorik izango lata aurkitzeko. Lata hartu behar du eta bere kolorearen erpinera eraman behar du.

Esperimentuak_002 bideoan ikus dezakegu nola robotak lata hartzen duen eta lata kolore horren erpinera eramaten duen.

Batezbesteko denbora: 1:20

Esperimentuaren emaitza: Arrakastatsua.

3. **LATA BAKARRA EREMUAN:** Lata edozein tokian kokatuko da. Robota lata bilatzen egongo da eta lata aurkitzen duenean, lata hartu eta bere erpinera eramango du. Robotak, lata utzitakoan, beste lata baten bila abiatuko da.

Esperimentuak_003 bideoan ikus daiteke robotak latak sailkatzen dituela. Robotak lata uzten du eta hurrengo lata bilatzen hasten da.

Batezbesteko denbora: 4:00

Esperimentuaren emaitza: Arrakastatsua.

4. **KOLORE BERDINEKO LATAK EREMUAN:** Robotak lata bat bere lekuan utziko du eta beste baten bila joango da, baina kasu honetan kolore berdineko lata izango da.

Esperimentuak_004 bideoan ikus daiteke robota latak ondo sailkatzen dituela. Honekin, konproba daiteke, robotari berdin zaiola zein koloreko latak hartzen dituen, beti kolore horren erpinera eramango duela.

Batezbesteko denbora: 3:30

Esperimentuaren emaitza: Arrakastatsua.

5. **LATA BAT BAINO GEHIAGO:** Esperimentuak amaitzeko, azkeneko froga egin da. Bi lata karratuan jarriko dira. Eta robota gai izan behar da lata guztiak biltzeko.

Esperimentuak_005 bideoan bi latak hartzen dituela ikus daiteke. Baina egia da, latak ezin direla edonon jarri. Robotak lata bat duenean ez du ultrasoinua erabiltzen, orduan beste latak identifikatzerik ez dauka.

Batezbesteko denbora: 5:24

Esperimentuaren emaitza: Erdi-arrakastatsua.

3. zatia: Ondorioak

5. kapitulua:

Ondorioak

5. kapitulua: Ondorioak

Robota ROS-ekin kontrolatzeko,robota eta ROSeko prozesuak exekutatzen diren makinaren arteko komunikazioa behar da. ROSek TCP/IP protokoloa erabiltzan du eta NXTak USB bidezko konexioa soilik eskaintzen duenez, uneoro PCa eta NXTa kable bidez batuta mantendu behar dira, robotaren autonomia murriztuz. Arazoa da NXT-ren adreilua ez duela ROS-eko *host*-ari lotzeko gaitasuna. Lotune hau, ROS_MASTES_URI parametroaren bitartez egiten da. Konponbidea 6.kapituluan proposatzen da.

Robotak bete behar zuen portaerari begira, proposatu ditugun zehaztasun guztiak bete dira. Robotak, ataza guztiak bete ditu eta portaera inplementatzea posible izan da, izan ditugun baliabideekin.

Karrera amaierako proiektu hau gauzatu aurretik roboten funtzionamenduaren inguruan hainbat zalantza eta galdera zeuden. Proiektu honek robotikaren arloa hobeto ulertzeko balio izan du. Gainera robot batek ataza sinple bat gauzatzeko kontuan hartu behar diren atal desberdinen funtzionamendua eta koordinazioa ulertu da.

Azkenik, proiektuaren kudeaketari dagokionez, hasiera batean zehaztu ziren helburuak ez dira bete. Planifikazio kaskarra egin zen hasiera batean eta gainera proiektuan zehar izan diren arriskuak ez ziren ondo identifikatu. Egin zitezkeen akats guztiak bete ditut hilabete hauetan zehar. Baina positibo moduan, hurrengo proiektuetan akats hauetatik ikaste beharra ateratzen dut, berriro gerta ez dadin.

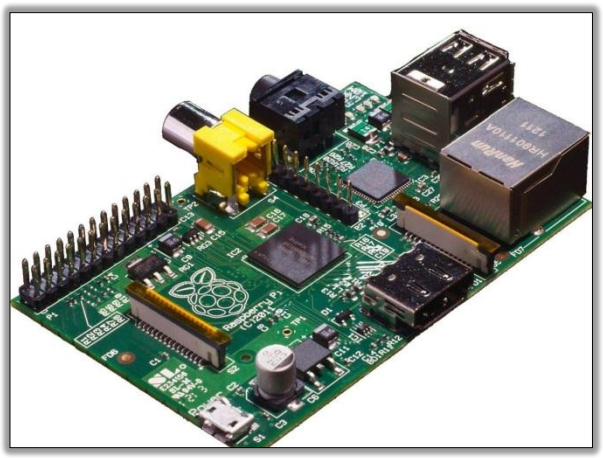
6. kapitulua:

Etorkizuneko lana

6. kapitulua: Etorkizuneko lana

Proiektu honetan garatu den robotak arazo txiki bat dauka. Robot ez autonomo bat eraiki da. Robotak ROS edukitzeko plaka ahaltsu bat behar du, bere brick-a kontrolagailu bat baino ez da. Autonomoa bihurtzeko konputazio gailu bat erantsi behar zaio, kasu honetan robota USB-aren bitartez konektatzen zaio ordenagailuari.

Hau konpontzeko, Raspberry Pi bat erabili daiteke. Raspberry Pi, kreditu txartelen tamainadun eta zirkuitu-plaka bakarreko ordenagailu pertsonalen serie bat da. Eskoletan konputazio zientzien irakaskuntza sustatzeko asmoarekin garatu zen. Raspberry Pi-ak mahaigaineko ordenagailuek egiten dituzten lan gehienak egiten ditu, adibidez, kalkulu-orriak, testu prozesua eta bideojokoak. 6.46. irudiko txartela, Raspberry Pi-a da.



6.42. Irudia: Raspberry Pi

Raspberry Pi-ak, Broadcomen BCM2835 txip bakarreko sisteman oinarritzen da. ARM1176JZF-S 700 MHzeko prozesadorea dauka, VideoCore IV GPUa, eta hasieran 256 MBeko RAM memoria zeukan. Raspberri Pi-ak ez dauka disko gogorrik. Sistema abiarazteko eta memoria iraunkor bezala, SD txartela dauka. Modelo ezberdinak daude eta bakoitzak espezifikazio ezberdinak ditu. Denborarekin espezifikazioak hobetzen joan dira.

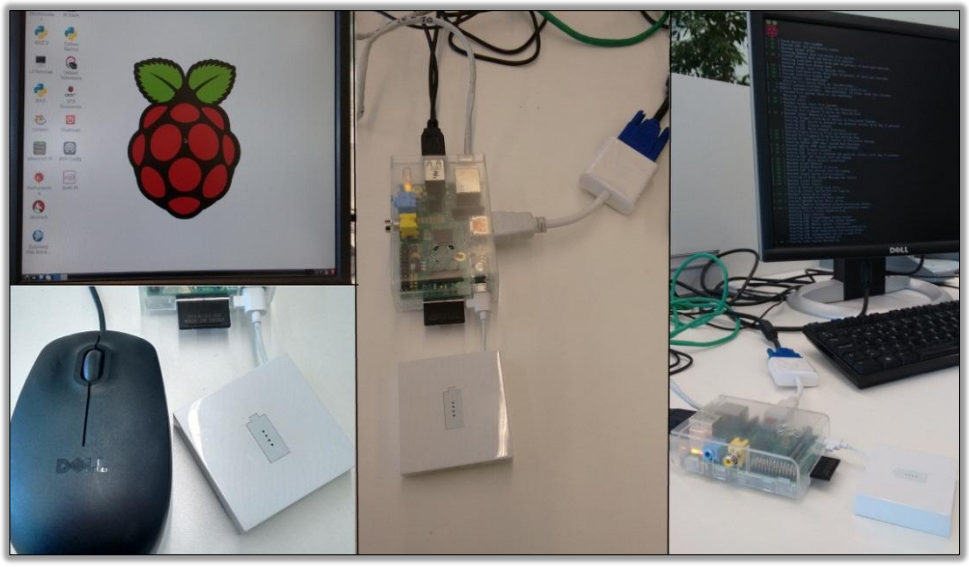
Raspberry Pi fundazioak ARM prozesagailuentzat egokitutako LINUX banaketa desberdinak jartzen ditu eskuragarri deskargatzeko. Gure kasuan ROS Groovy

instalatu behar dugu eta *Raspbian Wheezy* bertsioa deskargatu eta instalatu behar izango da.

Raspberry Pi-ak HDMi konektorea dauka eta pantaila batetara konekta daiteke, ordenagailu bat balitz bezala. Bi USB konexio dauzka ere bai non arratoia eta teklatua atzitu daiteke. Honek sistema instalatzea utziko digu eta behin programaketa guztia eginda dagoela, robotaren muntaketa pixka bat eraldatu beharko genuke Raspeberry Pia kokatzeko.

Nola egin dezakegu *Raspberry Pi*-aren elikadura? Normalean korronteari entxufatuta doa, baina robota autonomoa bihurtu nahi bada, beste era bat aurkitu behar da. Mugikorretan erabiltzen den kanpo-bateria bat erabiltzea izan daiteke aukera bat. Raspberry Piak micro-USBa erabiltzen du elikatzeko, mugikorrek bezala. Bateria mota hau, Raspberry Pi bat konektatzean, zenbat irauten duen aztertu beharko da.

Raspberry Pi bat instalatzea izan dut, baina ez da aukerarik izan proiektuaren robotean probatzeko. 6.47. irudian, ikus daiteke egin den instalazioa. Pantaila, teklatua eta arratoia konektatuta dauzka, *wifi* konexioa duenez, urruneko konexio baten bidez exekutatu ahal izango lirateke programak.



6.43. Irudia: Raspberry Pi-aren instalazioa

Bibliografia

[1] ROSeko tutorialak. www.ros.org

[2] Tully Foote, Wim Meeussen, eta Melonee Wise. NXT paketearen ROS bertsioa. www.ros.org/wiki/Robots/NXT

[3] Laurens Valk: *The LEGO Mindstorms NXT 2.0 Discovery Book*

[4] NXT sentsoreen espezifikazioak. www.ro-botika.com

[5] Motorren espezifikazioak. http://www.philohome.com/nxtmotor/nxtmotor.htm

[6] Ruth Suehle eta Tom Callaway: *Raspberry Pi Hacks*

[7] Ubuntu bertsioa eskuratzeko. http://www.ubuntu.com/download/desktop

[8] Ubuntu instalatzeko. http://www.ubuntu.com/download/desktop/install-ubuntu-desktop

[9] ROS GROOVY instalaketa. http://wiki.ros.org/groovy/Installation/Ubuntu

A Eranskina

Eranskin honetan, NXT eta ROS-en arteko konexioa egiteko jarraitu behar diren pausuak azalduko dira. Zein bertsio instalatu behar den azalduko da, eta dena prest dagoenean, exekutatu behar diren komandoak agertzen dira.

1. **Ubuntu instalatu.**

Ubunturen 12.04 bertsioa instalatu behar da. [7] Ubuntu bertsioa eskuratzeko, www.ubuntu.com web orritik deskargatu .iso fitxategia eta ubuntu-ko partizio bat sortu, ([8] Ubuntu instalaketa. http://www.ubuntu.com/download/desktop/install-ubuntu-desktop) web orri hau jarraitu instalatzeko.

2. **ROS instalatu.**

ROS GROOVY bertsioa instalatu behar da. [9] ROS Groovy instalazioa, http://wiki.ros.org/groovy/Installation/Ubuntu.

3. **NXT paketea instalatu.**

- Dokumentu honekin eransten den paketea eskuratu.
- ROS-eko *workspace*-a eratu.

```
$rosws init ~/rosbuild_ws /opt/ros/groovy

$mkdir –p ~/rosbuild/src

--Hurrengo komandoa .bashrc fitxategian itsatsi.

$source ~/rosbuild_ws/setup.bash
```

- Ordenagailuaren konfigurazioa.

```
$sudo groupadd lego

$sudo usermod –a –G lego <username>
```

```
$echo “SUBSYSTEM==\”usb\”, ATTRS{idVendor}==\”0694\”, GROUP=\”le
go\”, MODE=\”0660\””> /tmp/70-lego.rules && sudo mv /tmp/70-lego
.rules /etc/udev/rules.d/70-lego.rules

$sudo restart udev
```

- NXT paketea instalatzen:
    - Nxt paketea ~/rosbuild_ws/src paketean kopiatu.
    - `$cd ~/rosbuild_ws`
    - `rosmake`
- Portaera exekutatzen:
    - `roslaunch nxt_kap lateroa_in_cave.launch`