Universidad del País Vasco

Euskal Herriko Unibertsitatea

Pololu 3pi robotaren urrutiko kontrola

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

Konputagailuen ingeniaritza

Egilea: Telmo Aldalur Goenaga

Tutoreak: Elena Lazkano eta Txelo Ruiz

Donostia, 2014-ko Ekaina

Esker onak

Bost urte Informatika fakultatean igaro ondoren iritsi zaigu guzti honi amaiera emateko garaia, ez da erraza bukaera ematea, bizitzako ordu asko eta asko pasa baititugu eraikin honetan. Zenbat lan, zenbat ordu ikasten, zenbat poz… zenbatezinak izango dira bertan igarotako momentu bikainak. Azkar igaro dira 5 urte hauek eta bukaera emateko proiektu interesgarri bezain politari ekin genion 2014-ko urtarrilean.

Ahaztu baino lehen eskerrak eman Elena Lazkanori eta Txelo Ruiz-i beti egon baitira niri laguntzeko prest. Eskerrak eman 5 urtean zehar eduki ditudan klase kideei, beraien laguntzarik gabe ez nintzateke izango eta orain naizena. Eta azkenik eskerrak familiari eta neska lagunari beti egon baitzarete nire ondoan.

Proiektu honi heldu izanaren arrazoiak oso argi izan nituen hasieratik. Informatikaren arlo zabaletik, gehien gustatzen zaidan atala robot eta hardwarerena da. Honenbestez Txelok proposatu zidan momentutik argi izan nuen hau izango zela nire proiektua. Bestetik aipatu behar da bideojokoen eta teknologiaren mundua oso gustuko dudala eta horregatik Wii-ko urrutiko aginte batekin lana

Laburpena

Denontzat jakina da, azken urteetan robotikaren munduak zelako garrantzia hartu duen teknologiaren munduan. Hurrengo urteetan badirudi hainbat lanetan giza faktorearen beharra txikiagoa izango dela, roboten erabilera handiagotzen baitoa. Robotak lana modu eraginkor batean egiten du eta horrek bere abantailak ditu, bai produktibitate aldetik, bai alde ekonomikotik.

Proiektu honetan *Pololu 3pi* izeneko robot txiki batekin egin da lan. Nahiz eta robot honek bere mugak badituen, aisialdiko eremuetan pieza garrantzitsua izan daitekela uste da. Robot honek eduki dezakeen ohiko erabilera bat da lurrean marrazturik dagoen labirinto batean ibiltzea marrari jarraituz.

Proiektu honetan sistema bat garatu nahi izan da, *Pololu 3pi* robotaren mugimendua urrutiko aginte batez kontrolatzeko, non erabiltzaileak aukeratu ahal duen zein funtzionalitate nahi duen robota kontrolatzeko: Biraketa Zoroa, Mugitu azeleratuz, Wiimote-a mugituz eta Biraketa normal.

Hortaz, proiektuan bi zati uztartzen dira, alde batetik *Pololu 3pi* robota eta bestetik urrutiko aginte bat, kasu honetan *Wii* bideokontsolaren *Wiimote*-a. Programa bat sortu da *Pololu 3pi*-a *Wiimote*-arekin mugitu eta kontrolatu ahal izateko. Hori posible izan da *Atmega328* izeneko mikrokontrolagailuaren bitartez, berari esker programatu ahal izan baita robotaren programa. Aplikazioak bi zati ditu, alde batetik ikusgai zaigun interfaze grafiko bat, non robotaren funtzionalitate ezberdinak ikusten diren PC-aren pantailan. Bestetik robotak berak duen programa, interfazetik eta *Wiimote*-tik bidaltzen diren aginduei kasu egiten diena.

Gaien aurkibidea

Irudien aurkibidea

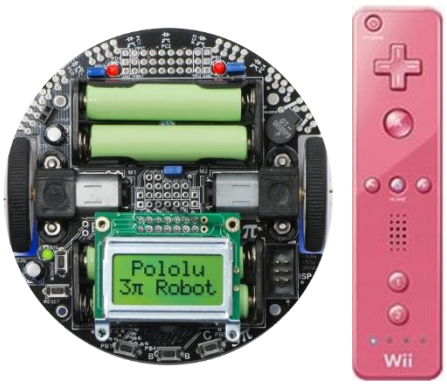
1. Kapitulua

1 Motibazioa

Denontzat jakina da, azken urteetan robotikaren munduak zelako garrantzia hartu duen teknologiaren munduan. Hurrengo urteetan badirudi hainbat lanetan giza faktorearen beharra txikiagoa izango dela, roboten erabilera handiagotzen baitoa. Robotak lana modu eraginkor batean egiten du eta horrek bere abantailak ditu, bai produktibitate aldetik, bai alde ekonomikotik.

Gaur egun teknologiaren laguntzaz elbarriak diren edo ezintasunak dituzten pertsonek aurrerapen handia lortu dute beraien autonomiari dagokionez. Esku robotikoek lagundu dezaketen moduan, urrutiko aginte baten bitartez kontrola daitezkeen robotek ezinduen bizitza errazagoa egiten lagun dezakete. Bada merkatuan *roomba* izeneko robot bat, etxean xurgagailuaren lanak egiten dituena. Erosoa da eta lana fin egiten du, baina adibidez urrutiko aginte batekin kontrolatzea ongi egon litekela uste da zuk nahi duzun lekua garbi baitezake.

Proiektuaren onura moduan, esan daiteke erabili diren teknologiak nahiko merkeak izan direla eta horrek erakusten du proiektu probetxugarri bat egiteko ez dela dirutza gastatu behar. Hurrengo 1.1 irudian ikus daiteke erabili diren bi osagai nagusiak: *Pololu 3pi* robota eta *Wiimote*a.



1.1 Irudia: *Pololu* eta *Wiimote*-a

Pololu-ren ezaugarrietako bat hardware librea dela da honek dakartzan onurekin, *Pololu*-k erabiltzen duen teknologia *arduino*-n oinarritzen baita. Azken urteetan *arduino* eta mikrokontrolagailuen arloa geroz eta garrantzi handiagoa hartzen ari da teknologia eta

informatika munduan. Geroz eta ugariagoak dira *Arduino* plataformaren baitan egiten diren proiektu interesgarriak.

Arduino, mikrokontrolagailu plaka baten eta garapen ingurune batean oinarritzen den hardware libreko plataforma bat da. Eta gehien erabiltzen diren mikrokontrolagailuen artean: *Atmega168, Atmega328, Atmega1280* eta *Atmega8* daude, eta hain zuzen *Atmega328* mikrokontrolagailua da proiektuan erabili dena.

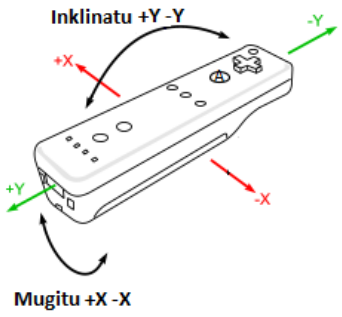
Wiimote-a *Wii* bideo kontsoletako periferiko nagusia da, urrutiko agintearen funtzioa betetzen du. Hainbat botoi ditu *Wiimote*-ak 1.1 irudian ikus daitekeen bezala, baina proiektuan bertako sei botoi erabiliko dira, hau da 1.2 irudian ikus daitezkeenak: *up, down, left, right, A* eta *B*.

1.2 Irudia: *Wiimote*-tik erabili diren botoiak

Norabide botoiekin (*up, down, left* eta *right*) robotak hartuko duen norabidea kontrolatuko da. Aldiz, *A* botoia *Wiimote*-a mugituz funtzionalitatean erabiliko da robota mugi dadin eta *B* botoia robotaren abiadura jaisteko erabiliko da.

Proiektuaren helburu nagusia *Pololu* robotaren urrutiko kontrola lortzea zen *Wiimote* baten bitartez. Proiektua aurrera joan ahala robotak gauza ezberdinak egin zitzakeela konturatu zen, hau da robotak portaera ezberdinak izan zitezkeela. Horregatik 4 funtzionalitate ezberdin inplementatzea erabaki zen:

- **Biraketa Zoroa:** Funtzionalitate honen berezitasuna, biraketa zoroak egiten dituela da. Robota martxan jarri aurretik abiadura aukeratu behar zaio robotari.
- **Mugitu azeleratuz:** Bere berezitasuna norabide botoiak erabiltzean robotaren abiadura azeleratu egiten dela da. Horretaz gain abiaduraren muga bat izango du ezingo duena hortik pasa.
- ***Wiimote*-a mugituz:** Robota *Wiimote*-ko norabide botoiei sakatu gabe mugitzeko aukera eskaintzen du. Robota mugitzeko 1.3 irudian agertzen den mugimenduak egiten dira *Wiimote*-arekin.
- **Biraketa normal:** Funtzionalitate honen berezitasuna 90 graduko biraketak egiten dituela da. Robota martxan jarri aurretik abiadura aukeratu behar da robotarentzat.



1.3 Irudia: *Wiimote*-aren mugimendu eskema

2. Kapitulua

2 Planifikazioa

2.1 Proiektuaren helburuak

Proiektuaren azken helburua *3pi* robotaren urrutiko kontrola gauzatzea den arren, proiektua garatzen hasi aurretik, honek eskaintzen zituen interesak eta helburuak zerrendatzea egokia zela pentsatu da. Honakoak dira:

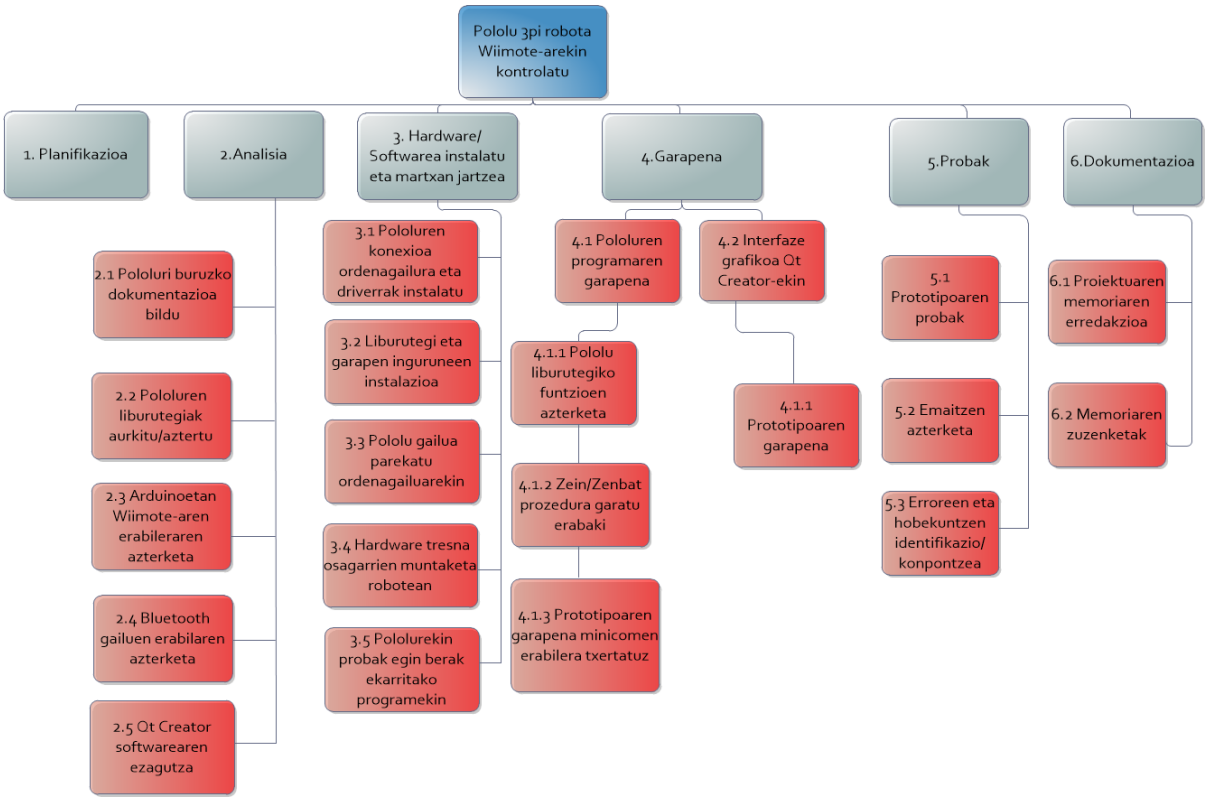
- *Pololu 3pi* robotaren ikerketa
    - Robotaren erabilera aztertu.
    - Beharrezko software eta hardware gailuen bilketa/azterketa robotarekin probak egin ahal izateko.
    - Robotaren instalazio, proba eta konfigurazioa.
- Urrutiko aginteen azterketa
    - *Wii* eta *PS3*-ko urrutiko aginte, *Joystick* eta halako gailuekin *Pololu* bezalako robot bat kontrolatzea posible zela egiaztatu.
    - Bideojokoetako urrutiko aginteekin egindako proiektuak topatu.
    - Urrutiko aginteen portaera aztertu *arduino* gailuekin.
- Robot eta PC-aren arteko *bluetooth* konexioa aztertu
    - Beharrezko driverrak instalatu PC-an *bluetooth*-a konfiguratzeko.
    - PC-*Pololu* arteko komunikazioa gauzatu serie konexioa emulatuz *bluetooth* bitartez.
    - Komunikazio hori modu garbiago batean egiten saiatu.
- *Wii-ko* urrutiko agintea eta robotaren arteko komunikazioa aztertu
    - *Wi-ko* urrutiko agintea *PC*-rekin konektatzeko modu ezberdinak aztertu.
    - *Wii-ko* urrutiko agintearen erabilera ezberdinak aztertu.

Hauek izan ziren proiektua hasi aurretik finkaturiko helburuak, baina normala den bezala proiektuak aurrera egin ahala helburu berriak sortu dira proiektua bera hobetzeko eta ikusgarriagoa egiteko asmoz. Proiektuan zehar egin den aldaketarik garrantzitsuenetako bat interfaze grafiko baten sorrera izan da. Hasiera batean ez zegoen aurreikusirik interfaze grafiko bat egitea, errazagoa ikusten baitzen *minicom* (ikus 3.2.6 atala) izeneko programa batekin aurrera egitea. Baina irakasleen laguntzaz konturatu ginen ez zela eroso eta ikusgarria, eta honenbestez interfaze grafikoa sortzeari ekin zitzaion. Honakoak dira aldaketa horrekin sortu ziren helburu berriak:

- Interfaze grafiko baten sorrera ikusgarritasuna emateko
    - Horretarako beharrezko software baten bilaketa.
    - *Gt* liburutegien erabilpenaren dokumentazio lana.
    - Interfaze grafiko bat sortu, *PC-Pololu* arteko komunikazioa argiago izan dadin *minicom*-ekin baino.
    - Robotaren programa berri baten sorrera, interfazea proiektuan integra dadin ahalik eta aldaketa gutxien sorraraziz.
- Aplikazioaren erabilpenaren ebaluaketa
    - Aplikazio osoaren proba ezberdinak, zer hobetu ikusteko.
    - Interfazeko detaileak perfekzionatu.

2.2 LDE diagrama

Ondorengo diagraman ikus ditzakegu proiektua arrakastatsua izan dadin garatu beharko diren ataza eta azpiatazak.



2.1 Irudia: LDE Diagrama

Hona hemen ataza nagusien azalpena:

- **Plangintza:** Proiektua aurrera atera ahal izateko egin beharreko plangintza. Atal honetan islatuko da proiektuan zehar egin den lana.
- **Analisia:** Proiektua garatu ahal izateko eskuratu behar izan den ezagutza lortu da atal honetan. Hala nola interneten, liburuetan, edo irakasleen bitartez lortu da beharrezko ezagutza.
- **Hardware/Softwarea instalatu eta martxan jartzea:** Atal honetan proiektuan erabili izan diren software eta hardwarea instalatu eta montatu dira.
- **Garapena:** Izenak dioen bezala, aplikazioaren garapena.
- **Probak:** Atal honetan, inplementazioan zehar eta inplementazioa bukatzean robotarekin egin diren proba ezberdinak sartzen dira. Aplikazioaren akats ezberdinak zuzendu dira.
- **Dokumentazioa:** Proiektuko memoriaren idazketa eta zuzenketa.

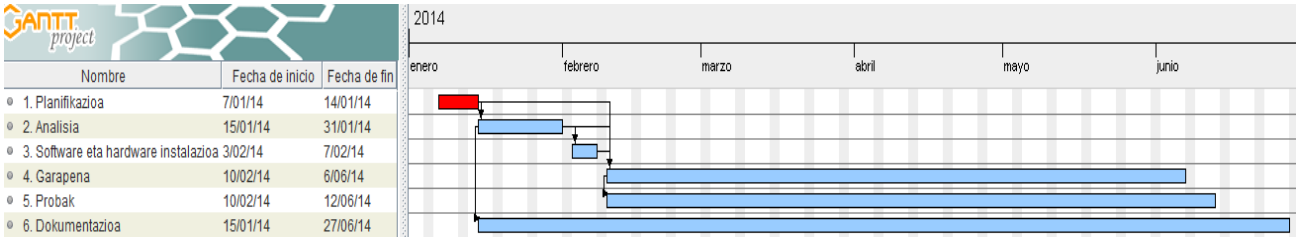
2.3 Kudeaketa

Atal honetan proiektuko kudeaketaren erabakiak azalduko dira. Alde batetik *Gant* diagrama izango dugu eta bestetik atazen denbora taula. *Gant* diagraman proiektuko atazen denbora lerroa eta atazen arteko dependentziak islatuko dira. Aldiz atazen denbora taulan ataza bakoitza egiteko planifikatuko denbora eta hartutako denbora agertuko dira.

Proiektua 2014-ko urtarrilean hasteko planifikatua zegoen, 2. lauhilabetearen hasierarekin batera eta urte bereko uztailean bukatzeko. Nire plana astelehenetik ostiralera egunean 4 ordu lan egitea izan da. Azken hilabetean kopuru hori aldatu egin da, 6 orduko lan egunak izatera igaroaz.

2.3.1 Gantt Diagrama

Diagrama honetan LDE diagraman identifikatutako atazak noiz egin diren adierazten da.



2.2 Irudia: *Gantt* diagrama

2.3.2 Atazen denboraren taula

ATAZA NAGUSIA	AZPI ATAZA	PLANIF. DENBORA	AZKEN DENBORA
1. Planifikazioa		**14 h**	**12 h**
2. Analisia	Pololuri buruzko dokumentazioa bildu	7 h	10 h
	Pololuren liburutegiak aurkitu/aztertu	3 h	5 h
	Arduino gailuetan Wiimote-aren erabileraren azterketa	4 h	5 h
	Bluetooth gailuen erabileraren azterketa	4 h	7 h
	Qt Creator softwarearen ezagutza	4 h	6 h
	Guztira	**22 h**	**33 h**
3.Hardware/Software Instalatu eta martxan jartzea	Pololuren konexioa ordenagailura eta driverrak instalatu	5 h	6 h
	Liburutegi eta garapen inguruneen instalazioa	5 h	3 h
	Pololu gailua parekatu ordenagailuarekin	1 h	3 h
	Hardware tresna osagarrien muntaketa robotean	2 h	3 h
	Pololurekin probak liburutegiak ekarritako programekin	6 h	6 h
	Guztira	**19 h**	**21 h**
4.Garapena	Pololuren programaren garapena	80 h	97 h
	Interfaze grafikoa Qt Creator ekin	80 h	114 h
	Guztira	**160 h**	**211 h**

5.Probak	Prototipoaren probak	18 h	20 h
	Emaitzen azterketa	15 h	12 h
	Erroreen eta hobekuntzen identifikazio/konpontzea	30 h	33 h
	Guztira	**63 h**	**75 h**
6. Dokumentazioa	Proiektuaren memoriaren erredakzioa	80 h	84 h
	Memoriaren zuzenketak	5 h	6 h
	Guztira	**85 h**	**90 h**
Pololu 3pi robotaren urrutiko kontrola	**Guztira**	**363 h**	**442 h**

2.4 Komunikazioa

Komunikazioa proiektuan irakasle-ikaslearen artean ona izan da, zalantzaren bat zegoen bakoitzean irakaslearen laguntza jasotzen baitzuen ikasleak. Proiektuan zehar hainbat bilera izan dira ikaslearen eta irakasleen artean. Bilera hauek hasieran eta bukaera partean izan dira gehienak. Bilera hauen zehaztasunak ikusi nahi izanez gero C erasnkinean azalduko dira.

2.5 Aldaketak

Proiektuan zehar hainbat aldaketa gertatu dira hainbat faktoreren ondorioz, planifikatua zegoenarekin alderatuz. Aldaketa batzuk beharrezkoak izan dira garapenean gertatu diren egoerengatik, baina aldiz beste batzuk ezustean egin beharreko aldaketak izan dira. Jarraian, egindako aldaketak zerrendatuko dira duten garrantziaren arabera:

- Garrantzi txikia
    - Hasierako robota beste berdin batengatik aldatu behar izatea, robotaren pila kargagailuak ez funtzionatzeagatik.
    - Robotarentzat okerreko *bluetooth* gailua erosi izana, eta honenbestez beste gailu bat erosi beharra.

- Garrantzi ertaina
    - Pila kargagailua erabiltzearen erosotasun ezagatik, pilak kargatzeko modua aldatu beharra. Horregatik, robotean bertan kargatu dira pilak robotean muntaketa batzuk eginda.

- Garrantzi handia
    - *Minicom,* serie komunikazioak burutzeko programa erabiltzeari utzi zitzaion, eta interfaze grafiko bat egitea erabaki zen.
    - *Windows*-en hasi arren proiektua, garapenean zehar bertan izandako zailtasunengatik *Ubuntu*-ra migratu beharra.
    - Garapeneko ordenagailu nagusiak funtzionatzeari uztea, eta honenbestez beste ordenagailu batera pasa behar izatea egindako guztia.

2.6 Arriskuak

Hurrengo atal honetan proiektuan zehar sor daitezkeen arrisku posibleak zerrendatu dira. Arriskuaren analisi egoki batentzako beraren eragina, inpaktua, probabilitatea eta konpontzeko soluzioa azalduko dira.

Hona hemen proiektuan zehar sor daitezkeen arrisku posibleak:

Arriskua	Dokumentazioa galtzea
Deskribapena	Proiektuan zehar bildu eta idatzi diren esteka, informazio eta testuak galtzea
Probabilitatea	Baxua
Eragina	Altua
Ondorioa	Dokumentazio lana berriro 0tik hasi behar izatea
Soluzioa	Dokumentazio lan guztiaren segurtasun kopiak hainbat lekutan izatea, hala nola *Dropbox* bezalako lekuetan edota ordenagailuan bertan.

Arriskua	Piezaren batek ez funtzionatzea
Deskribapena	Robotaren piezaren batek funtzionatzeari uztea
Probabilitatea	Ertaina
Eragina	Altua
Ondorioa	Robotarekin probak egiteko ezintasuna
Soluzioa	Piezaren zerbitzu teknikoarekin komunikatu, berau konpontzen saiatu edo beste pieza berri batengatik aldatu

Arriskua	Lengoaia ez menderatzea
Deskribapena	Erabiltzen ari zaren lengoaiaren bat guztiz ez menderatzea
Probabilitatea	Ertaina
Eragina	Altua
Ondorioa	Inplementazioan aurrera egiteko zailtasunak sortzen ditu
Soluzioa	Lengoaiari buruzko dokumentazio eta ikasketa lana egin

Arriskua	Pilak deskargatzea
Deskribapena	Robota erabiltzearen eraginez pilak deskargatzea
Probabilitatea	Altua
Eragina	Altua
Ondorioa	Robotarekin probak ezin egitea
Soluzioa	Kargaturiko pila batzuengatik aldatu

3. Kapitulua

3 Erabilitako teknologia eta azpiegitura

3.1 Hardwarea

Atal honetan proiektuan zehar erabili diren tresna eta teknologiak azalduko ditugu. Alde batetik hardware eta bestetik software mailakoak ditugu. Hardware aldetik, nahiz eta robota bera handia ez izan, tresna ugari erabili dira prozesu guztian. Hala eta guztiz ere, hardware aldetik tresnarik garrantzitsuenak *Pololu 3pi* robota eta berau programatzeko konexio kableak izan dira. Softwarearen aldetik, erabilitako liburutegi, driver eta abarren azalpena egingo dugu.

3.1.1 *Pololu 3pi* robota

Pololu 3pi robotaren izenaren jatorria bere tamainari dagokio, zeren eta bere diametroa cm-tan *pi* zenbakia 3 aldiz biderkatua baita. Errendimendu altuko robot bat da (ikus 3.1 irudia), eta bere funtzio garrantzitsuenak labirintoen ebazpenak dira. Hala ere, robotari beste erabilpen ezberdin bat emango zaio. Gure robota elikatzeko *AAA* motako 4 pila erabili ditugu. Gure robotak trakziorako sistema bakarra du motoreentzako eta motore hauen bitartez robota 100 cm/s –ko abiadurak hartzeko gai da. Gainera robota gai izango da bira eta zentzu aldaketa zehatzak egiteko bateriaren tentsioari eragin gabe. Robota programatu ahal izateko 3.1.2 atalean azaldutako *USB AVR Programmer* kanpoko *AVR ISP* programatzaile bat behar dugu. Robota honako elementuez hornitua dator eta 3.2 irudian ikus dezakegu elementu bakoitzaren lekua robotean.

- 2 motore
- *QTR* 5 sentsore infragorri
- *LCD* bat 2 lerrotan 8 karaktere bistaratzeko ahalmenarekin
- 3 pultsadore
- Burrunbagailu bat

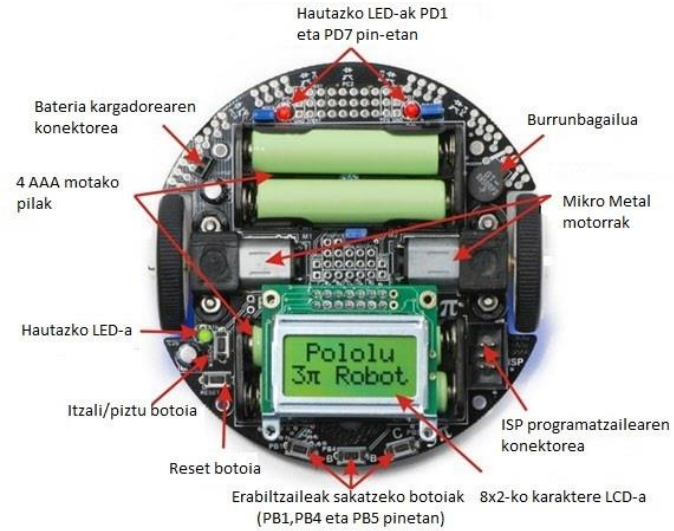
Robotak *ATmega328* mikrokontrolagailu bat du 20 MHz-tara doana. Beronek 32 k*B*-ko flash memoria eta 2 kB-eko *RAM* memoria ditu, horietaz gain 1 kB-ko *EEPROM*-a du mikrokontrolagailuak. *ATmega328*-aren erabilera bateragarria da *Arduino* bezalako garapenerako plataformarekin.

Robotari osagarri gehigarri batzuk gehitu behar izan zaizkio guk burutu nahi genuena lortzeko, horietako bat da *expansion kit* –a (ikus 3.1.3 atala).

Ondorengo zerrendan robotaren ezaugarri teknikoak ikus ditzakegu.

- **Abiadura maximoa:** 100 *cm/s*
- **Diametroa:** 95 *mm*
- **Pisua:** 83 g (bateria gabe)
- **Mikrokontrolagailua:** ATmega328
- **Motoreen driverra:** TB6612FNG
- **Tentsio Minimoa:** 3 Vcc
- **Tentsio Maximoa:** 7 Vcc
- **PWM frekuentzia maximoa:** 80 kHz

3.1 Irudia: Pololu 3pi robota



3.2 Irudia: Pololuren osagaien eskema

3.1.2 USB AVR programmer

Gailu hau AVR RISC motako mikrokontrolagailu familian oinarritzen diren mikrokontrolagailuen programatzailea da, kasu honetan *3pi Pololu* robotaren mikrokontrolagailuarena. Programatzaileak *STK500*[1] –a serie portu batean emulatzen du. USB eta *ISP* kablea ere badakar hurrengo irudian ikus dezakegun bezala.

[1] AVR flash-a martxan jartzeko garapen Kit-a

3.3 Irudia: USB *AVR programmer-a*

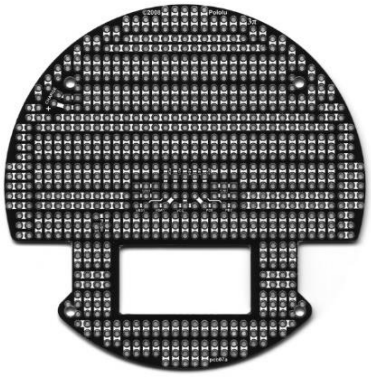
3.1.3 Expansion Kit-a

Expansion Kit hau robotaren osagarririk garrantzitsuenetakoa da, bertan kokatu baitira kanpoko gailu gehienak, hala nola, *BlueSMiRF Silver Bluetooth Modem* (ikus 3.1.4 atala).

Kit hau osatzen duten osagaiak honakoak dira, 3.4 irudian ikus daitezkeen moduan: Zirkuitu inprimatu biribildu bat (*PCB Printed Circuit Board,* 3.5 irudia), zulodun parrila bat, konektore ar bat, eta beste konektore eme bat 2x7 pin dituena, 2 konektore ar luzanga eta 2 konektore eme 2x1 pin dituena, plastikozko 4 bereizgailu 7/8 hazbetekoak, 4 torloju beren 1-1/4 hazbeteko azkoinekin.

3.4 Irudia: Kit-a osatzen duten osagaiak

Expansion kit-aren PCB-a eta *Pololu*-arenak antzekotasun ugari dituzte bai neurri eta koloreetan, biak beltzak baitira eta diametro berdina dute. *Kit*-a robotaren gainean muntatzen da lehen aipatu ditugun 4 torloju eta bereizgailuen laguntzaz 4.2 irudian ikus daitekeen moduan. *PCB*-a jada montatua dugunean hasi gaitezke gure interfaze elektronikoa prestatzen, hori burutzeko eztainuarekin soldaketa batzuk egingo dira bi zirkuituen artean.



3.5 Irudia: *Expansion Kit*-aren PCB-a

3.1.4 Bluetooth Modem - BlueSMiRF Silver

Bluetooth modem hau *expansion kit*-aren *PCB*-aren gainean jarri da robotaren eta ordenagailuaren arteko komunikazioa burutu ahal izateko. Modem honek serie komunikazioa (TX/RX) ahalbidetzen du. Datuak bidaltzeko orduan, kasu honetan ordenagailutik robotera izango da burutuko duen ibilbidea. Datu fluxuari dagokionez 9600 eta 115200 *bps*-ren arteko datuak ahal dira bidali. Gailu honek *RN-42* izeneko modulua dakar 3.7 irudian ikus daitekeen bezala, non azken finean berau izango den *bluetooth* seinaleak bidali/jasotzea ahalbidetuko duena.

RN-42 modulua egokia da bateriak erabiltzen dituzten gailuetarako (kasu honetan *Pololu* robota) eta distantzia motzetan komunikazioak burutu behar direnerako. Honek potentzia modu ezberdinak ditu, gure aplikazioaren arabera konfiguratu daitezkeenak. Horretaz gain tentsio erregulatzailea darama eta honenbestez 3.3 eta 6 VDC arteko elikadura iturria duten aplikazioetarako aproposa da.

Modem-aren atzeko aldean, 3.6 irudian ikus dezakegun bezala, 6 zulo ditu, baina 6 horietatik 4 bakarrik aprobetxatu dira. Batetik RX eta TX, serie komunikazio hori simulatu ahal izateko erabili ditugunak, bestetik *GND* (lur konexioa) eta *VCC* (zuzeneko korronteko tentsioa).

Modem hau robotera konektatu ondoren, eta robota piztean ordenagailuak segituan atzemango du *bluetooth* gailua.

3.6 Irudia: BlueSMIRF HID modulua-ren atzeko aldea

3.7 Irudia: BlueSMIRF HID modulua-ren aurreko aldea RN 42 moduluarekin

3.1.5 Wiimote

Wii Remote, laburtua *Wiimote* deitzen zaiolarik, *Wii* bideo kontsoletako periferiko nagusia dugu. *Wii U* bideokontsolan ere urrutiko aginte nagusi bezala erabili daiteke. Bere ezaugarririk esanguratsuena pantailan mugimenduak detektatzeko duen ahalmena da.

3.2 Softwarea

3.2.1 Sistema eragilea

Sistema eragilearen kasuan, bi izan dira erabili direnak, batetik *Ubuntu 13.10* eta *14.04* eta bestetik *Windows 8.1*. Aipatu behar da proiektuaren mamia *Ubuntun* garatu dela, baina hasierako urratsak *Windows*-en eman ziren, hasiera batean *Windows*-ek garatzeko ingurune erosoagoak eskaintzen zituelako.

Nire esperientzia ordenagailuen munduan handiagoa da *Windows*-en eta horregatik jo zen hara hasiera batean. *Pololu*-ren lehenbiziko probak bertan egin ziren, eta hasierako kode zatiak ere bai, baina puntu bat iritsi zen ezinezkoa zela *Windows*-en jarraitzea *Bluetooth modem*-a eta *Wii*-ko urruneko agintea konfiguratzean arazo ugari ematen baitzituen. Beraz migrazioa egin zen *Linux*-era, *Ubuntu 13.10*-era lehenbizi, *Ubuntu 14.04*-ra ondoren. Bertan oso gustura aritu naiz, nahiz eta hasieran pixka bat kostatu garatzeko ingurune berrietara ohitzea; azkenean neurria hartu zaio. Gainera *Bluetooth* eta *Wii*-ko urrutiko agintea konfiguratzerako orduan ez da arazorik izan eta nahiko erraza izan da zati hori.

3.2.2 Programazio lengoaiak

Garapenean zehar bi lengoaia oso komun erabili dira, *C* eta *C++*. Nahiz eta biak potentzia handikoak eta egokiak izan programazio mota honetarako, programen zati gehienak *C*-z garatu dira hainbat arrazoiengatik:

- Arrazoi nagusia izandako ezagutza maila da *C-ekiko*, karreran zehar gehien erabili izan den lengoaia baita eta berau erabiltzeko indar guztiekin sentitu naiz proiektuan zehar.
- Lengoaia efiziente bat dela ezin da ukatu, dituen behe mailako ezaugarriekin inplementazio ezin hobeak egiten baitira.
- Nahiz eta bere maila baxua izan, existitzen den lengoaia zabalduena da, non ia sistema guztientzako dagoen *C*-ren konpilatzailea.
- Programa modulatuak egiteko erraztasunak ematen ditu edota existitzen diren liburutegietako kodea erabiltzekoa.

Hala eta guztiz ere *C++* -ren erabilera nabaria izan da proiektuan zehar. Nahiz eta zuzenki askorik ez erabili, baliatu diren *Pololu*-ren liburutegietan eta *Qt Creator* interfaze grafikoa sortzeko softwarean automatikoki sortu diren kode zati batzuk *C++*-z zeuden.

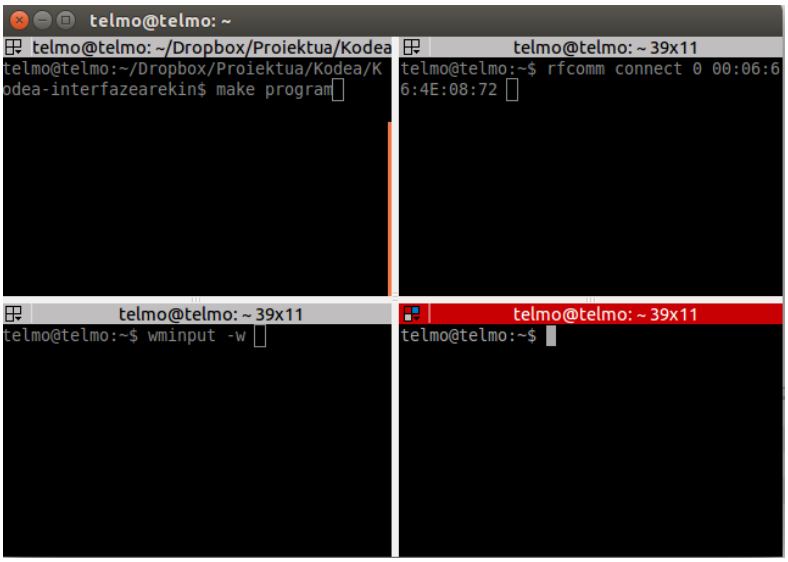
3.2.3 Komando interpretatzailea

Aipatu behar da interfaze grafikoa guztiz bukatua egon den arte, terminala erabili behar izan dela komando batzuk exekutatu ahal izateko. Behin interfaze grafikoa bukatuta, komandoak automatikoki A.1 irudian ikus daitekeen exekutagarriak jaurtikitzen zituen *system* funtzioaren bitartez.

Ubuntu-rekin lan egitean ezinbestekoa da *terminala*-rekin nolabaiteko trebetasuna hartzea lanaren zati bat komandoak exekutatzean datza eta. Fitxategiei baimenak eman, programen exekuzioa, programak konpilatu, *etab*.

Kasu honetan, hasiera batean *terminal*-a erabiltzen hasi zen ez baitzen beste tresnarik ezagutzen. Baina denbora aurrera joan ahala konturatu izan da nahiko deserosoa zela, hainbat *terminal* eduki behar baitziren aldi berean irekita. Arrazoi horiengatik tresna sofistikatuago bat erabili da *terminal*-aren lanak egin ahal izateko. Komando interpretatzaile berri honek *Terminator* du izena eta *Ubuntu*-ren biltegian topa dezakegu. *Terminator Python* programazio lengoaian garatua dagoen aplikazio bat da.

Tresna honek *terminal*-a hainbat leihotan banatzeko aukera ematen digu hainbat posiziotan (ikus 3.8 irudia) eta baita leiho hauek azpi leihoetan banatzekoa ere. Honen guztiagatik proiektuaren fase batzuetan ezinbesteko tresna bilakatu da ematen duen erosotasunagatik.



3.8 Irudia: Terminator aplikazioa martxan jartzeko komandoekin

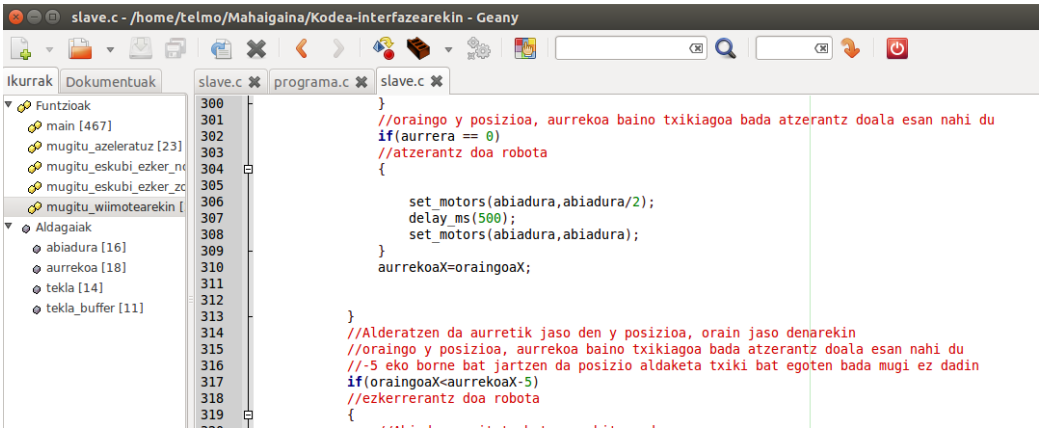
3.2.4 Garapen ingurunea

Zorionez *Ubuntu*-rekin bateragarriak diren hainbat garapen ingurune (*IDE*) existitzen dira. Aukera zabal hauen artean bi nabarmendu nahiko nituzke, batetik *Gedit* eta bestetik *Geany.* Hasiera batean *Gedit* testu editorearekin hasi zen lanean, eta bere ezaugarrien artean oso arina eta dinamikoa dela nabarmendu behar da. Baina bere alde txarra da ez duela *debugger*-ik eta horrek lan erritmoa makaltzen du kodean trazak ipintzen joan behar baitzara erroreak aurkitzeko. Bestetik, liburutegietako funtzioen izenak ez dizkizu gogorarazten, eta horrek ere lan erritmoa pixka bat makaldu dezake. Bere alde on bezala aipatu behar da *plugin*-ak gehitu daitezkeela eta honek aberastasuna ematen dio testu editoreari programa sinpleak egiterako orduan.

Bi arrazoi horiengatik pentsatu zen garapen ingurune berri bat bilatzea, eta hala *Geany*-rekin lanean hastea erabaki zen. *Geany, Gedit*-en moduan testu editore txiki eta arin bat da. Garapenen ingurune (*IDE)* batek eduki behar dituen oinarrizko ezaugarriak ditu eta *Scintilla*-n (iturri-kode edizioko osagai librea) oinarritua dago. Horretaz gain GTK liburutegiak erabiltzen ditu bere funtzionamendurako. Sistema eragile ezberdinetan dago eskuragarri, baina GNU/Linux –rako dagoen bertsioa erabili da. Bere ezaugarririk nabarmenenak honakoak dira:

- Kodearen auto osaketa
- Sinboloen zerrendak
- Pluginentzako euskarria
- Kodearen tolestea
- Aholkuak erakustea

Bi testu editoreen artean konparaketa bat egiterako orduan, ezberdintasun nagusia kodea idaztean *Geany*-k erakusten dituen erraztasunak dira. Bestela bi editoreak nahiko antzekoak dira, biak nahiko sinple eta arinak baitira.

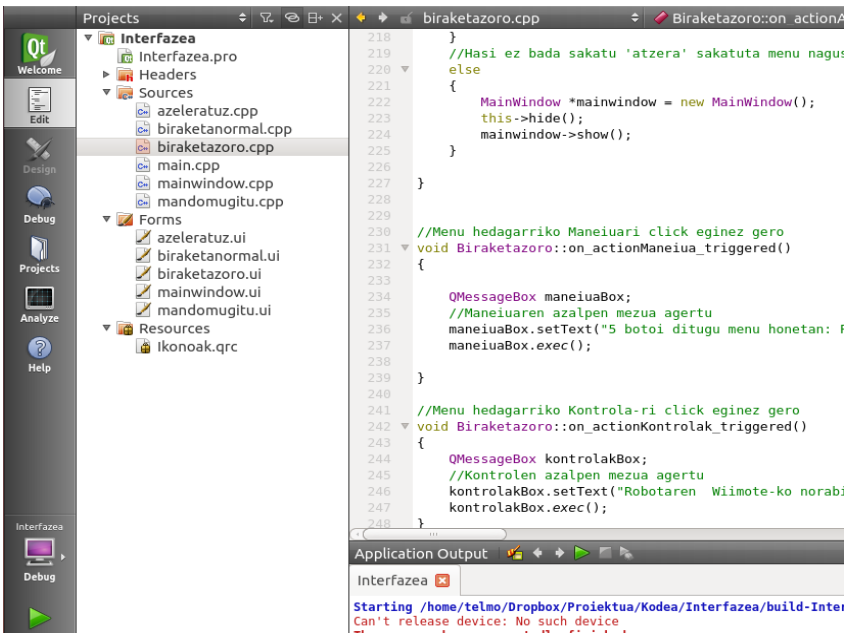


3.9 Irudia: *Geany*-n robotaren programa

Aurreko bi *IDE*-ak *Pololu*-n exekutatu behar den programa garatzeko erabili ditugu. Aldiz, interfaze grafikoa garatzeko *QT Creator* izeneko *IDE*-a erabili dugu. Trolltech-k sorturiko *IDE* hau *Qt* liburutegiekin garatu behar diren aplikazioak sortzeko erabiltzen da. Sistema eragile ezberdinetarako eskuragarri dago, hala nola, *Linux, Mac* eta *Windows.* Hala, hainbat sistema eta plataforma mugikorretarako aplikazioak sortzeko aukera ematen digu.

Linux-erako bertsioa erabili da, proiektu osoa bertan garatu baita. Interfaze grafiko bat garatu behar zela konturatu zen momentutik, berau egiteko software baten bila hasi zen. Baina *IDE* hau aukeratzeko prozesua ez da erraza izan, hainbat aukera ezberdin probatu baitira (*Anjuta, Glade...*) baina guzti hauek ez zuten asebetetzen, eta azkenenean *QT Creator*-en alde egin zen.

Indar handiko tresna bat da, edonolako interfaze grafikoak sortzea ahalbidetzen duena. Kasu honetan ez zen oso interfaze konplexua eta beti eman izan ditu erraztasunak nahi zen diseinua eta funtzioak egin ahal izateko. Hasieran kostatu egin zen moldatzea tresnara, baina eskuliburuei esker nahiko azkar hartu ahal izan zen trebetasuna.



3.10 Irudia: *QT Creator*-en interfaze grafikoaren kodea

3.2.5 Konpiladorea

Gure aplikazioa bi zatitan banatzen da, alde batetik *Pololu*-k exekutatu behar duen programa, eta bestetik interfaze grafikoa. *Pololu*-aren programa garatzeko lehen aipatu moduan *Geany* erabili dela esan dugu. Baina tresna honek badu desabantaila handi bat, *Pololu*-ren liburutegi bereziak erabili behar direnez *Geany* ez da gai garaturiko programa konpilatzeko. Honenbestez *Pololu*-ren liburutegiek ekartzen dizkiguten adibideen laguntzaz, beraiek prestatuak dituzten *makefile*-en bitartez konpilatu dugu egindako programa. *Makefile*-ak erabili ahal izateko *make* izeneko tresna beharrezkoa da. Aldiz interfaze grafikoa inplementatzeko erabili dugun softwareak, *QT Creator*-k badu bere konpiladore propioa.

- *make*

Askotan fitxategiek beraien artean dependentziak izaten dituzte, eta fitxategi guzti horiek batera automatikoki konpilatzeko garapen softwarea da *make.* Konpilazioa arintzeko asmoarekin, bere eginbehar nagusia berriro zein fitxategi konpilatu behar diren erabakitzea da.

- *makefile*

Lehenago aipatu den *make* softwareak *makefile* izeneko fitxategi batzuk erabiltzen ditu konpilazioaren kudeaketa egiteko. Adibide xume bat jartzekotan, *makefile*-ak pelikuletako gidoi bezala ikus ditzakegu, non *make*-k gidoi hori jarraitzen duen. Fitxategi hauek modu ordenatuan daude, eta erregela ezberdinek osatzen dituzte fitxategi hauek (ikusi 3.11 irudia).

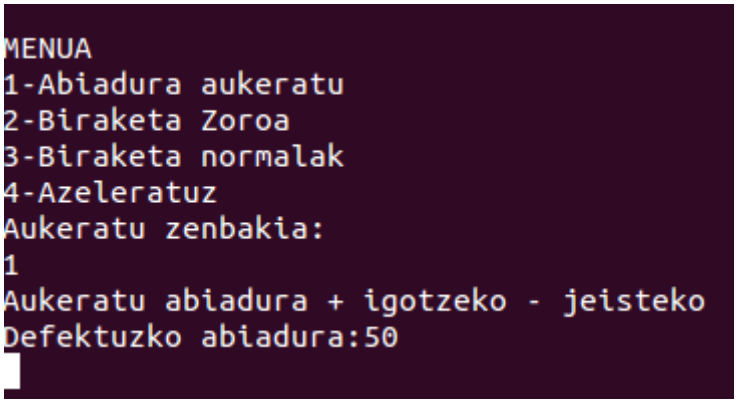
```
MCU = atmega328p
AVRDUDE_DEVICE = m328p
DEVICE ?= atmega168
MCU ?= atmega168
AVRDUDE_DEVICE ?= m168

CFLAGS=-g -Wall -mcall-prologues -mmcu=$(MCU) $(DEVICE_SPECIFIC_CFLAGS) -Os
CC=avr-gcc
OBJ2HEX=avr-objcopy
LDFLAGS=-Wl,-gc-sections -lpololu_$(DEVICE) -Wl,-relax

PORT ?= /dev/ttyACM0
AVRDUDE=avrdude

TARGET=slave
OBJECT_FILES=slave.o

all: $(TARGET).hex

clean:
    rm -f *.o *.hex *.obj *.hex
```

3.11 Irudia: *Makefile* fitxategia

3.2.6 Minicom

Windows sistemako *Hyperterminal* softwarearen baliokidea da *Linux* erabiltzen duten erabiltzaileentzako. Erabiltzaile hauek *RS232* serie portua kontrolatzeko aukera izango dute mikrokontrolagailu, *PIC, router* edota beste aplikazioren batean. Gure kasuan mikrokontrolagailuarekin erabili da, datuak serie moduan jaso zein bidaltzerako garaian. Bertan

menu moduko bat egin da non erabiltzaileak aukera ezberdinak dituen ikusgai 3.12 irudian ikus daitekeen moduan.



3.12 Irudia: *Minicom*-eko menua

4. Kapitulua

4 Garapena

Atal honetan proiektuaren garapena nola egin den azalduko da. Alde batetik robotaren osagai gehigarrien muntaketa nola egin den azalduko da, eta bestetik proiektuko inplementazioaren azalpena egingo da.

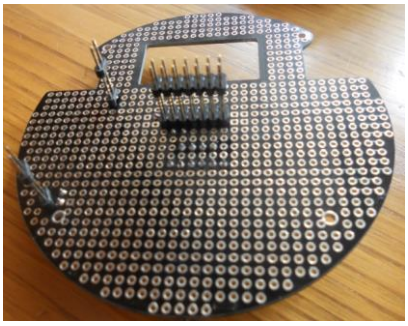
4.1 Osagaien muntaketa

Atal honetan robotari egin zaizkion muntaketa gehigarri horiek azalduko dira. Robotaren hasierako bertsiotik bukaerakora dezente aldatu da, proiektua aurrera zihoala gauza gehiago gehitu behar izan baitzaio robotari. Muntaketa 3 fasetan banatuko genuke, batetik *expansion kit*-a gehitzea, bestetik *Bluetooth* modem-aren txertaketa eta azkenik pila kargatzailea robotean gehitzea.

4.1.1 Expansion Kit-a

Expansion Kit-a roboteko ezinbesteko osagaia da, bertan txertatu baitira *bluetooth* gailua eta bateria kargadorea. Robotak eta *PCB*-ak izkinako 4 ertzetan 4 zulo dituzte, 3.1 eta 3.5 irudietan ikus daitekeen bezala.

Lehen pausoa, *kit*-ak ekartzen dituen 3 konektore emeak *PCB*-an eztainuz soldatzea izango da dagokien lekuan (ikus 4.1 irudia). Ondoren 4 torlojuak robotaren eta *PCB*-aren 4 zuloetatik sartzen dira, eta azkoinekin lotzen dira. Torlojuek plastikozko bereizgailu bat izango dute torlojua babesteko eta politagoa geratzeko. *Expansion kit*-a montatua 4.2 irudian ikus dezakegu nola geratuko litzatekeen.



4.1 Irudia: *PCB*-aren atzeko aldea konektoreekin

4.2 Irudia: *Expansion Kit*-a dena montaturik

4.1.2 *Bluetooth* modem-aren txertaketa

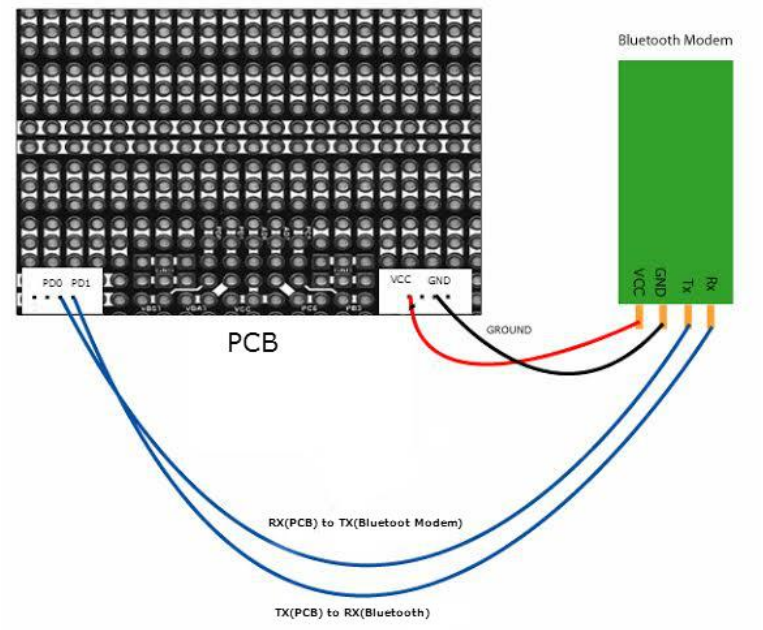
Bluetooth modem-a beste osagai garrantzitsu bat izan da proiektuan zehar *Pololu-Wiimote* arteko komunikazioa burutu ahal izateko. Hasiera batean gailua eskuartean zegoela, ezjakintasuna jabetu zen ez baitzegoen oso argi *PCB*-ko zein lekutan txertatu behar zen eta nola. Baina irakasle eta interneteko informazioari esker bidea argitu zen eta azkenean lortu zen gailua *PCB*-an txertatzea (ikus 4.5 irudia).

Gailuak 6 zulo ditu (*CTS, VCC, GND, TX, RX* eta *RTS*) eta zulo horietan hankatxo ilara bat eztainuztatu zen 4.3 irudian ikus daitekeen bezala.

4.3 Irudia: Gailua hankatxo ilararekin konektatuta

Behin hori eginda, 6 zuloko eme bat eztainuztatu da *PCB*-aren gainean eta bertan kokatuko da lehenago moldatu dugun *bluetooth* gailua.

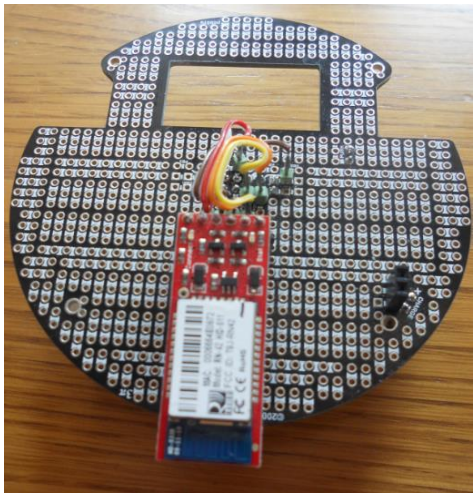
Hurrengo pausoa, *PCB*-aren gainean kableen bidez konexioak sortzea da modema eta PC-aren arteko komunikazioa burutu ahal izateko. *PCB*-an portu ezberdinak ditugu *AD7,AD6,VBAT.*.baina *PD0,VCC,PD1* eta *GND* erabiliko dira beheko irudian ikus dezakegun moduan. Serie komunikazioa ahalbidetuko duten portuak *PD0(RX)* eta *PD1(TX)* dira. *PD0*-k datuak jasotzeko lana egingo du, aldiz *PD1*-k datuak bidaltzearen lana beteko du. 4.4 irudian ikus daiteke *PCB*-a eta *bluetooth* gailuaren arteko konexioak:



4.4 Irudia: *PCB* eta *Bluetooth* gailuaren arteko konexioak

Bluetooth gailuan baita ere 4 portu erabiliko dira: *GND* (lurra)*, VCC* (elikatze tentsioa)*, RX* eta *TX*. Serie komunikaziorako *RX* eta *TX* portuak erabiliko dira, lehena datuak jasotzeko eta datuak bidaltzeko bestea.

4.4 argazkian ikus daiteke alde batetik *bluetooth* gailua dagoela, eta bestetik *PCB*-a. Gailutik 4 kable irteten dira, eta kable horiek *PCB*-an eztainuztatu dira irudian ikusten den ordenean.



4.5 Irudia: *BlueSMIRF* modulua *PCB*-an txertatua, bere konexioekin batera

4.1.3 Pila kargadorea robotean

Hasiera batean, pilak kanpoko kargadore batean kargatzen ziren, baina nahiko deserosoa zen pilak gastatzen ziren bakoitzean robota desmuntatzen ibili beharra. Horregatik robotean bertan pila kargadore bat muntatzea pentsatu zen. Horretarako 3D inprimagailu batekin pieza berezi bat egin zen kargadorearen konektorearen euskarri moduan erabiltzeko.

Ondoren, *PCB*-an konektore ar bat txertatu zen *charge* jartzen duen lekuan, eta bertara euskarritik irteten den kable bi konektatzen dira. Horrela kargadorea euskarrira konektatzen da eta pilak robotean daudela kargatu egiten dira.

4.2 Inplementazioa

Hasiera batean aurreikusia zegoen interfazerik ez egitea lehenago aipatu den moduan, baina aurrera egin ahala planak aldatu ziren, honenbestez inplementazioa ere.

Proiektuaren inplementazioak bi zati nabarmen ditu, alde batetik robotean exekutatuko den programa eta bestetik erabiltzaileak robotaren funtzionalitateak aukeratzeko erabiliko duen interfaze grafikoa.

Bi zatiak independenteak dira inplementatzerako orduan, baina beraien artean lotura bat dago. Dena martxan jartzen denean bi zatiak komunikatu egiten dira, eta komunikazio hori *rfcomm0* portuaren bitartez lortzen da. *RFCOMM* protokoloa garraiorako erabiltzen diren protokoloen multzoa da. Honek, aldi bereko 60 konexio sortzen ditu, hala *RS-232* serie portua emulatuz. Beraz *rfcomm0* portuak, serie portu baten lana burutzen du.

4.2.1 Funtzio nagusiak

- `serial_receive_blocking(char * buffer, unsigned char size)`

Funtzio honek *rfcomm0* portutik interfazeak bidaltzen dituen datuak jasotzen ditu *buffer* aldagaian. *buffer* aldagaiak izango duen luzera ere zehaztu behar da funtzio honi dei egitean. Funtzio honek interfaze grafikotik eta *Wiimote*-tik bidaltzen diren datuak jasoko ditu.

- `void set_motors(int m1Speed, int m2Speed)`

Lehenago aipatu den moduan, robotak bi motor ditu eta motor bakoitzak gurpil bakoitzaren abiadura kontrolatzen du. Prozedura honekin bi motor horiei ipini nahi zaion abiadura ezartzen zaio. Abiaduraren balioa -255 eta +255 artean egongo da, zeinuak motorraren norabidea adieraziko duelarik.

Robota aurrerantz joatea nahi bada `m1Speed` eta `m2Speed` –k balio berdina izan beharko dute. Aldiz atzerantz joatea nahi bada, bi aldagaien balioak berdinak izan behar dute baina zentzu negatiboan (`-m1Speed, -m2Speed`). Robotak norabide aldaketaren bat egitea nahi bada adibidez ezkerrera, ezker gurpilaren abiadurak eskuin gurpilarena baina txikiagoa izan beharko du, hau da `m1Speed < m2Speed`. Robotak eskuinerantz bira egitea nahi bada aldiz, `m1Speed > m2Speed` izan beharko da.

- `print(const char *str)`

Robotaren *LCD* pantailan karakterez osaturiko string-ak idazteko erabiltzen da.

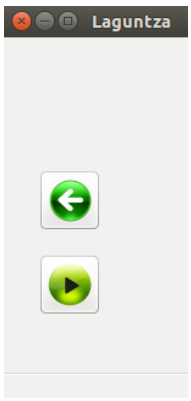
4.2.2 Robotaren programa

Robotaren programa 5 bloke ezberdinetan banatzen da. Batetik programaren *main*-a dugu, eta ondoren 4 prozedura. 4 prozedura hauek robotak izango dituen 4 funtzionalitateekin datoz bat, prozedura bakoitzak robotaren funtzionalitate bakoitza inplementatuko duelarik. Prozedura guztiek egitura gehiena berdina dute, bi begiztek osatzen dute:

```
 while(tekla!='h' && tekla!='a')
 {
while(tekla != 's' && tekla!='a')
{
```

4.6 Irudia: Prozeduretako bi begiztak

Begizta hauek interfaze grafikoa eta robotaren arteko kontrol lana egiten dute. Lehenbiziko begiztan interfazean *Play* edo *Atzera* (ikus 4.7 irudia) botoiari sakatu bitartean zain egongo da.



4.7 Irudia: *Atzera* eta *Play* botoiak

Play botoiari sakatzerako kasuan 2. begizta nagusian sartuko da. Aldiz, *Atzera* sakatuz gero prozeduratik irten eta *main* funtzioa exekutatuko da.

Bigarren begiztan prozeduretako mamia dago, bertan erabakitzen da robotak hartuko dituen norabideak eta abiadurak. Begizta honetako irteera baldintzak, interfaze grafikoan *Stop* edota *Atzera* botoia sakatzea izango da.

Funtzionalitate guztiek beste kode zati bat dute komunean, hau da interfaze grafikoan *Stop* eta *Pause* botoiak sakatzerakoan. Momentu horietan robota geratu egingo da:

```
    //Robota geratu egingo da
    case 'p':
       set_motors(0,0);
       break;
    //Robota geratu egingo da
    case 's':
       set_motors(0,0);
       break;
```

4.8 Irudia: *Pause* eta *Stop* botoiak sakatzean kodean gertatzen dena

Aldagai globalak

Programa honetan garrantzi handia izango dute aldagai globalek, funtzio ezberdinetan aldagai berdinak erabili behar baititugu askotan. Erabili behar ditugun aldagai horiek `buffer`, `tekla` eta `abiadura` dira.

Aldagairik garrantzitsuena `buffer` izango da, bertan jasoko baitira *rfcomm0* portutik jasotzen diren datuak `serial_receive_blocking` funtzioa erabiltzen dugunean. Interfaze

grafikotik teklaren bat bidaltzen bada, `serial_receive_blocking` funtzioaren bitartez jaso eta `tekla` aldagaian gordeko da.

Programa honetan robotak dituen bi motorrei abiadura bat ezarri behar zaienez beharrezkoa izango zaigu `abiadura` aldagaia.

4.2.2.1 Programa nagusia

Programako funtziorik garrantzitsuena dugu honako hau. Funtzio hau, inplementatu diren 4 prozedurak antolatuko dituena da eta horretaz gain interfaze grafikoarekin lehenengo kontaktua izango duena. Proiektu honetan, interfaze eta robotaren arteko komunikazioa *rfcomm0* portuaren bitartez burutu da.

Honenbestez *main*-ean, `serial_set_baud_rate(x)` funtzioa erabiliko da gure serie komunikazioaren modulazio abiadura ezartzeko. Kasu honetan, *Pololu*-ren liburutegiak funtzio honen erabileran, `x`=115200 baudeko balio estandarra izatea gomendatzen digu.

Behin modulazio abiadura ezarrita, begizta batean sartuko gara. Begizta hau erabiltzaileak interfazean klikatzen duenaren menpe egongo da. Erabiltzaileak programa ixtea erabakitzen badu begiztatik atera eta aplikazioa amaituko da, robotaren *lcd* pantailan berau itzaltzeko mezua agertzen den artean. Aldiz erabiltzaileak robotaren funtzionalitateren bat probatzea erabakitzen badu aukeraren batean klikatuz, 4.9 irudian agertzen den bezala klikaturiko funtzionalitatearen prozedurara exekutatuko da.

```
switch(tekla)
{
    case '1':
        mugitu_eskubi_ezker_zoro();
        break;
```

4.9 Irudia: Aukeraturiko funtzionalitatera salto

Interfaze eta robotaren arteko komunikazioa posible izateko, *Pololu*-ren liburutegiak eskaintzen duen `serial_receive_blocking()` funtzioa erabili da. Kasu honetan, *buffer*-aren luzera 1 izango da interfazetik zenbaki bat bidaltzen baitu. Hala, `switch case` egitura baten bitartez jasotzen den zenbakiaren arabera funtzio bat edo bestea exekutatuko da robotean.

4.2.2.2 Urrutiko agintearekin

Robotaren funtzionalitaterik interesgarriena dugu honakoa, *Wiimote*-ren norabide botoirik erabili gabe kontrolatzeko aukera eskaintzen duena.

Funtzionalitate hau inplementatuko duen prozedurak `mugitu_wiimotearekin()`du izena. Inplementatzerako orduan prozedurarik konplexuena izan da dudarik gabe, faktore asko hartu behar izan baitira kontuan. Prozedura honen funtsa PC-tik bidaltzen diren kurtsorearen x eta y posizioak jaso eta horren arabera gauza bat edo bestea egitean datza. Prozedura honen mamia 2. begiztan dago eta hainbat fase bereizi ditzakegu.

1. Begizta etengabe PC-ko kurtsorearen *x* eta *y* posizioak jasotzen egongo da `serial_receive_blocking` funtzioaren bitartez .

2. Begiztan jasotzen diren *x* eta *y* posizioak, aurreko pasaldian jaso diren x eta y balioekin alderatzen dira jakiteko robota aurrera/atzera edo ezker/eskuin-erantz doan. Balioak (*x* eta *y*) aurreko pasalditik aldatzen badira abiadura unitate batean areagotu egingo da.
3. X eta Y posizioak jasotzen dira `posx` eta `posy` aldagaietan, eta ondoren balio horiek `oraingoaX` eta `oraingoaY` ri esleituko zaizkie.
4. Posizio balio horiek aurreko pasaldian jasotzen direnekin konparatuko dira, honatx robota aurrerantz joaten denaren adibidea :

```
if(oraingoaY>aurrekoaY)
//aurrerantz doa robota
{
```

4.10 Irudia: Uneko eta aurreko posizioen alderaketa

5. Ezker edo eskuin norabide aldaketak egiterako orduan arazo txiki bat egon zen. *Wiimote*-a zertxobait mugituz gero ezker edo eskuinerantz `oraingoaX>aurrekoaX` edota `oraingoaX<aurrekoaX` baldintzak betetzen ziren. Horrek robota eskuin edo ezkerrerantz mugitzea zekarren, hau da oso sentsiblea zen mugimenduetara. Hori konpontzeko 5 unitateko borne bat ipintzea erabaki, eta hala funtzionamendu egokiago bat lortu da:

```
if(oraingoaX<aurrekoaX-5)
//ezkerrerantz doa robota
{
 if(oraingoaX>aurrekoaX+5)
 //eskuinerantz doa robota
 {
```

4.11 Irudia: Bost unitateko bornea baldintzetan

6. Robota martxan dagoela robota momentu baterako edo denbora mugagabe baterako geratzeko ‘p’ edo ‘s’ teklak jasoko dira interfazetik.

4.2.2.3 Robota biraketa zoroak eginaz mugitu

Robotaren beste funtzionalitatea dugu honakoa, begientzat dibertigarria izan daitekelarik bere mugimendu zoroekin. Robota *Wiimote*-ko gora/behera/ezker/eskuin botoiekin kontrolatuko dugu.

Funtzionalitate hau inplementatuko duen prozedurak `mugitu_eskubi_ezker_zoro()` du izena. 4.2.2 atalean azaldu den moduan prozeduren egitura bi begiztek osatzen dute. Lehenbiziko begiztan, erabiltzaileak interfazean aukeratzen duen robotaren abiadura jasoko da eta `abiadura` aldagaian gordeko da.

Bigarren begizta *Wiimote*-an eta interfazean sakatzen diren botoien datuak jasotzen egongo da. Ondoren, 4.12 irudian ikus daitekeen `switch case` egituraren bitartez alderatuz joango da jaso diren datuak, eta horren arabera ekintza bat edo bestea gauzatuko da.

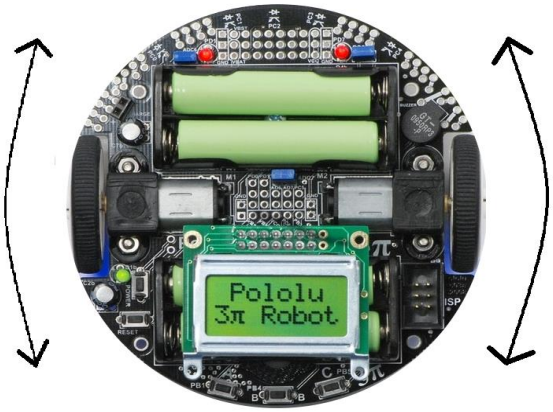
```
switch(tekla)
{
   //Aurrerantz doa robota
   case 'u':
```

4.12 Irudia: *Swicth case* egitura bigarren begiztan

Jasoriko datua “u” bada, robota aurrerantz joango da, aldiz “d” jasotzean robotak atzera egingo du. Beste hainbeste “l” (ezkerrerantz) eta “r” (eskuinerantz) jasotzean.

Robota martxan jarri, eta ibili dadin lehenago aipatu den `set_motors()` (ikus 4.2.1 atala) prozedura erabiliko da.

Funtzionalitate honen berezitasuna eskuinerantz eta ezkerrerantz biratzean ikusten da. *Wiimote*-ko ezker edo eskuin botoiari sakatuz gero, aukeraturiko norabidean etengabe biraka hasiko da. Horretarako `set_motors()` prozedura erabiliko dugu, eta 4.2.1 atalean azaltzen den bezala biraketa horiek egiteko balio jakin batzuk izan beharko dituzte aldagaiek prozedurari dei egitean. Robotak ezkerrerantz biratzea nahi bada `set_motors(0,abiadura)` deia egin beharko da, aldiz eskuinerantz biratzea nahi badugu `set_motors(abiadura,0)` deia egingo da. Hurrengo irudia robotaren funtzionamendua definitzen saiatzen da bere biraketa marrazkiekin.



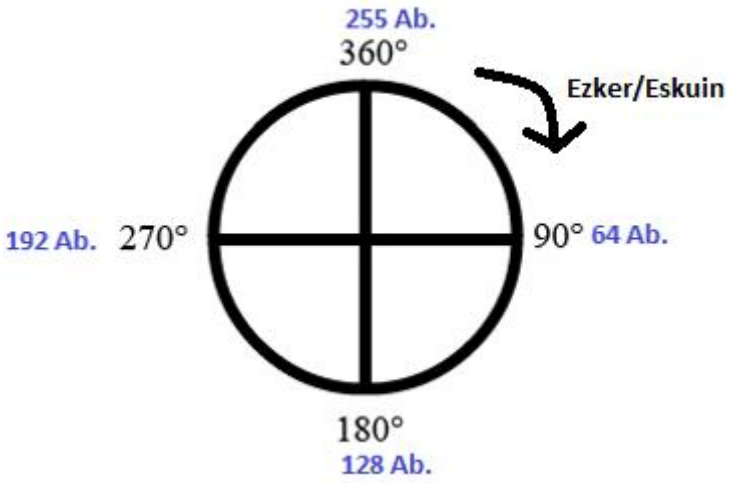
4.13 Irudia: Funtzionalitatea definitzen duen irudia

4.2.2.4 Robota 90 graduko biraketa eginaz mugitu

Robotaren hurrengo funtzionalitatea aurrekoaren (§4.2.2.3) nahiko antzekoa da, bestea bezala *Wiimote*-ko gora/behera/ezker/eskuin botoiekin kontrolatuko dugularik. Funtzionalitate hau `mugitu_eskubi_ezker_normal()` izeneko prozedurak inplementatuko du.

Diferentzia bakarra biraketa egiteko moduari dagokio. Kasu honetan ezker/eskuin biraketak egitean 90 graduko biraketa bat egingo dute aukeraturiko norabidean. Biraketa horiek egiteko besteetan bezala `set_motors()` prozedura erabili da, baina deia egitean `m1Speed` eta `m2Speed`-ri balio ezberdinak ematen zaizkio.

4.14 irudian ikus dezakegu biraketa hauen azalpena, non ezker/eskuin-erantz 90º-ko biraketa egiteko `m1Speed` edo `m2Speed`-ri 64 balioa eman behar zaion eta besteari 0 ezker edo eskuin biratu behar dugunaren arabera.



4.14 Irudia: 90 graduko biraketen irudikapen grafikoa

4.2.2.5 Robotaren abiadura azeleratuz mugitu

Robotaren funtzionalitate hau aurrekoekin alderatuz dezente ezberdina da. Komunean duten puntu bakarra *Wiimote*-arekin kontrolatuko dela baita ere funtzionalitate hau. Funtzionalitate hau `mugitu_azeleratuz()` izeneko prozedurak inplementatuko du. Segurtasun neurriengatik robotak kasu honetan, abiadura finkatzeko erabiltzen dugun balio maximoa 100 izango du.

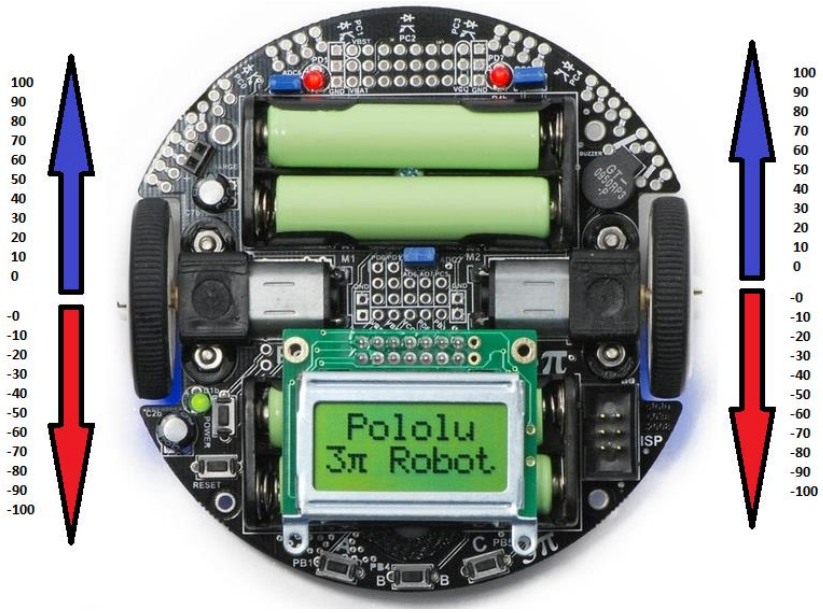
Honek duen berezitasuna abiadurarekin du zerikusia. Robota aurrera eta atzera dabilenean erabiltzaileak aukera izango du bere abiadura handitu edo txikitzeko *Wiimote*-ko aurrera/atzera botoiekin. Biraketak egiterako orduan ere badu berezitasuna, biraketak abiadura areagotuz egiten baitira. Horretaz gain aurrera/atzera/eskuin/ezker botoiak sakatzen diren aldiro abiadura aldagaia unitate batean areagotuko da.

Inplementatzerako garaian, adibidez ezkerraldera biratu nahi bada, eskuin gurpileko motorraren abiadura ezker gurpilaren motorrarenaren bikoitza izango da; modu berdinean eskuinerantz biratu nahi bada. Behin biraketa eginda, modu errealago baten antza hartu dezan, robotak aurrerantz/atzerantz egingo du. Hau izango litzateke biraketak inplementatzeko modua:

```
case '1':
   //abiadura gehitzen
   abiadura=abiadura+1;
   set_motors(abiadura/2,abiadura);
   delay_ms(500);
   set_motors(abiadura,abiadura);
```

4.15 Irudia: Ezkerraldera biratzeko inplementazioa

Erabiltzaileak robotaren abiadura jaisteko aukera ere izango du, eta horrela `abiadura` aldagaia unitate batean dekrementatuko da erabiltzaileak *Wiimote*-ko balaztari sakatzen dion bakoitzean.



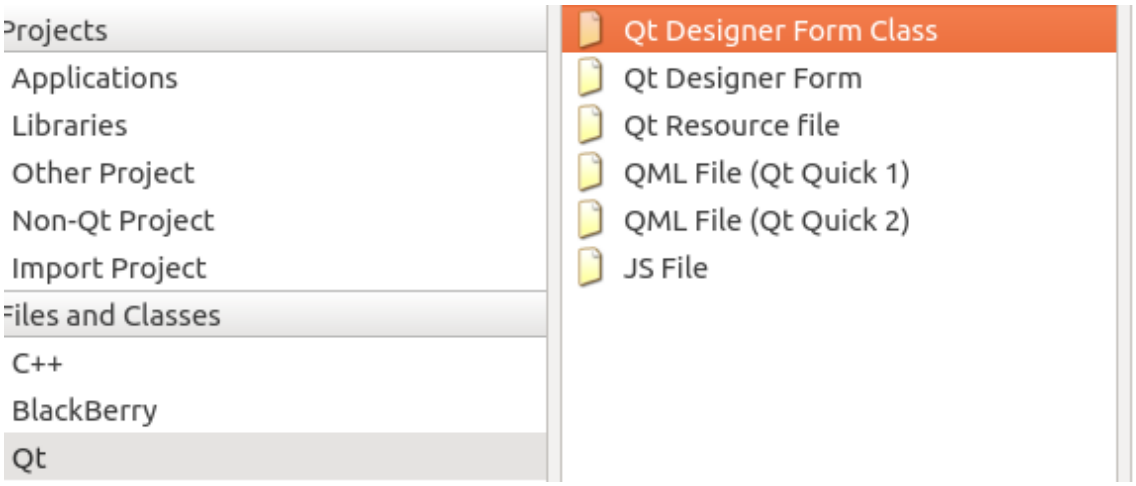
4.16 Irudia: Modalitatearen irudikapen grafikoa

4.2.3 Interfaze grafikoaren inplementazioa

Inplementazioaren bigarren zati hau erabiltzailearentzat interfaze grafiko bat sortzean datza. Interfazea egin ahal izateko *Qt* liburutegia erabili da eta bertako funtzio eta klase ezberdinak erabili dira. Interfaze grafikoaren inplementazioa 3.2.4 atalean azaldu den bezala *QT Creator* softwarearekin garatu da. Interfaze grafikoak robotarentzat inplementatu den programarekin lotura zuzena du.

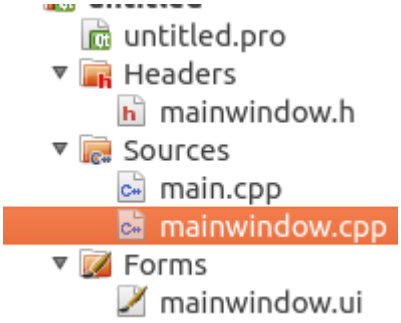
Proiektua sortzerakoan fitxategi gutxi batzuk sortzen dira automatikoki 4.18 irudian ikus daitekeen moduan. Sortzen diren *mainwindow.cpp* eta *mainwindow.h* fitxategiak *mainwindow.ui* (interfazearen leiho nagusia) –ren iturburu eta iturburu-kode fitxategiak dira.

Behin proiektua sortuta, hurrengo pausoa *Qt Designer Form Class* (*.ui, .h* eta *.cpp* fitxategiak) –ak sortzea litzateke. Funtzionalitate bakoitzeko hauetako bat sortuko dugu 4.17 eta 4.19 irudian ikus daitekeen moduan.



4.17 Irudia: *Qt Designer Form Class* sortzen

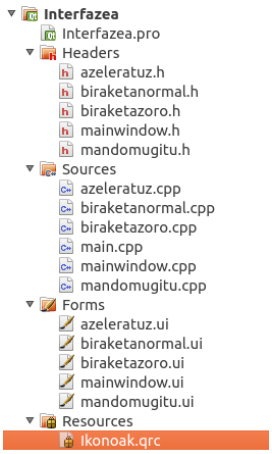
Softwarean modu ezberdinetan antolatzen dira fitxategiak 4.19 irudian ikus daitekeen moduan. Fitxategien antolaketa modu hau inplementatzerako orduan oso lagungarria da momentu oro baitakizu behar den fitxategia non dagoen.



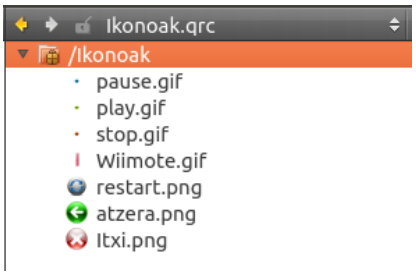
4.18 Irudia: Proiektu sortu berria

Jarraian azalduko da antolaketa honen egitura zertan datzan:

- **Headers fitxategiak:** Sortu diren *.cpp* fitxategien iturburu-kodeak egongo dira.
- **Sources fitxategiak:** *.ui* fitxategien iturburu fitxategiak.
- **Resources fitxategiak:** *.qrc* fitxategiak biltzen dira hemen, eta 4.20 irudian ikus daitekeen moduan *.qrc* fitxategiak interfazean erabiltzen diren ikonoak biltzeko erabil daitezke.
- **Forms fitxategiak:** Interfazearen leihoen *.ui* fitxategiak dira. *XML*-z idatzita daude, baina leihoen osagaiak automatikoki sortzen dira 4.21 irudian ikus daitekeen osagaien laguntzaz.



4.19 Irudia: *QT Creator*-eko fitxategi mota ezberdinak



4.20 Irudia: *Ikonoak.qrc*-en interfazean erabili diren ikonoak

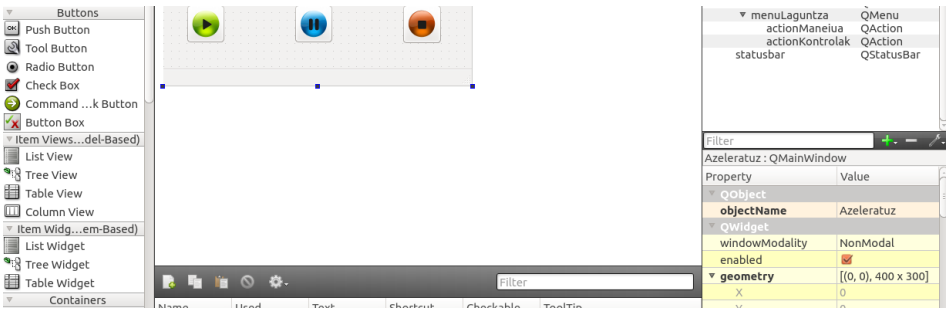
Behin klase eta fitxategi guztiak sortuta daudela inplementazioari ekingo zaio. Inplementazioa bi fasetan banatu da:

- **Diseinua:** Sortu diren leihoak diseinatu egin dira nahi den gustura.
- **Programazio lana:** Leihoetako botoiak sakatzean gertatzen dena inplementatu da *.cpp* eta *.h* fitxategietan.

4.2.3.1 Diseinua

Leihoak diseinatzerako orduan modu sinple batean egin da, garrantzitsuena ez baitzen interfazearen diseinua, baizik eta robota ongi kontrolatzea *Wiimote*-arekin. Hala eta guztiz ere interfaze txukun bat egin da, beharrezko botoiak jarriaz.

Horretarako *QT Creator*-ek eskaintzen dituen osagaiak erabili dira 4.21 irudian ikus daitekeen moduan. Bertan ikus daiteke mota ezberdinetako botoiak gehi daitezkeela eta gero beraien propietateak editatu.



4.21 Irudia: Diseinatzeko osagaiak

4.2.3.2 Programazio lana

Zati honetan egongo da interfazearen mamia. 4.22 irudian interfazearen itxura ikus daiteke eta botoien egitura. Funtzionalitate ezberdin bakoitzaren leihoetan gertatzen dena inplementatu da zati honetan. Funtzionalitate bakoitzak bere klasea izango du *.cpp* fitxategi bat izango dena.

4.22 Irudia: Interfazearen itxura eta botoien egitura

4.2.3.2.1 Komunak diren kode zatiak

Leiho bakoitzak bere berezitasunak dituen arren badaude gauza batzuk modu berean inplementatuta daudenak:

- **Datuak *rfcomm* portuaren bitartez bidaltzea**

Prozesu hori egiteko funtzio berdinak erabiltzen dira daturen bat robotari bidali nahi zaion bakoitzean. Prozesua 4.23 irudian agertzen den bezala 3 pausotan betetzen da:

1. *rfcomm* portua ireki *open* funtzioaren bitartez.
2. *fputc* funtzioaren bitartez nahi den datua/karakterea bidaliko da ireki berri den portutik.
3. Irekiriko portua itxiko dugu *fclose* funtzioaren bitartez.

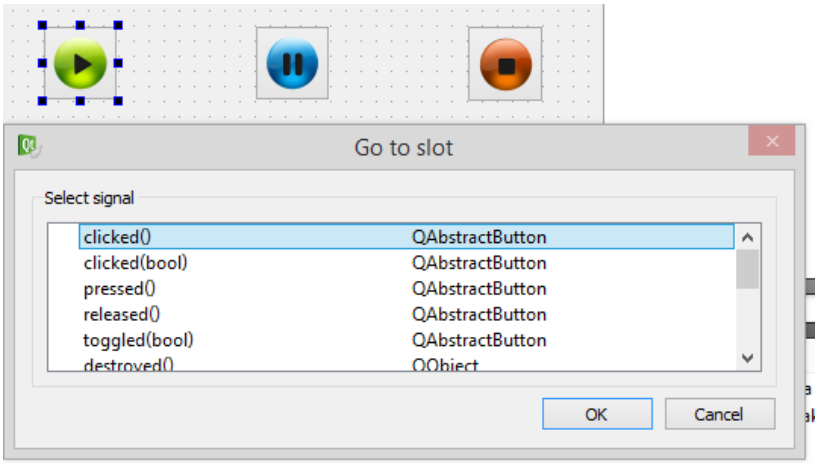
```
//RFCOMM0 portua ireki
FILE * portua = fopen ("/dev/rfcomm0", "w");
//Sakaturiko botoiaren 'zenbakia' aldagaia bi
fputc(zenbakia,portua);
//RFCOMM0 portua ixten da
fclose(portua);
```

4.23 Irudia: *rfcomm* portutik datuak bidaltzen

- **Botoien ekintzak (*Play, Pause, Stop, Atzera* eta *Berrabiarazi)***

Botoi bat sakatu eta zerbait burutzea nahi bada, botoi horri ekintza bat programatu behar zaio, botoia sakatzean guk nahi duguna egin dezan. Horretarako 4.24 irudian agertzen den bezala *Go*

To slot tresna erabiliko dugu. Bertan hainbat ekintza daude aukeratzeko, baina kasu honetan *clicked()* ekintza erabiliko da, botoia sakatzean zerbait gertatzea nahi baita.



4.24 Irudia: Ekintza sortzailea

clicked() ekintza aukeratzean, dagokion *.cpp* fitxategian 4.25 irudian agertzen den moduko kodea izango dugu. Aukeratu dugun botoia sakatzen dugunean, bertan programaturikoa gertatuko da.

```
//Pause botoia sakatzen denean sortzen den ekintza
void Biraketanormal::on_pausaButton_clicked()
{
    //RFCOMM0 portua ireki
    FILE * portua = fopen ("/dev/rfcomm0", "w");
    tekla='p';
    //'p'(pause) tekla bidaltzen zaio robotaren programari RFCOMM0 portutik
    fputc(tekla,portua);
```

4.25 Irudia: *Pause* botoiak sortzen duen ekintza

Kasu honetan 4.25 irudian ikusten den moduan *Pause* botoia sakatuz gero, robotaren programari komunikatzen zaio *fputc()* funtzioaren bitartez robota geratu egin behar dela. Funtzionalitateen leihoan dauden *Play, Pause, Stop* eta *Berrabiarazi* botoien ekintzen kasuan kodea berdina izango da 4 funtzionalitateetan.

- **Teklen ekintzak**

Wiimote-ko botoiek teklatuko teklak emulatzen dituzte B.6 atalean azaltzen den moduan da. Hau da *Wiimote*-ko adibidez *B* botoia sakatzen bada ordenagailuak ulertuko du konfigurazioan *B* botoiari esleitua dagoen tekla sakatu dela. Robota *Wiimote*-ko botoiekin kontrolatzen denez, aldiro ekintzak sortzen egongo da interfazearen programa.

Hau da, teklaren bat sakatzen denean programatuta dagoen ekintza bat burutuko da. *Wiimote*-aren erabilpena teklatuarekin zuzenki erlazionatua dagoenez B.6 atalean azaltzen den moduan, horregatik teklen ekintzak ezinbestekoak dira *Wiimote*-arekin robota kontrolatu ahal izateko.

Proiektuan zehar *Wiimote*-ko 5 botoi erabiliko direnez, 5 ekintza ezberdin sortuko dira funtzionalitate bakoitzeko *.cpp* fitxategian. Botoien ekintzetan ez bezala, ekintza hauek eskuz sortu behar dira, eta ekintza honen izena `keyPressEvent` izango da 4.26 irudian ikus daitekeen moduan. Ekintza hau modu berdinean egongo da programatua 3 funtzionalitateetan (*Biraketa Zoroa, Mugitu azeleratuz* eta *Biraketa normal).* Kasu hauetan 'U', 'D', 'L', 'R' eta 'B'

teklen ekintzak izango dira. Aldiz *Wiimote-a mugituz* funtzionalitatea ezberdin programatu da 4.2.3.2.5 atalean azaltzen den moduan.

```
//Teklatuko ekintzak
void Biraketazoro::keyPressEvent( QKeyEvent * event )
{
    //rfcomm0 portua ireki
    FILE * portua = fopen ("/dev/rfcomm0", "w");
    //idatzitako teklaren kodea jasotzen da
    const int botoi = event->key() ;
        //Tekla bakoitzaren kodearekin konparatzen da, eta koinziditzen badute, letra hori
        switch (botoi)
        {
            //U letraren kodea
            case Qt::Key_U :
                //'u'(gora) tekla bidaltzen zaio robotaren programari RFCOMM0 portutik
                tekla='u';
                fputc(tekla, portua);
                break;
            //D letraren kodea
            case Qt::Key_D :
                //'d'(behera) tekla bidaltzen zaio robotaren programari RFCOMM0 portutik
                tekla='d';
                fputc(tekla, portua);
                break;
```

4.26 Irudia: Tekla ezberdinen ekintzak

4.2.3.2.2 *MainWindow* klasea

MainWindow klasea interfazeko klase nagusia da, eta bertatik 2 gauza aipatu behar dira beren rengatik.

- ***system* funtzioaren erabilpena:** Exekutagarria martxan jartzen denean 4.27 irudian ikusten den moduan *system* funtzioaren bitartez *linux*-eko bi agindu exekutatzen dira:
    1. Robotean dagoen *bluetooth* gailua ordenagailuarekin konektatzen da, *rfcomm connect* komandoa erabiliz.
    2. *Wiimote*-a teklatuarekin konektatzen da *wminput –w* komandoaren bitartez.

```
system("rfcomm release 0");
//Pololu-PC-ren arteko bluetooth konexioa ahalbidetzen da
system("rfcomm connect 0 00:06:66:4E:08:72 &");
//Wiimote-ak arratoia emulatuko du
system("wminput -w &");
```

4.27 Irudia: Komandoen exekuzioa *system* funtzioarekin

- **Menu nagusian aukeraketa:** Atal honetan menu nagusiko botoien ekintzen azalpena egingo da. Kasu honetan, A.3 irudian ikus daitekeen moduan 5 botoi ditugu, 4 funtzionalitateei dagokionez eta beste bat aplikazioa ixteko. Funtzionalitateei dagokien botoien ekintzak antzeko modu batean inplementatu dira.
    1. `zenbakia` aldagaiari sakatu berri den botoiaren zenbakia esleitzen zaio. Kasu honetan 4.28 irudian ikus daitekeen moduan *Biraketa Zoroa* funtzionalitateari dagokion ‘2’ zenbakia esleitzen zaio aldagaiari.
    2. *fputc()* funtzioaren bitartez bidaliko zaio `zenbakia` aldagaia robotaren programako *main()* –ri eta horren arabera funtzio bat edo bestea exekutatuko da robotean 4.9 irudian agertzen den moduan.

```
void MainWindow::on_biraketaZoroaButton_clicked()
{
    //robotaren programari bidaliko zaion aukeraketa aldagari '2' zenbakia esleitzen zaio
    zenbakia = '2';
    //RFCOMM0 portua ireki
    FILE * portua = fopen ("/dev/rfcomm0", "w");
    //Sakaturiko botoiaren 'zenbakia' aldagaia bidaltzen zaio robotaren programari RFCOMM0 portutik
    fputc(zenbakia,portua);
```

4.28 Irudia: `zenbakia` aldagaia bidaltzen da

Funtzionalitateen botoien bati sakatzen bazaio, beste leiho berri bat azalduko da.

4.2.3.2.3 *Azeleratuz* klasea

Klase honetan 4.2.2.5 atalean azaltzen den bezala robotaren abiadura azeleratuz joango da. Honenbestez bi gauza nabarmendu behar dira kodetik:

- ***Wiimote*-ko botoiren bat sakatzen bada:** Aztertuko dugun kasua, *Wiimote*-ko goranzko norabideari buruz izango da 4.29 irudian ikus daitekeen bezala. Kasu honetan robotari komunikatuko zaio *aurrera* norabidea sakatu dela, eta honenbestez robotaren programan abiadura unitate batean handituko da. Irudian ikus daitekeen bezala, baita ere abiadura unitate batean handitu da, A.6 irudian ikus daitekeen bezala abiadura erakusten baita.
- **Abiadura muga 100ean:** A.3.2 atalean azaltzen den bezala abiadura muga bat du funtzionalitate honek, honenbestez klase honetan muga hori kontrolatu behar dugu. Horretarako 4.30 irudian agertzen den moduan *if* baten bitartez kontrolatuko dugu abiadura. Abiadura muga pasatzen bada abisu mezu bat agertuko da interfazean.

```
//U letraren kodea
case Qt::Key_U :
    tekla='u';
    //'u'(gora) tekla bidaltzen zaio robotaren programari RFCOMM0 portutik
    fputc(tekla, portua);
    //Lcd-an agertuko den abiadura markagailua inkrementatu
    abiaduraAz++;
    //Inkrementatu ahala, abiadura erakutsi
    ui->lcdNumber->display(abiaduraAz);
    break;
```

4.29 Irudia: *Wiimote*-an aurrera botoia sakatu da

```
if(abiaduraAz < 100)
{
//Sakatu den teklaren arabera gauza bat edo bestea egingo da

else
{
    //Mezua agertuko da balaztatzeko esanaz
    abiaduraMaxAbisua.warning(this,"Azkarregi!","Abiadura muga gainditu duzu,balazta ezazu");
}
```

4.30 Irudia: Abiadura mugaren kontrola

4.2.3.2.4 *Biraketazoro* eta *Biraketanormal* klaseak

Bi klase hauek nahiko antzekoak dira, bere funtzionalitateek beraien artean antzekotasun asko baitituzte A.3.1 eta A.3.4 ataletan azaltzen den bezala. Bi gauza dira nabarmendu beharrekoak:

- **Abiadura aldatzailea:** Funtzionalitate hauetan robota martxan jarri aurretik abiadura aukeratu behar da A.5 irudian ikus daitekeen moduan. Horretarako leihoen diseinu tresnetako *SpinBox* objektua txertatu zaie, horren bitartez abiadura balio ezberdinak aukeratu daitezke. Objektu horri ekintza bat sortu behar zaio, abiadura balioa aldatzen den bakoitzean zerbait gerta dadin, eta ekintza hori `valueChanged()` izango da. Bertan abiadura balioa aldatzen den bakoitzean `abiadura` aldagaia eguneratu egingo da 4.31 irudian ikusten den moduan.
- ***Wiimote*-ko botoiren bat sakatzen bada:** Kasu honetan, *Wiimote*-ko botoiren bat sakatuz gero robotari komunikatuko zaio zein botoi sakatu den *fputc()* funtzioaren bitartez.

```
//Abiadura aukeratzailean, balioren bat aldatzean
void Biraketazoro::on_abiaduraAldatzailea_valueChanged(int arg1)
{
    //Abiaduraren balio berria esleitzen zaio aldagaiari
    abiadura=arg1;
    //Abiadura balio aldatu dela adierazteko
    abiadaldatuBai = true;
```

4.31 Irudia: Abiadura eguneratzen da

4.2.3.2.5 ***Mandomugitu* klasea**

Honakoa da, klaseetan bereziena beste funtzionalitaterenekin alderatzen badugu, batez ere teklatuko ekintzei dagokionean. *Wiimote*-a mugituz robota mugitzen den funtzionalitate honetan, gauzarik aipagarriena da kurtsorearen posizioa nola lortu eta bidaltzen den.

- **Kurtsorearen posizioa lortu eta bidali:**
    1. Robotaren mugimendua 4.2.2.2 atalean aipatzen den moduan kurtsorearen posizioaren menpekoa da. Horregatik kurtsorearen momentuko posizioa lortzen da `pos()` funtzioarekin. Funtzio horrek kurtsorearen *x* eta *y* posizioa itzultzen ditu 4.32 irudian agertzen den moduan.
    2. Kasu honetan, pantailak 1366x768 –ko bereizmena du, hau da *x* eta *y* –ren balio posibleak. Baina balio hauek handiegiak direnez robotari bidaltzeko, 13.65 eta 7.67 balioengatik zatituko dira hurrenez hurren. Honela robotari 0 eta 100 bitarteko *x* eta *y*-ren balioak bidaliko zaizkio.
    3. Behin *x* eta *y* –ren balioak lortu direla *fputc()* funtzioaren bitartez robotari bidaliko zaizkio.

```
//posiz objektuan pos() funtzioak itzultzen dituen pantailako
posiz = QCursor::pos();
//Pantailak har dezakeen X eta Y posiziorik altuena 1365 eta
//Horrela  posx eta posy-k izango duten baliorik handiena 100
//posx aldagaiean posiz objektuko x posizioa esleituko da
posx = posiz.x()/13.65;
//posy aldagaiean posiz objektuko y posizioa esleituko da
posy = posiz.y()/7.67;
```

4.32 Irudia: Kurtsorearen posizioa lortzen

4.2.4 Probak

Proiektuan zehar inplementazioa eta probak lotuta egon dira, zerbait berria inplementatzen zen bakoitzean probatu egiten zen robota martxan jartzen. Egia esan probak nahiko ongi atera dira proiektuan zehar ez baita zailtasun nabarmenik egon.

Dezente kostatu den zatia *Wiimote*-a mugituz funtzionalitatea izan da. Bere konplexutasuna jakinda, ez da erraza izan robotak portaera egoki bat izatea. Zailena biraketen kasua izan da, robota oso sentikorra zen biraketetara nahiz eta *Wiimote*-arekin robota aurreranzko norabidean izan eskuin edo ezkerretara biratzen zuen batzuetan. Hori konpontzeko kodean aldaketa batzuk egin ziren borne batzuk jarrita 4.2.2.2 atalean azaltzen den bezala.

Probetan agertu den beste arazo bat abiadura handiko norabide aldaketa bortitzena izan da. Robotak, aurkitu gabeko arrazoiren batengatik ez zituen ongi hartzen abiadura handiko norabide aldaketak. Imajina dezagun robota aurrerantz doala aukeratu den 150-eko abiadura balioan eta bat-batean atzerantz botoiari emanez gero seguraski robota itzali egingo zen. Arazo hori konpondu ahal izateko abiadura balioei muga batzuk jarri zaizkio, honela robotek ezingo dute 100 –eko abiadura balioa gainditu.

5. Kapitulua

5 Ondorioak

5.1 Ondorioak

Proiektuan esan daiteke hasieran ezarri ziren helburuak bete direla, proiektuaren helburu nagusia *Pololu 3pi* robota *Wiimote*-arekin kontrolatzea baitzen. Egindako lanarekin pozik nago, hasieran finkaturiko helburuak ongi bete direlako, hasieran pentsatu baino zailtasun gutxiagorekin gainera.

Hala ere onartu beharra dago momentu zailak ere egon direla eta batzuetan pentsatu izan da finkatu ziren helburu guztiak ez zirela beteko, adibidez *Wiimote*-a mugituz robota martxan jartzea. Baina pazientzia ez galtzea, bakoitzaren buruarengan sinestea eta irakasleen laguntza ezinbestekoak izan dira helburuak lortu ahal izateko.

Proiektuak hainbat gauza ikasteko aukera eman du, baita hainbat gauza berri barneratzeko, programa berrietatik hasi eta programatze teknika berrietaraino. Ikasitako gauzen artean azpimarratzekoa izan da interfaze grafiko dotoreak egiten ikastea, eta *bluetooth* gailuekin egin daitezkeen gauzak ikustea.

Esperientzia aberasgarri honetan konturatzen zara robotikaren arloa zeinen aberasgarria izan daitekeen, eta teknologikoaren munduan nolako garrantzia duen robotikaren arloak. Nahiz eta proiektua ez izan horren probetxugarria mundu profesionalean, aisialdirako entretenigarria izan daitekela uste dut.

Maila pertsonalari dagokionez uste dut proiektu hau ikasketa aukera on bat izan dela eskuragarri izan ditudan azpiegitura eta materialak ikusten badira. Gainera proiektu honek motibazio gisara balio izan dit, datorren urtean egingo ditudan ikasketen inguruan eta etorkizunean landu nahi dudan esparrua zehazten lagundu baitit.

5.2 Etorkizunerako lan lerroak

Landu den proiektua robotikako arloan sailka dezakegunez, beti geratuko da zerbait egiteko, beti egongo da zerbait hobetzeko. Hurrengo lerroetan etorkizunean gara daitezkeen ideia batzuk azalduko dira.

Interesgarriena, *android* plataformarentzako aplikazioa sortzea litzateke. Gaur egun oso modan daude *android* gailuak, telefono mugikorrak edo tabletak adibidez, eta oso praktikoa litzateke bertarako *app* bat egitea. Ukimen bitartez kontrolatzeak praktikotasun eta erosotasuna emango lioke erabiltzaileari. Aldaketa nabaria izango litzateke *app* garatu nahi duenarentzat

inplementatzerako orduan, interfazea *QT Creator*-ekin egin baita. Aldiz *android app*-a egiteko *Eclipse* edo antzeko softwareren bat erabili beharko litzateke, eta posible da zailtasunen batekin aurkitzea.

Beste hobekuntza bat, interfazea zertxobait politagoa eta dinamikoagoa egitea litzateke. Egindakoa ongi dago, baina agian sinpleegia geratzen da eta etorkizunerako ongi legoke beste ukitu bat ematea interfaze grafikoari.

Aipatu behar da, *Wimote-a mugituz* funtzionalitatea pixka bat motz geratu dela, inplementatu den moduak jartzen zituen murriztapenengatik. Hau egiteko *QT* liburutegi bat erabili da, eta agian *Wii*-ren liburutegiren bat erabiliz gero azelerometroak erabiliz funtzionalitate hau hobetu daiteke.

6. Kapitulua

6 Bibliografia

Arduino Proiektuei buruzko informazioa

http://www.proyectosarduino.com/category/proyectos/

Pololu 3pi-ri buruzko informazioa

http://all-robotics.com/www/productos/robotica/86-pololu-3pi

http://www.electronicaembajadores.com/Productos/Detalle/-1/ROBT001/robot-seguidor-de-lineas-pololu-3pi

http://www.bricogeek.com/shop/robotica/106-seguidor-de-lineas-pololu-3pi.html

Bluetooth Modem-ari buruzko informazioa

http://www.pololu.com/product/2725

http://www.330ohms.com/Bluetooth-Mate-Silver_p_101.html

C lengoaiari buruzko informazioa Wikipedian

http://es.wikipedia.org/wiki/C_(lenguaje_de_programación)#Ventajas

***Make*-ri buruzko informazioa Wikipedian**

http://es.wikipedia.org/wiki/Make

Bluetooth protokoloei buruzko informazioa Wikipedian

http://es.wikipedia.org/wiki/Protocolos_Bluetooth

Baud Rate-ri buruzko informazioa Wikipedian

http://es.wikipedia.org/wiki/Baud_rate

AVRDUDE-ren instalazioari buruzko informazioa

http://naxus.jimdo.com/documentacion/programador-nxprog/avrdude

WIIUSE-ri buruzko informazioa

http://www.macs.hw.ac.uk/~ruth/year4VEs/Labs/wiiuse.html

WIIC-ri buruzko informazioa

http://wiic.sourceforge.net/index.php

VCC eta GND-ri buruzko informazioa

http://www.pololu.com/docs/0J26/3

Serie komunikazioari buruzko informazioa

http://www.pololu.com/docs/0J23

Arduino bat kontrolatu Wiiko urrutiko aginte batekin

http://www.grupoera.com.co/index.php?option=com_content&view=article&id=62:mando-a-distancia-con-wiimote-a-arduino&catid=25:proyectos&Itemid=2

RN42 Bluetooth module-ri buruzko informazioa

https://www.sparkfun.com/products/10253

Wminput-ri buruzko informazioa

http://abstrakraft.org/cwiid/wiki/wminput

PD0 eta PD1 portuei buruzko informazioa

http://www.pololu.com/docs/0J21/10.a

Bluetooth Konexioari buruzko informazioa

http://wannabe-programmer.blogspot.com.es/2013/05/comunicacion-bluetooth-arduino-pc-linux.html

http://playground.arduino.cc/Learning/Tutorial01

http://playground.arduino.cc/Learning/ArduinoBT-Ubuntu

7 Eranskinak

A. Eranskina: Gidaliburua

B. Eranskina: Instalazioa

C. Eranskina: Bilera Aktak

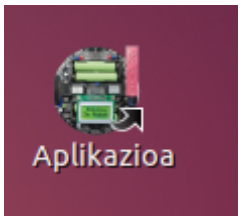
A. Eranskina

A Gidaliburua

Jarraian, aplikazioaren gidaliburu bat dago, erabiltzaileak aplikazioa nola erabili ikusi ahal izateko.

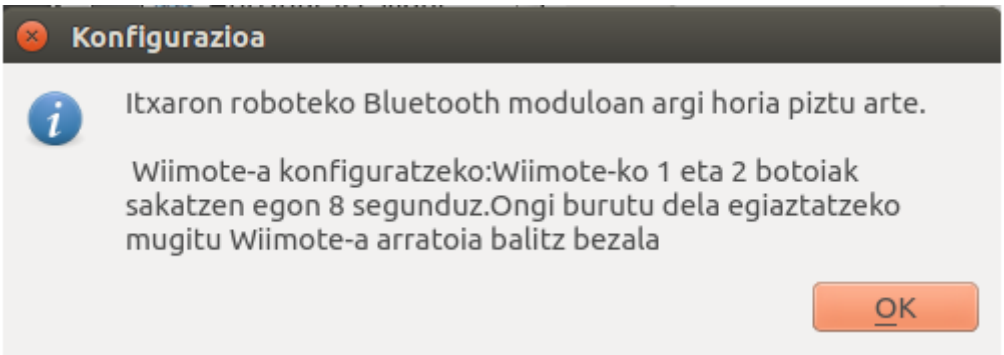
A.1 Aplikazioa abiarazi

Lehendabizi robota piztu beharko dugu. Robota martxan dugula aplikazioa martxan jartzeko exekutagarri (ikus A.1 irudia) bat izango dugu PC-an, eta nahikoa izango da berau exekutatzea dena martxan jartzeko. Exekutagarri honi ematean interfaze grafikoa azalduko zaigu.



A.1 Irudia: Aplikazioaren exekutagarria

Programa martxan jartzean mezu hau agertuko da, aplikazioaren hasierako konfigurazioak ongi egin direla egiaztatzeko arau batzuk jarraituaz.

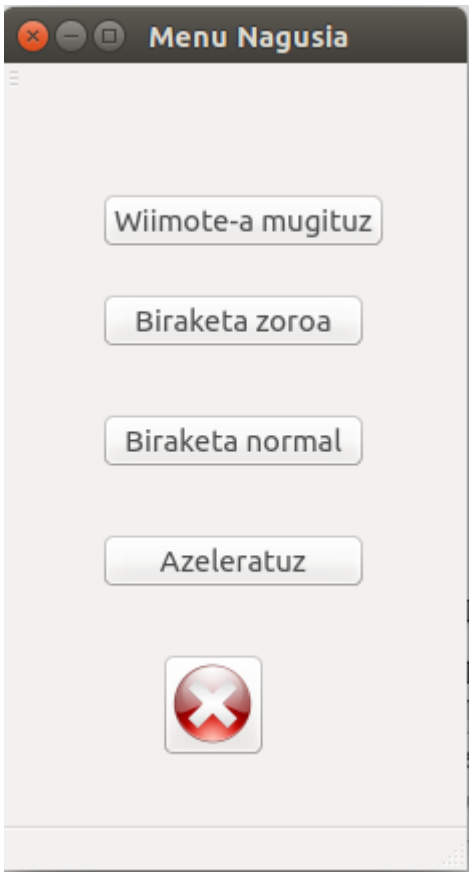


A.2 Irudia: Konfigurazio mezua

Bi gauza egin beharko ditugu A.2 irudian agertzen moduan:

- Lehendabizi itxaron egin beharko dugu roboteko *bluetooth* moduluan argi horia piztu arte. Pizten denean, PC-*Bluetooth* modem arteko konexioa burutu dela esan nahiko du.
- Azkenik *Wiimote*-a ordenagailuarekin konektatu beharko dugu, eta horretarako *Wiimote*-ko '1' eta '2' botoiak sakatzen egon beharko 8 segundoz.

Hurrengo irudian dezakegu aplikazioaren menu nagusia:



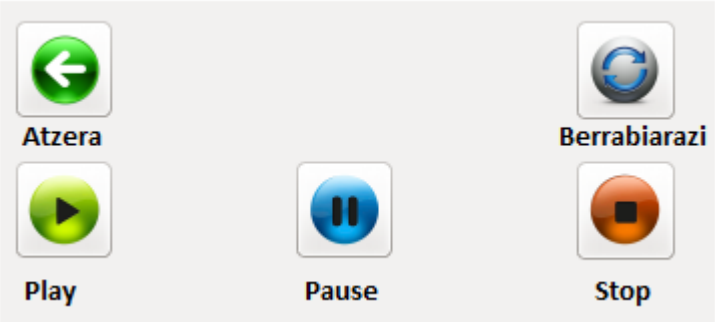
A.3 Irudia: Aplikazioaren menu nagusia

Bertan 4 aukerak ikusi ditzakegu eta programa ixteko botoia. Hau sakatuta programa guztiz geratuko litzateke. Nahi duzun aukeraren gainean klikatuta aukeraren leihoa irekiko da.

A.2 Menuetako botoien azalpena

Funtzionalitate guztietan botoi batzuk komunak izango dira (ikus A.4 irudia), eta berdin funtzionatuko dute guztiek. Robota martxan hasteko, lehendabizi *Play* botoiari eman behar zaio, eman ezean robota ezingo da mugitu.

Momenturen batean robota geratzea nahi bada denbora baterako, *Pause* botoiari emateko aukera egongo da. Robota berriro martxan jarri nahi bada ez da arazorik egongo. *Stop* botoia aldiz robota guztiz geratzea nahi badugu sakatuko dugu. Botoi honi sakatu ondoren ezingo dugu robota berriro mugiarazi. Eta azkenik robota berriro martxan jartzea nahi bada *Berrabiarazi* botoiari eman beharko zaio.



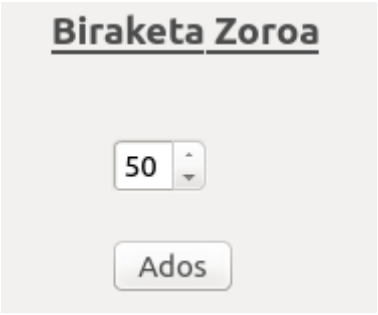
A.4 Irudia: Funtzionalitateen menuetako botoiak

A.3 Funtzionalitate ezberdinak

Jarraian aplikazioaren funtzionalitate ezberdinak erabiltzeko azalpena egingo da.

A.3.1 Biraketa Zoroa

Funtzionalitate honen berezitasuna, biraketa zoroak egiten dituela da. Menuan komunak diren *Play, Pause, Stop* eta *Berrabiarazi* botoiak izango ditugu. Menu honen berezitasuna abiadura aukeratzea posible izango duzula da A.5 irudian ikusten den moduan. Bertan 10tik hasita 100 arteko abiadura bat aukeratu beharko da, eta *Ados* botoiari sakatu. Ez badugu abiadurarik aukeratzen *Play* botoia klikatzean abisu mezu bat agertuko zaigu abiadura aukeratzeko esanaz.



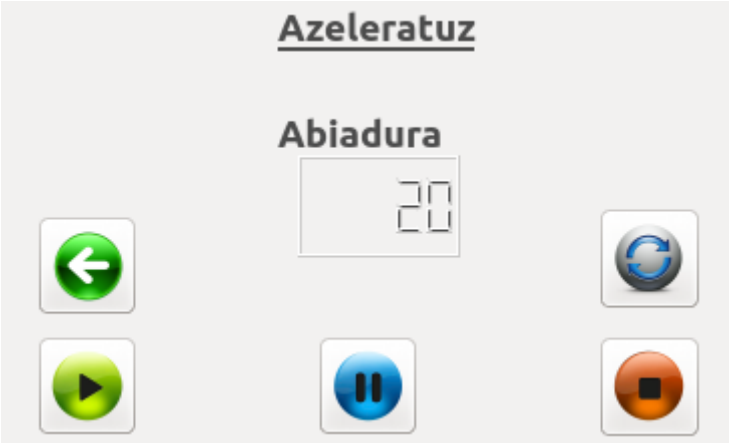
A.5 Irudia: Abiadura aukeraketa

Behin abiadura aukeratuta, *Wiimote*-ko norabide botoiekin(aurrera/atzera/ezker/eskuin) kontrolatu ahal izango da robota. Biraketak egiterako orduan, ezker edo eskuin botoiei sakatuz gero etengabe hasiko da biraka.

A.3.2 Mugitu azeleratuz

Funtzionalitate hau besteak bezala *Wiimote*-ko norabide botoiekin kontrolatzen da. Bere berezitasuna botoi hauek erabiltzean robotaren abiadura azeleratu egiten dela da. Besteetan ez bezala hemen ez dago lehenago abiadura aukeratu beharrik, eta menuan momentuko abiaduraren adierazle bat egongo da (ikus A.6 irudia).

Abiadura areagotzen goazen heinean muga batera iristen bada, kasu honetan 100, abisu mezu bat agertuko da abiadura jaitsi behar dugula ohartaraziz.



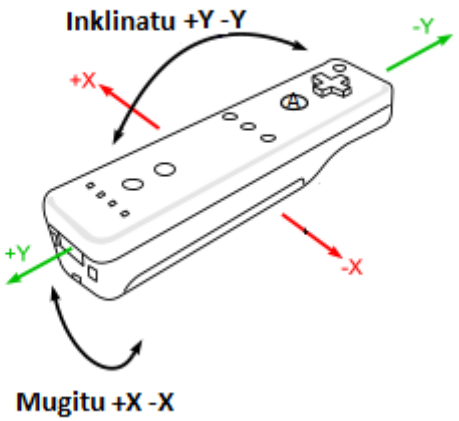
A.6 Irudia: Abiadura adierazlea

A.3.3 Wiimote-a mugituz

Hau da seguraski aplikazioaren funtzionalitaterik politena. Robota *Wiimote*-ko norabide botoiei sakatu gabe mugitzeko aukera eskaintzen baitu. Funtzionalitate honen menua A.9 irudian ikus dezakegu, eta beste funtzionalitateen menuaren antzekoa dela nabari daiteke.

Robota aurrera/atzera/eskuin/ezkerrerantz ibili dadin, *Wiimote*-ko 'A' botoia sakatzen egon beharko da, berau sakatu ezean robota ez baita mugituko. X eta Y ardatzak ditugu A.7 irudian ikus daitekeen moduan, eta robota ardatz horien arabera mugituko da.

Robota aurrerantz edo atzerantz mugitu nahi bada, 'A' botoia sakatzen dugun bitartean *Wiimote*-a gorantz(aurrera) eta beherantz(atzera) mugitu beharko da. Kontuan hartu behar da mugitzen ari garen heinean abiadura areagotzen joango dela, bai aurrera bai atzeranzko kasuetan. Robotaren abiadura gutxiagotu nahi bada, *Wiimote*-ren atzean dagoen 'B' botoia (ikus A.8 irudia) sakatu beharko da. Robota eskuin edo ezkerrerantz mugitu nahi bada, 'A' botoia sakatzen dugun bitartean ezker edo eskuinerantz mugitu beharko dugu *Wiimote*-a.



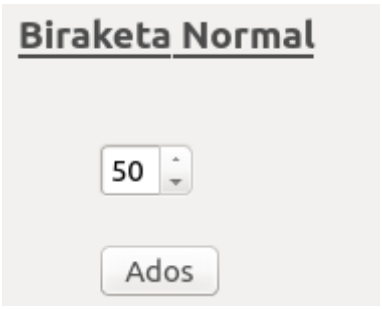
A.7 Irudia: *Wiimote*-aren mugimendu eskema

A.8 Irudia: *Wiimote*-aren B botoia

A.9 Irudia: Funtzionalitatearen menua

A.3.4 Biraketa normal

Funtzionalitate honen berezitasuna atzerago aipatu den moduan, 90 graduko biraketak egiten dituela da. Menuan komunak diren *Play, Pause, Stop* eta *Berrabiarazi* botoiak izango ditugu. Menu honen berezitasuna da abiadura aukeratzea posible izango dela A.10 argazkian ikusten den moduan, *Biraketa zoroa* (ikus A.3.1 atala) aukeran bezala. Bertan 10tik hasita 100-ra arteko abiadura bat aukeratu beharko da, eta *Ados* botoiari sakatu. Ez bada abiadurarik aukeratzen *Play* botoia klikatzean abisu mezu bat agertuko da abiadura aukeratzeko esanaz.



A.10 Irudia: Abiadura aukeraketa

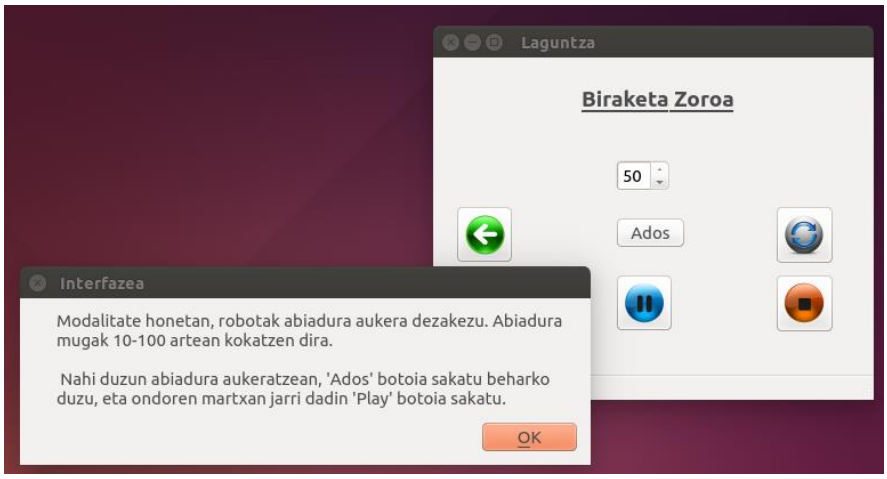
Behin abiadura aukeratu dela, *Wiimote*-ko norabide botoiekin(aurrera/atzera/ezker/eskuin) kontrolatu ahal izango da robota. Biraketak egiterako orduan lehenago aipatu moduan, ezker edo eskuin botoiei sakatuz gero 90 graduko biraketa egingo du aukeraturiko norabidean.

A.4 Laguntza sekzioa

Menu nagusian, funtzionalitate bakoitzaren botoiaren gainean kurtsorea badago, mezu bat agertuko da funtzionalitatearen deskribapen txiki batekin.

Interfaze grafikoak bere laguntza sekzioa du, funtzionalitate bakoitzeko botoiak nola erabili (ikus A.11 irudia) eta bakoitzaren maneiua (ikusi A.12 irudia) erakusten duena. Biraketa normala eta Biraketa Zoroa funtzionalitateen kasuan beste laguntza leiho bat gehiago dago, (ikus A.13 irudia), abiadura aukeraketarena alegia. Menuan *Laguntza* erlaitzaren bitartez aurki daitezke mezu hauek.

A.11 Irudia: Kontrolen laguntza



A.12 Irudia: Abiadura aukeraketaren laguntza

A.13 Irudia: Maneiuaren laguntza

B. Eranskina

B Instalazioa

Atal honetan aplikazioa *Ubuntu*-n behar bezala martxan jartzeko urratsak azalduko dira.

B.1 Pololu USB AVR programmer-en instalazioa

Linuxen *Pololu USB AVR programmer* (ikus 3.1.2 atala) erabili ahal izateko ez dugu inongo driveren instalazioen beharrik. Linuxen *Kernel*-ak *USB-to-serial* driverra dakar eta honek automatikoki funtzionatzea ahalbidetzen du inongo driverrik instalatu gabe.

USB programmer-a ordenagailura konektatzen denean, *CDC* (*Communications Device Class*) driverrak automatikoki hautemango du eta ondoren bi serie portu sortuko ditu (ikus B.1 irudia). Serie konexio hauexek izango dituzten izenak **`/dev/ttyACM0`** (programatzeko portua) eta **`/dev/ttyACM1`** (USB-to-TTL-serie egokigailua) izango dira.

B.1 Irudia: *ACM0* eta *ACM1* sorturiko portuak

B.2 AVR-ren beharrezko Softwarearen instalazioa

Linuxen *AVR* (Atmel-ek garaturiko *RISC* motako) mikrokontrolagailuak programatu ahal izateko, lehendabizi 3 software pakete hauek instalatu behar dira:

- **gcc-avr:** GNU-ko C-ko konpilatzailea
- **avr-libc:** *AVR*-ren funtzio berezi batzuetara sarbidea ahalbidetzen duen liburutegia
- **avrdude**: *AVR* mikrokontrolagailuak programatzeko programa

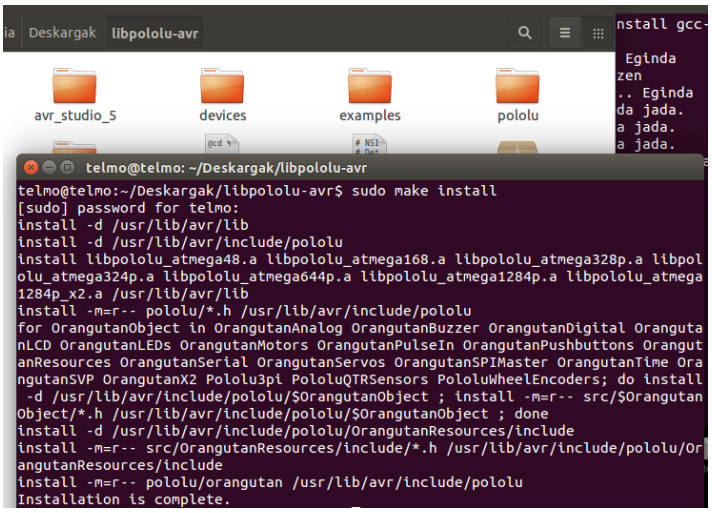
Instalatzerako orduan komando hau terminalean exekutatzea nahikoa izango da:

```
telmo@telmo:~$ sudo apt-get install gcc-avr avr-libc avrdude
```

B.2 Irudia: Programen instalazio komandoa

B.3 AVR C/C++ liburutegiaren instalazioa

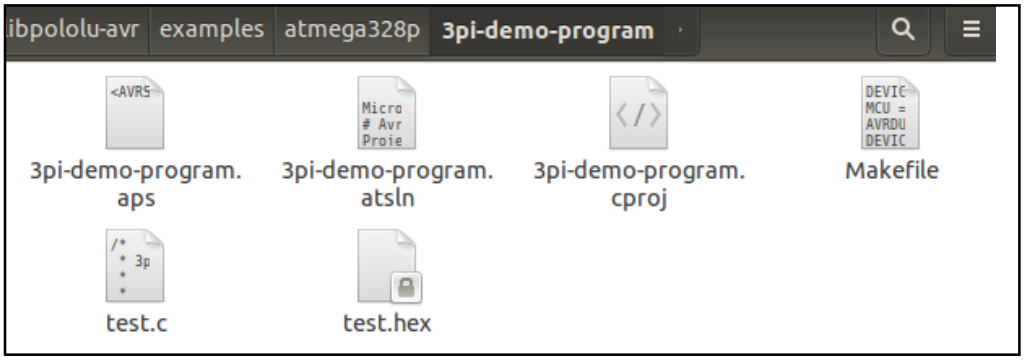
Lehendabizi http://www.pololu.com/file/download/libpololuavr140513.zip?file_id=0J757 helbidetik *.zip* fitxategia jaitsiko dugu. Behin fitxategia deskargatu eta nahi den lekuan erauzi ondoren, `libpololu-avr` karpetaren barnera joango gara. Karpetan bertan terminala ireki eta `sudo make install` (ikus B.3 irudia) exekutatuko dugu. Liburutegiko objektu (*.a*) eta buruko (*.h*) fitxategiak dagokion *avr* azpidirektorioetan kopiatuko dira.



B.3 Irudia: *libpololu-avr* liburutegiaren instalazioa

B.4 Probak

Pauso hauek jarraiturik jada prest gaude gure programak eta probak egiteko. Modurik errazena ea instalazioa ondo egin den jakiteko, *libpololu-avr* liburutegiak eskaintzen dizkigun demo programa horiek exekutatzea da. Horretarako B.4 irudian agertzen den `/libpololu-avr/examples/atmega328p/3pi-demo-program` direktoriora joan beharko da. Bertan *test.c* fitxategia egongo da, eta bera izango da konpilatu beharreko fitxategia.



B.4 Irudia: *3pi-demo-program* karpeta

Karpeta horretan terminal bat irekiko dugu, eta bertan fitxategi hori konpilatzeko *make* komandoa exekutatuko da. Errorerik ez bada izan, programa robotera pasatzea besterik ez da geratzen. Horretarako terminal berdinetik *make program* komandoa exekutatuko da B.5 irudian ikus daitekeen moduan, eta dena ongi joan bada robota prest egongo da.

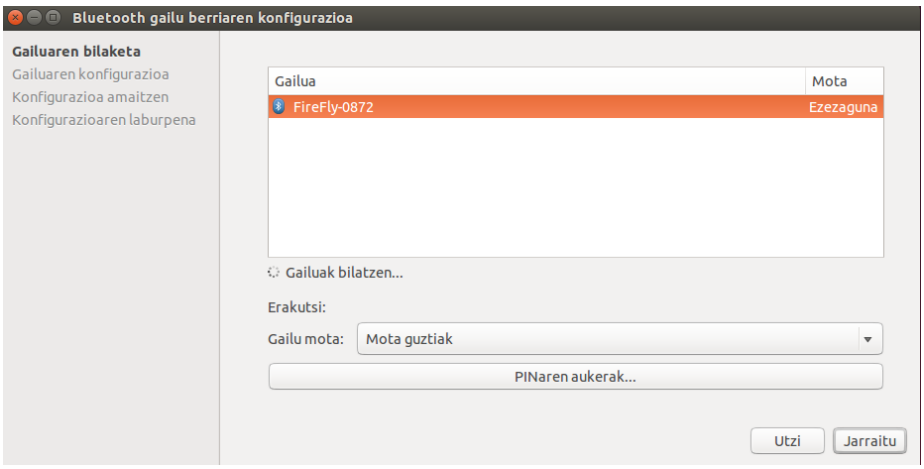
```
telmo@telmo:~/Deskargak/libpololu-avr/examples/atmega328p/3pi-demo-program$ m
ake program
avrdude -p m328p -c avrisp2 -P /dev/ttyACM0 -U flash:w:test.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be perf
ormed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "test.hex"
avrdude: input file test.hex auto detected as Intel Hex
avrdude: writing flash (10000 bytes):

Writing | ################################################## | 100% 1.15s

avrdude: 10000 bytes of flash written
```

B.5 Irudia: *make-program* exekutatzean

B.5 Bluetooth

Lehenago aipatu moduan robotak *bluetooth* gailu bat du erantsia, zeinaren instalazioa nahiko sinplea den *Linux*-en.

1. Lehendabizi ***bluez*** (bluetooth portuetako zerbitzu guztiak kudeatzen dituen aplikazioa) instalatuko dugu `apt-get install bluez` komandoaren bitartez.
2. Jarraian ordenagailuan *bluetooth* ikurrari sakatu eta bertan gailu berri bat konfiguratu nahi duzula jartzen duen aukeran klikatu.
3. Ondoren robota piztu, eta ordenagailuak berehala hautemango du *bluetooth* gailua, kasu honetan *FireFly-0872* izena izango du B.6 irudian ikus daitekeen bezala.



B.6 Irudia: *Bluetooth* gailuaren detekzioa

4. Behin gailua detektatua, hurrengo pausoan B.7 irudian ikusi bezala *Bat dator* –i klikatuko diogu.

B.7 Irudia: *PIN*-a bat datorrela berretsi

5. Ez bada errorerik gertatu prozesuan *Blue-Smirf bluetooth* gailua ongi konfiguratua egongo da.

   Urrats guzti hauek ongi eman badira, prest egongo gara robotarekin lanean hasteko.

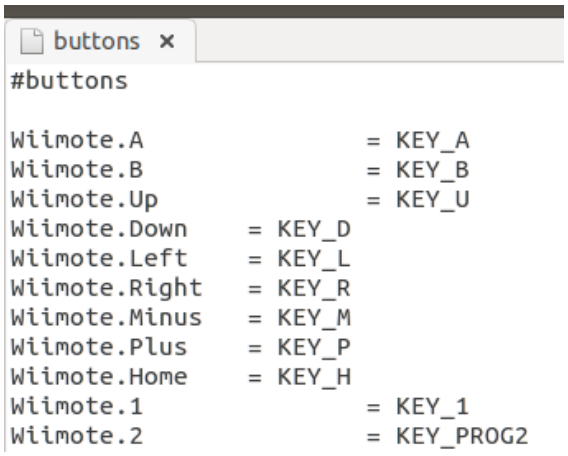
B.6 Wiimote-a

Wiimote-aren erabilera oso garrantzitsua izan da proiektuan zehar, berarekin maneiatu izan baita robota. *Wiimote*-ak teklatuaren papera hartu du, berau emulatuz. Horretarako *wminput* paketea instalatu behar izan da `apt-get install wminput` komandoaren bitartez.

Behin paketea instalatua dugula, *wminput* paketea dagoen direktoriora joko dugu eta B.8 irudian ikus dezakegun *buttons* fitxategia irekiko dugu.

B.8 Irudia: *wminput* paketearen direktorioa

Behin fitxategian, *Wiimote*-ko botoi bakoitza teklatuko zein botoirekin parekatzea nahi dugun erabaki behar da. Kasu honetan B.9 irudian ikus daiteke egin diren parekatzeak.



B.9 Irudia: *buttons* fitxategia

C. Eranskina

C Bilera Aktak

Atal honetan, proiektuan zehar egin diren bilerak azalduko dira. Guztira 10 bilera ofizial egon dira, baina aipatu behar da zalantza txikiren bat egon bada bulegora joan izan naizela galdetzera edota *email* bidez argitu dela, eta horiek ez dira bilera moduan hartu.

1. Bilera

Data: 2014-ko urtarrilaren 20a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Proiektuaren ibilbidea zehaztu
- Zein sistema eragile erabili erabaki
- Zein *bluetooth* gailu eta nola erabili zehaztu
- Orain arte egindakoaren aurkezpena
- *Expansion* Kit-a beharko dugula erabaki
- Oier Mees-ek esandakoari buruz hitz egin

2. Bilera

Data: 2014-ko otsailaren 13

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- *Bluetooth* gailua(*BlueSmirf)* expansion kit-ean soldatu eztainuarekin
- Programaren erabilpen kasu berriak proposatu dira
- Programa garatzeko inguruneari buruzko galderak
- Orain arte egindakoaren aurkezpena

3. Bilera

Data: 2014-ko otsailaren 25a

Partaideak: Txelo Ruiz

Lekua: 1.2 laborategia

- *Pololu* 3pi-ren erabilpenaren aurkezpen txikia *Wii*-ko urrutiko agintearen bitartez
- Programan egin daitezkeen hobekuntzak komentatu
- Dokumentazioari buruzko zalantza batzuk argitu
- Pilek ez dutela ongi kargatzen helarazi

4. Bilera

Data: 2014-ko martxoaren 28a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Robotean pilak kargatu ahal izateko muntaia egin
- Interfaze grafiko bat egitearen ideia planteatu
- Kodeko zuzenketak proposatu

5. Bilera

Data: 2014-ko apirilaren 4a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Interfazea egiteko zein liburutegi, software, lengoaia.. erabili adostu
- *Wiimote*-a mugituz funtzionalitatea egiteko dauden zailtasunak azaldu
- Bateria-kargadorearekin robotean dauden arazoak konpondu

6. Bilera

Data: 2014-ko maiatzaren 5a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Serie komunikazioa interfazearen bitartez egiteko dauden zailtasunak planteatu
- Interfaze grafikoaren diseinu finkoa adostu
- Memoriari buruzko aholkuak eman

7. Bilera

Data: 2014-ko maiatzaren 9a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Serie komunikazioari buruz egindako aurrerapen nabarmenak erakutsi interfaze grafikoan
- *Wiimote*-a mugituz funtzionalitatean nituen zailtasunei aurre egin diedala erakutsi
- Inplementazioa bukatzeko data finkatu

8. Bilera

Data: 2014-ko maiatzaren 20a

Partaideak: Elena Lazkano eta Txelo Ruiz

Lekua: Robotikako laborategia

- Inplementazioari buruzko azken ukitu eta aldaketak egin
- *system* funtzioari esker terminalean exekutatu behar ziren komandoak interfazetik abiatzea lortu
- Egindako prototipoaren hobespena lortu

9. Bilera

Data: 2014-ko maiatzaren 28a

Partaideak: Txelo Ruiz

Lekua: Txeloren bulegoa

- Memoriaren itxura, diseinua, marjina, edukia, eta halakoei buruzko hainbat galdera egin
- Ordurarte egindakoari buruzko iritzia eta zuzenketak komentatu

10. Bilera

Data: 2014-ko ekainaren 12

Partaideak: Txelo Ruiz

Lekua: Txeloren bulegoa

- Ordurarte memorian idatzitakori buruzko hainbat galdera, memoria perfekzionatzeko.
- Egin beharreko hobekuntza eta zuzenketak komentatu
- Memoria bukatzeko gutxi gora beherako data ezarri