eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea

OUTDOOR ROBOT MUGIKOR BATEN ERAIKUNTZA ETA KONTROLA ARDUINO BITARTEZ

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

Konputagailuen Ingeniaritza

Egilea: Nerea Murua Sarriegi

Tutoreak: Txelo Ruiz eta Elena Lazkano

informatika fakultatea — facultad de informática

Donostia, 2015-eko Ekaina

AURKIBIDEA

IRUDIEN AURKIBIDEA

TAULEN AURKIBIDEA

1. LABURPENA

Proiektu honen helburua Outdoor robot mugikor bat eraiki eta kontrolatzea da. Proiektu hau RSAIT ikerketa taldearentzat egin da, eta ezinbestekoa izan da talde honen egungo egoera eta helburuak ezagutzea proiektuaren helburuak ulertu ahal izateko.

Robotika eta Sistema Autonomoen Ikerketa Taldea Euskal Herriko Unibertsitateko Informatika Fakultatean sortu zen eta egun, bertan jarraitzen du bost informatikari, bi matematikari eta industria-ingeniari batez osatutako taldearekin.

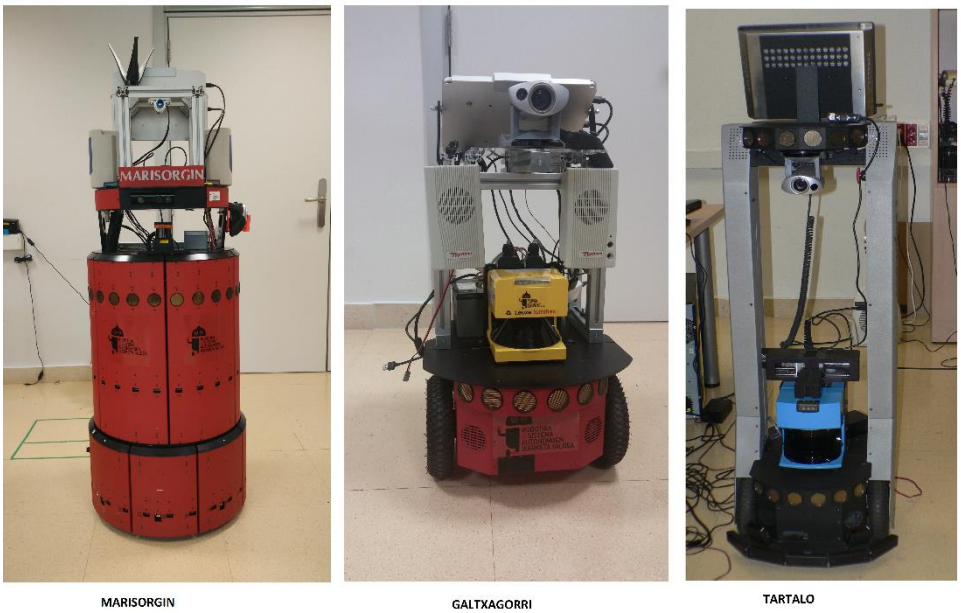
Taldearen ikerkuntza alor nagusia robotika mugikorra da eta bertan aplikatzen dira estatistika eta ikasketa autonomorako teknikak, robotaren autonomia maila areagotzeko. Taldeak honako gai hauek lantzen ditu:

- Robot autonomoen esplorazioa eta nabigazioa
- Ikasketa automatikoa (Machine Learning)
- Gizaki eta roboten arteko elkarrekintza
- Ordenagailu bidezko ikusmena
- Estatistika

ROBOTAK

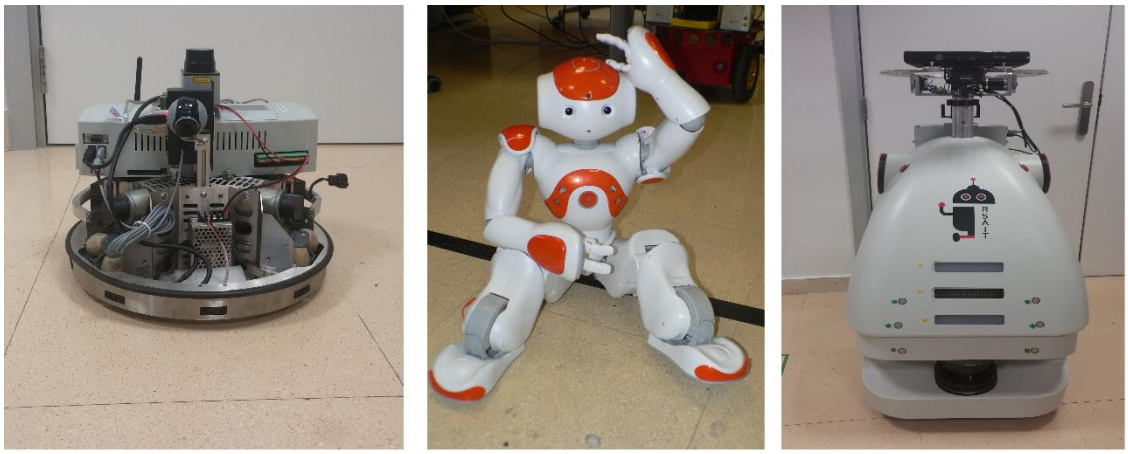
RSAIT taldean robot desberdinak daude ikerketa lanak burutzeko. Robot hauek mugimendu-sistema desberdinak dituzte, sinkronoa batek, diferentziala gehienek eta holonomoa Robotinoak. Bestalde, robot guztiak ROS sistemaren bidez kontrolatzen dituzte.

Hona hemen egun taldean dituzten robotak:



MARISORGIN GALTXAGORRI TARTALO

1. irudia: RSAIT taldeko robotak 1



2. irudia: RSAIT taldeko robotak 2

Robot hauek guztiek bi ezaugarri dituzte amankomunean: Denak dira *indoor* motako robotak eta denak robot komertzialak dira (eraikita erosi dira). Hau kontuan izanik, proiektu honen helburua ezaugarri hauek apurtuko dituen robot baten garapena da: **kanpoan** erabiliko den robot bat garatu eta **eraikitzea**, hain zuzen ere.

Proiektu honetan garatutako robota, aurrerago oinarri gisa erabiliko da proiektu berrietarako, robotari osagai gehiago gehituz eta funtzionalitate berriak eskainiz. Hau kontuan izanik, garrantzitsua izan da proiektua ongi dokumentatzea.

2. PLANGINTZA

Atal honetan proiektua aurrera eramateko eginiko plangintza dago ikusgai. Proiektu baten arrakastan izugarrizko garrantzia du plangintzak, beraz, ataza honi garrantzia handia eman behar zaio. Plangintza hau aurrera eramateko erreferentzia gisa *A guide to the project management body of konwledge* (Project Management Institute, 2008) liburua erabili da.

Atal honetan plangintza batean oinarrizkoak diren elementuak zehaztu dira; hala nola, irismena, ataza eta azpiatazen deskribapena, emangarrien identifikazioa eta ezaugarriak, denboraren plangintza, lan metodologia eta kalitate eta arriskuen plana.

2. 1 IRISMENA

Proiektu baten irismena aztertzerakoan, kontuan izan behar da proiektuen testuinguruan irismenak bi adierazpen dituela: batetik, produktuaren irismena, eta bestetik, proiektuarena:

- **Produktuaren irismena:** Produktu, zerbitzu edo emaitza definitzen duten ezaugarri eta funtzioak.

- **Proiektuaren irismena:** Produktu, zerbitzu edo emaitza bat ezaugarri eta funtzio jakin batekin lortzeko egin beharreko lana.

Irismena finkatzerako orduan, lehenik eta behin proiektuaren/produktuaren betekizunak bildu behar dira. Ondoren, datu hauek aztertuz, irismena zehaztuko da eta puntu honetan proiektuko LDEa (Lanaren Deskonposaketa Egitura) eraikitzeko beharreko informazioa osoa edukiko da. Behin irismenak finkatuta, ezinbestekoa izango da irismenaren egiaztapen eta kontrola burutzea, finkatutakoa betetzen ari dela ziurtatzeko; eta hala ez bada, beharrezko neurriak hartzeko.

2.1.1 Produktuaren irismena

Proiektuaren helburua *Outdoor* robot mugikor bat eraiki eta hau mikrokontrolagailu baten bitartez kontrolatzea da. Beraz, esan liteke proiektu honetan alor edo diziplina ezberdinak landu direla. Batetik, robotaren **eraikuntza** burutu da, bertan, robotaren ezaugarri eta helburuetarako diseinu egokia zehaztu da eta ondoren, diseinatutako robota eraiki, beharrezko suerta daitezkeen erraminta eta osagaiak erabiliz. Bestalde, robotaren **elikadurak** ere garrantzia handia du proiektuan; robotak zein elikadura behar duen zehaztu, elikaduraren kokapen optimoa finkatu eta ondoren,

elikadura iturria kargatzeko eta piztu edo itzaltzeko aukera eskaini. **Hardwarearen** alorrak ere berebiziko garrantzia du; robotaren osagai bakoitzerako oso garrantzitsua da honen funtzionamendu, egitura, konexioak eta elikadura ezagutzea; ondoren, zuzen programatu ahal izateko. Azkenik, **softwarearen** atala ere landu da; behin robotaren osagai bakoitzaren funtzionamendua ezagutu ondoren, desio den portaera programatuz.

Proiektu hau burutzeko, Romeo-All in one mirkokontrolagailua erabili da; robotaren programa bertan kargatu da eta osagai guztiak mikrokontrolagailu honetara konektatu dira.

Hona hemen eraiki eta programatu den robotak izan beharreko ezaugarri nagusien zerrenda:

- Robota kanpo ingurune ez-kontrolatu batean ibiliko da
- Korronte zuzeneko 4 motor izango ditu mugimendurako
- Robotaren nabigazioa, bluetooth bidez bidalitako aginduen bidez burutuko da
- Nahiz eta urruneko agindu bidez kontrolatu robota, segurtasun neurri gisa, ultrasoinu sentsore bat izango du aurrealdean, talkak ekiditeko.
- Robotak GPS bat izango du bere uneko posizioari dagozkion latitude eta longitude balioak kalkulatzeko.

Ikus daitekeen bezala, ez dago finkaturik zehatz-mehatz robotak zein osagai izango dituen; honek izan beharreko ezaugarriak baizik. Aipatutako ezaugarri bakoitza lortzeko ahaleginean, horretarako eskuragarri dauden osagai eta beharren inguruko hausnarketa bat egin da eta bertan erabaki da, dauden aukeren artean, zein diren beharko diren osagaiak ezaugarri hori lortu ahal izateko.

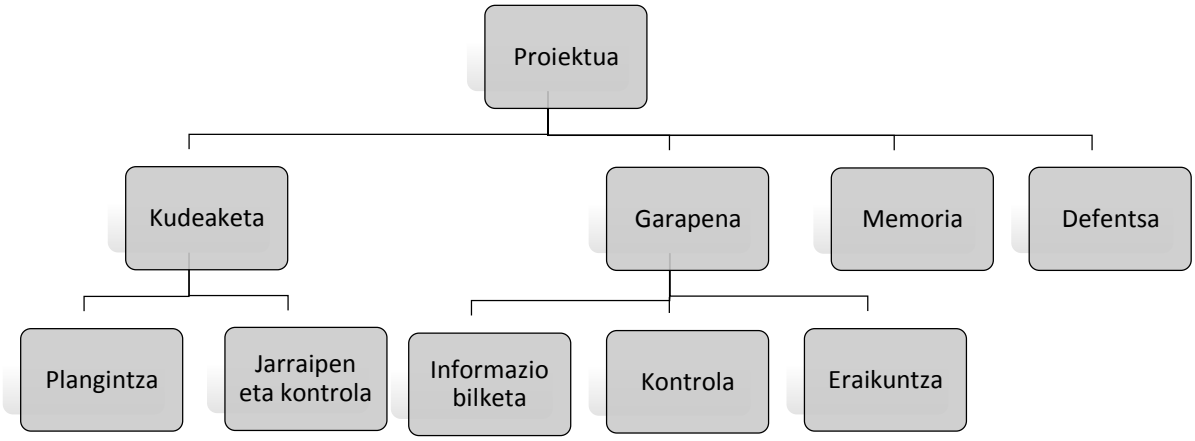
2.1.2 Proiektuaren irismena

Aurreko puntuan deskribatutako produktua helburu horiekin lortzeko egin beharreko lanari dagokionez, honako hau espero izan da: Kalitate planean finkatukotako oinarrizko kalitate maila lortzea, eta oinarrizko maila hau lortu ahal izateko denbora nahikoa izatea, ataza bakoitzari behar duen denbora eskainiz. Behin oinarrizko kalitate maila lortuta, kalitate gehiago eskuratzeko denbora izatea ere espero izan da.

Bestalde, proiektu honek ikaslearen ikasketa zikloan laguntzea eta robotika mugikorraren inguruan sakontzea eta gehiago jakitea desio izan da, gai honekiko interesa asebetez.

2.1.3 Lanaren deskonposaketa egitura

Aipatutako helburuak lortzeko eta beharrezko emangarriak sortzeko jarraitutako lanaren deskonposaketa hierarkikoa honako hau izan da. Egitura honen bitartez, proiektuko irismen osoa antolatu eta definitzen da.



3. irudia: Lanaren Deskonposaketa Egitura

Beraz, proiektuan 4 ataza nagusi bereiztuko dira: Kudeaketa, garapena, memoria eta defentsa. Kudeaketak berebiziko garrantzia du proiektuaren arrakastan. Ataza honek bi azpiataza nagusi ditu: plangintza eta jarraipen eta kontrola.

Garapenari dagokionez, ataza honetan robot mugikorraren eraikuntza, kontrola eta informazio bilketa azpiatazak bereiztu dira. Lehen bi atazak (eraikuntza eta kontrola) bereiztuta dauden arren, aldi berean burutu dira, robotaren osagai bakoitzeko honako pauso hauek jarraituz: honen inguruko informazioa lortu, funtzionamendua ulertu eta desio den portaera lortzeko kontrol programa idatzi. Eraikuntzari dagokionez, garapenaren azken pauso gisa zehaztu da. Behin robotaren osagai guztiei dagozkien programak izanik, osagai hauek banan-banan elkartu dira; bai software aldetik, bai eta hardware aldetik ere.

Hirugarren ataza nagusia proiektuaren memoria egitea izan da. Memoria idazte hau aurreko bi atazekin batera egin da, proiektuan aurrera egin ahala, egindakoa idatziz.

Azkenik, behin aurreko atazak buruturik eta proiektua entregatuta, proiektuaren defentsaren prestaketa izan da burutu beharreko azken ataza.

Denboraren plangintzaren atalean zehaztu da ataza bakoitzari esleitutako denboraren portzentaia zein izan den, bai eta ataza hauek proiektuaren bizi zikloan zein momentutan burutzea planifikatu den ere.

2.2 ATAZA ETA AZPIATAZEN DESKRIBAPENA

Plangintza

Ataza honen helburua proiektuaren nahiz produktuaren betekizunak aztertu eta identifikatzea izan da. Bestalde, kudeaketa plana ere burutu da, proiektuaren arrakasta ziurtatzeko jarraitutako plana zein izango zen zehaztuz.

Jarraipen eta kontrola

Ataza honen bidez, proiektuaren arrakasta ziurtatzeko honako hauen jarraipen eta kontrola egin beharko da: proiektuaren nahiz produktuaren irismenarena, kalitatearena, arriskuena eta denboraren jarraipen eta kontrola. Modu honetan, proiektua bide onetik doala ziurtatu da, eta hala ez den kasurako, arazoak atzeman eta berandu izan aurretik konponbide bat emateko aukera izan da.

Informazio bilketa

Robotak izan beharreko aipatutako ezaugarriak lortze aldera, ezinbestekoa izan da informazio bilketa prozesu bat aurrera eramatea, gailuen inguruko informazioa eskuratu eta ulertzea, ondoren nola programatu jakin ahal izateko.

Kontrola

Azpiataza honen helburua robotaren kontrol programa garatzea izan da.

Eraikuntza

Azpiataza honetan robotaren eraikuntza burutu da. Hasieran, robotaren oinarrizko egitura eraiki da; eta ondoren, osagai berriak gehitu zaizkio oinarrizko egitura honi.

Memoria idatzi

Gradu amaierako proiektuaren emangarrietako bat proiektuaren memoria da. Memoria honetan proiektuaren inguruko informazio osoa bildu da eta beraz, proiektuaren bizi ziklo osoan zehar egon da present ataza hau.

Defentsa

Behin proiektua amaitu eta entregatuta, azken eginbeharra proiektuaren defentsa prestatzea izan da.

2.3 EMANGARRIEN IDENTIFIKAZIOA ETA EZAUGARRIAK

Komenigarria da proiektu batekin hasterako, garbi izatea zein izango diren entregatu beharreko emangarriak eta hauen ezaugarriak. Hiru emangarri daude proiektu honetan: robota (produktua), egindako lana biltzen duen memoria eta proiektuaren aurkezpena (defentsa).

2.3.1 Produktua

Produktua, aurrez aipatu bezala, *outdoor* robot mugikorra izango da, irismenean finkatutako osagai eta kontrol ezaugarriekin. Produktu hau ekainaren 24 baino lehen amaitu behar izan da. Robotaz gain, garatutako kontrol programak ere emangarri honetan sartzen dira.

2.3.2 Memoria

Emangarri honetan proiektuaren kudeaketa eta garapenaren inguruko informazio osoa bildu da. Emangarri honek bete beharreko ezaugarriak bibliografian aipatutako *Memoriaren ezaugarri eta betebeharrak* erreferentzian ikus daitezke. Dokumentu honetan, estilo eta edukiaren inguruko betebeharrak zehazten dira, eta beraz, egingo den memoriaren emangarriak betekizun horiek bete behar izan ditu.

Emangarri hau ekainaren 24rako egin behar izan da eta ADDI plataforma digitalera igo.

2.3.3 Defentsa

Azken emangarria proiektuaren aurkezpena, defentsa izan da. Hau 2015eko uztailaren 8tik 10era bitarteko egun batean burutuko da, espezialitateari dagokion epaimahaiaren aurrean.

2.4 DENBORAREN PLANGINTZA

Gradu amaierako proiektuek 12 kredituko pisua dute, beraz, 300 orduko lan dedikazioa eskatzen dute. Proiektua kurtso hasieratik dago esleituta, baina 300 ordu hauen banaketa ez da uniformea izan kurtsoan zehar; lehenengo lauhilekoko irakasgai kopurua handiagoa izan delako bigarren lauhilekokoa baino. Bestalde, proiektua 2015eko ekainaren 24an entregatu behar izan da.

Lehenengo lauhilekoan egin beharrekoa proiektuaren plangintza osoa izan da. Plangintza egiteaz gain, erabiliko den Romeo-All in one mikrokontrolagailuaren inguruko informazio bilketa eta ikasketa ere egin dira, urtarrilean proiektuaren garapenarekin hasi ahal izateko beharrezko ezagutza izate aldera.

Beraz, helburua, urtarrilerako proiektuaren plangintza osoa eginda izatea izan da eta erabilitako mikrokontrolagailua eta hau programatzeko softwarea ezagutzea. Urtarriletik aurrera proiektuaren garapena, kudeaketa eta memoria idaztea izan dira ardura nagusiak.

Aipatutakoa plangintza orokorra litzateke, jarraian datozen azpiataletan denboraren plangintzaren inguruko informazio zehatzagoa dago ikusgai.

2.4.1 Mugarrien diagrama

Garrantzitsua da hasieratik proiektuaren mugarriak, hots, data esanguratsuak ezagutzea, eta horretaz gain, norberak bere buruari mugak jartzea zenbait ataza burutzeko. Jarraian ikus dezakegun taulan agertzen dira mugarri hauek; batzuk hasieratik ezarrita zeudenak  dira (epe ofizialak direlako), eta besteak, proiektu honetarako bereziki jarriak  .

Iraila	Urria	Azaroa	Abendua	Urtarrila	Otsaila	Martxoa	Apirila	Maiatza	Ekaina	Uztaila
1	1		2	3			4		5 2	3

1.taula: Proiektuko mugarrien diagrama

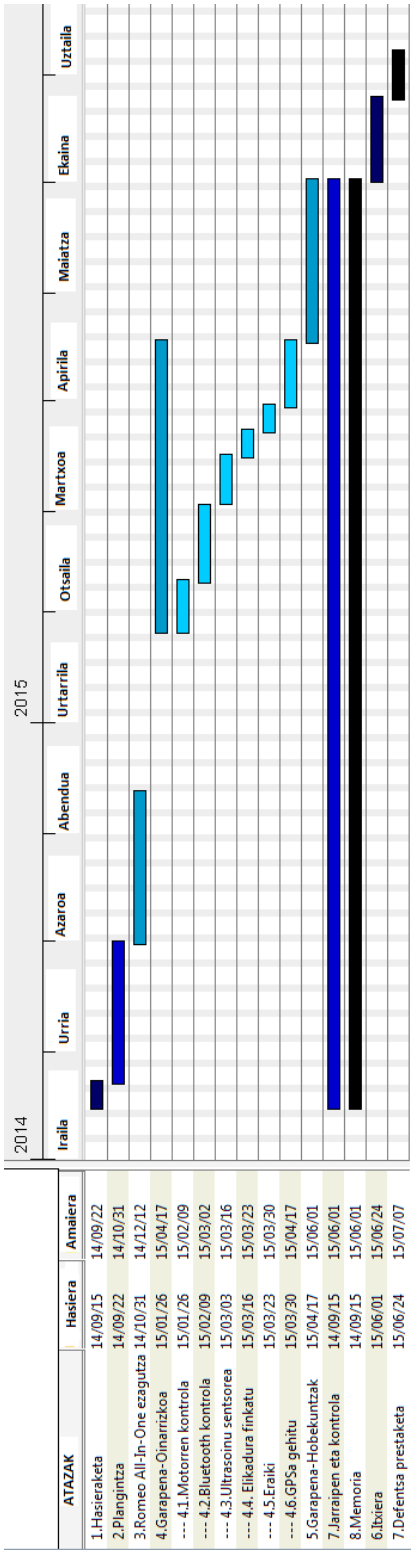
 1	2014-09-15	Proiektuaren hasieraketa
 1	2014-10-31	Proiektuaren plangintza eginda
 2	2014-12-12	Mikrokontrolagailuaren inguruko informazio bilketa amaituta
 3	2015-01-26	Robotaren garapenarekin hasi
 4	2015-04-17	Oinarrizko kalitate maila lortuta
 5	2015-06-01	Proiektuaren itxiera
 2	2015-06-24	Proiektuaren entrega
 3	2015-07-08/10	Proiektuaren defentsa

2.taula: Proiektuko mugarrien diagramaren azalpena

Aipatutakoak izan dira proiektuaren mugarri nagusiak; hala eta guztiz ere, proiektuan aurrera egin ahala, epe laburreko mugarri berriak jarri dira.

2.4.2 Kronograma

Behin mugarriak identifikatuta eta finkatuta, hurrengo pausoa denboran zehar atazak kokatzea izan da; hau da, ataza bakoitza nondik nora burutuko zen finkatzea. Datu hauek kronograma batean erakutsiko dira, modu grafikoan ulergarriago suertatzen baita atazen kokapen eta iraupena, bai eta atazen arteko dependentziak ere.



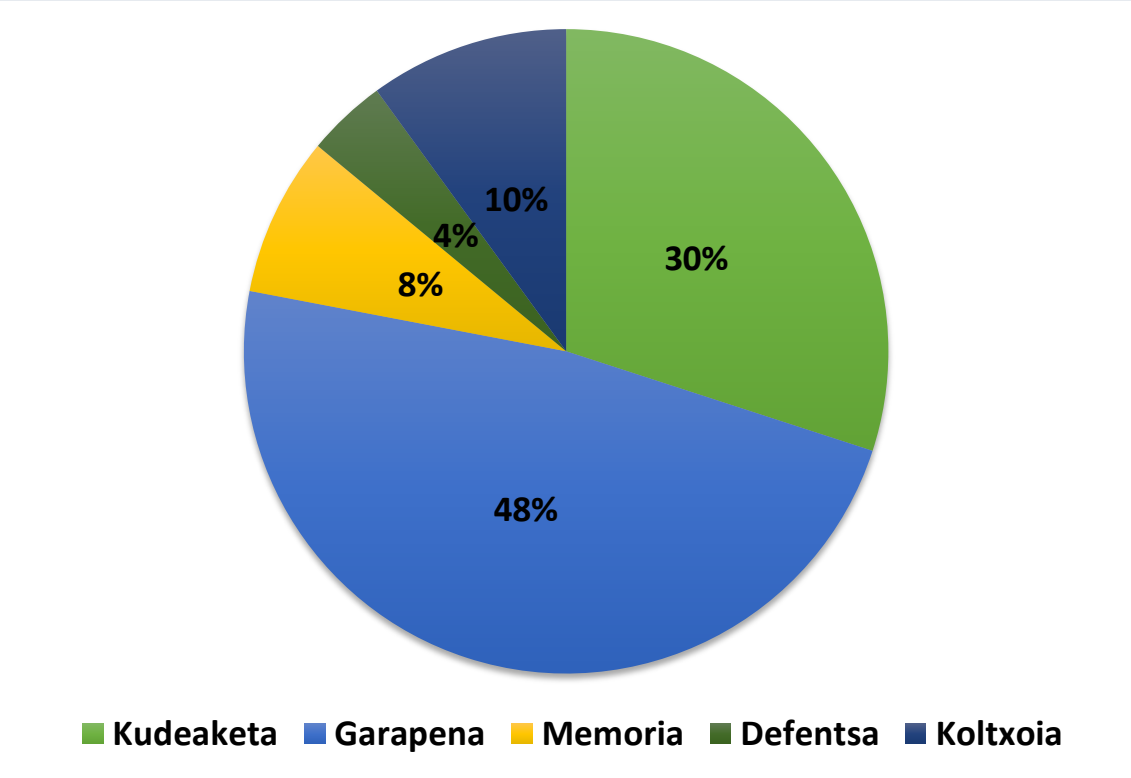
ATAZAK	Hasiera	Amaiera
1.Hasieraketa	14/09/15	14/09/22
2.Plangintza	14/09/22	14/10/31
3.Romeo All-In-One ezagutza	14/10/31	14/12/12
4.Garapena-Oinarrizkoa	15/01/26	15/04/17
--- 4.1.Motorren kontrola	15/01/26	15/02/09
--- 4.2.Bluetooth kontrola	15/02/09	15/03/02
--- 4.3.Ultrasoinu sentsorea	15/03/03	15/03/16
--- 4.4. Elikadura finkatu	15/03/16	15/03/23
--- 4.5.Eraiki	15/03/23	15/03/30
--- 4.6.GPSa gehitu	15/03/30	15/04/17
5.Garapena-Hobekuntzak	15/04/17	15/06/01
7.Jarraipen eta kontrola	14/09/15	15/06/01
8.Memoria	14/09/15	15/06/01
6.Itxiera	15/06/01	15/06/24
7.Defentsa prestaketa	15/06/24	15/07/07

4.irudia: Kronograma

2.4.4 Proiektuko lan banaketa orokorra

Proiektuaren irismena eta mugarriak kontuan izanda, hurrengo egitekoa atazei orduak esleitzea izan da, orain arte finkatutako datuak kontuan hartuz. Proiekturako 300 ordu dauden arren, lan banaketa honetan, ez dira 300 ordu horiek esleitu, orduen %10a erreserbatu egin da, hurrengo atalean azalduko diren helburuetarako.

Beraz, jarraian dagoen grafikoan, planifikatutako lan banaketa orokorra ikus daiteke, ataza bakoitzari esleitutako ehunekoa zehaztuz. Bestalde erreserbatutako orduak ere ikus daitezke, "koltxoi" izenez deiturik.



5.irudia: Planifikatutako lan banaketa

Garapenaren atalaren ondoren, jarraipen eta kontrolaren atalean, planifikatutako orduen eta benetan igarotako orduen arteko alderaketa dago ikusgai.

2.4.5 Erreserbatutako orduen erabilera

Aipatu bezala, planifikatzerako orduan proiektuari dagozkion 300 ordu horietatik %90a esleitu zaizkie ataza desberdinei. Proiektuaren irismena eta identifikatutako atazak kontuan izanik, espero izan da orduen %90a erabiliz, proiektuaren oinarrizko kalitate maila eskuratzea posible izango zela. Hala eta guztiz ere, hala izango ez balitz, oraindik 30 ordu libre izan dira kalitate maila hori eskuratu ahal izateko.

Behin oinarrizko maila lortuta, geratzen diren orduak proiektuari egin dakizkioken hobekuntzak egiteko erabiltzea planifikatu da; proiektuaren kalitate maila areagotuz.

Beraz, 30 ordu hauen erabileraren helburuak honako hauek izango dira: proiektuaren garapenean zehar suerta litezkeen arazoei aurre egiteko, lan dedikazioaren aurreikuspenean izan litezkeen erroreak konpontzeko eta proiektuaren kalitate maila handitzeko.

2.5 LAN METODOLOGIA

Burutu beharreko proiektua kontuan izanik, komenigarria suertatu da lanerako metodologia egokia finkatzea. Proiektu honetan, aste bakoitzaren hasieran aste horretako helburua zein den finkatu da, eta helburu hori lortzeko egin beharrekoa noiz eta nola egin erabaki.

Bestalde, aste bakoitzeko azken eguna kudeaketarako eta memoria idazteko erabili da. Kudeaketa egitea garrantzitsua da, proiektua bide onetik doala ziurtatzeko edota desbiderapenak garaiz antzemateko; arazoei aurre egin ahal izateko berandu izan baino lehen. Kudeaketa lan honetan orduen kontrola ere eraman da; astero-astero astean zehar proiektuari eskainitako orduak *excel* dokumentu batean txertatuz. Dokumentu honetan argi ikusiko da ataza bakoitzari zenbat ordu eskaini zaizkion, zenbat ordu sartu diren guztira proiektuan eta guztirako 300 ordu horietatik zenbat erabili diren.

Proiektuaren arriskurik larriena egindako lana galtzea izan da. Beraz, arrisku hau arazo bilaka ez dadin, aste bakoitzaren hasieran proiektuko karpetaren bi kopia gorde dira: bata kanpoko disko gogor batean eta bestea *Google Drive*-ra igo da. Kopia guztiak gorde dira eta nahasketarik ez gertatzeko, kopia bakoitza kopia egin den eguneko data jarriz izendatu da. Memoriari dagokionez, honetarako ere asteko azken eguna erabili da, astean zehar egindakoa idatziz. Guztiz komenigarria baita memoria pixkanaka idazten joatea eta ez uztea azkenerako.

Beraz, aste bakoitzean honako egitura hau jarraitu dut:

-Helburuak finkatu - Segurtasun kopiak egin	**Garapena**	**- Kudeaketa -Memoria idatzi**

3.taula: Asteen egitura

2.7 KALITATE PLANA

Proiektuaren kalitatea ere planifika eta kudea daitekeen zerbait da, beraz, plangintzan kalitateari ere eskaini zaio tartetxo bat. Kalitatea planifikatzerakoan, lehenik eta behin oinarrizko kalitate maila zein den finkatu da; hau da, proiektua amaitutzat jotzeko egin beharreko gutxieneko atazak zein diren finkatu.

Behin onargarritasun lerro hori lorturik, posible izango da, denbora izanez gero, kalitate maila areagotzea, kalitate dimentsioak gehituz. Oinarrizko kalitate lerroa lortu ahal izateko, proiektuaren irismenean aipatutako ezaugarriak bete behar ditu robotak; hau da:

- Robota kanpo ingurune ez-kontrolatu batean ibiliko da
- Korronte zuzeneko 4 motor izango ditu mugitu ahal izateko
- Robotaren nabigazioa, bluetooth bidez bidalitako aginduen bidez burutuko da
- Nahiz eta urruneko agindu bidez kontrolatu robota, segurtasun neurri gisa, ultrasoinu sentsore bat izango du aurrealdean talkak ekiditeko.
- Robotak, GPS bat izango du bere uneko posizioari dagozkion latitude eta longitude balioak kalkulatzeko.

Planifikatutakoaren arabera, ezaugarri hauek denak 2015eko apirilaren 17rako eginda izatea espero izan da. Behin onargarritasun maila hau eskuratuta eta hasieran erreserbatutako orduak libre izanez gero, proiektua hobetzeko saiakera egiteko aukera izanez.

Plangintzan proiektuaren kalitatea areagotzeko egin daitekeenaren inguruan hausnartu den arren, pentsatu da, garapenean aurrera egin ahala, errazago antzemango zirela egin litezkeen hobekuntzak. Beraz, plangintzan ez dira finkatu zein izango diren egin daitezkeen hobekuntzak. Behin oinarrizko maila lortuta, hobekuntza posibleen inguruan hausnartu da, aukera posibleak identifikatu eta geratzen diren orduen arabera, aukera posible hauetakoren bat (edo batzuk) aurrera eraman.

2.8 ARRISKUEN PLANA

Proiektua aurrera eramateko garaian, hainbat oztopo, arrisku topa zitezkeen bidean. Arrisku hauek aurrez identifikatzeari esker, hauek arazo ez bihurtzea ekidin ahal izan da. Jarraian aipatuko direnak dira proiektu honetarako identifikatutako arriskuak:

- **<u>Egindako lana galtzea</u>**

Proiektu honetako arrisku nagusia egindako lana galtzea da; bai memoria bai eta robotaren kontrol programa eta beharrezko diren beste fitxategiak ere. Arrisku hau

ekiditeko, aurrez aipatu bezala, aste bakoitzaren hasieran proiektuko karpetaren segurtasun kopia egin eta kanpoko disko gogor batean eta *Google Drive*-n gordeko da.

Probabilitatea: Txikia.

Eragina: Oso larria.

- **<u>Robotaren osagairen batek funtzionatzeari uztea</u>**

Robotaren osagai guztiak egoera onean egotea ezinbestekoa da robotak ongi funtzionatuko badu. Beraz, arrisku larri samarra litzateke hauetako osagairen batek funtzionatzeari uztea edota puskatzea. Hau ekiditeko egin daitekeen bakarra osagaiekin kontu handiz ibiltzea da eta hala ere zerbait izorratuko balitz, garrantzitsua izango da azkar antzematea eta konpontzen saiatzea. Konpontzerik ez badago, osagai berria eskatzea izango da azken irtenbidea.

Probabilitatea: Txikia.

Eragina: Larria.

3. LANEKO MATERIALA

Proiektu hau aurrera eraman ahal izateko erabili den laneko materiala zein izan den atal honetan azalduko da. Erabilitako osagai nagusiak bi izan dira: batetik, robotaren oinarrizko egitura osatzen duen *outdoor* plataforma eta bestetik, robota kontrolatzen duen Romeo-All in one mikrokontrolagailua. Oinarrizko bi osagai hauetaz gain, proiektuan aurrera egin ahala robotari hainbat osagai gehitu zaizkio, jarraian aipatuko den bezalaxe.

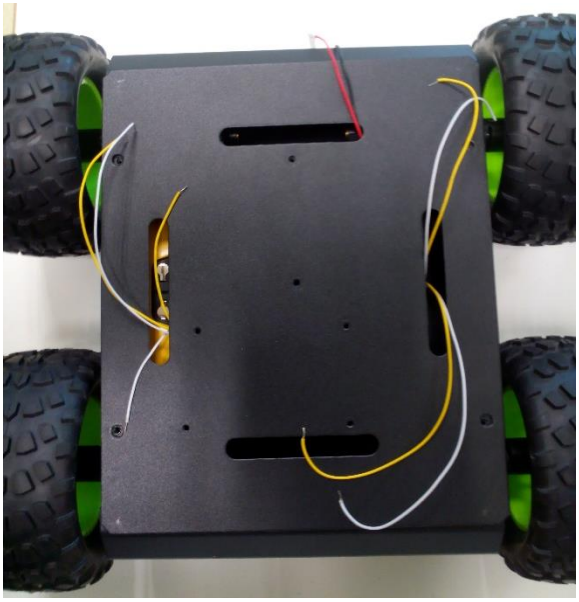
3.1 OUTDOOR PLATAFORMA

Erabilitako *outdoor* plataforma DFRobot enpresak egindakoa da: *4WD mobile Platform*. Plataforma hau Romeo-All in one mikrokontrolagailuarekin erabiltzeko helburuarekin diseinatuta dago eta 4 motor eta 4 gurpilez horniturik egoteaz gain, robot mugikorraren gorputza izango den txasis guztia ere badu.

Plataforma honen barruan nahikoa toki dago mikrokontrolagailua eta bateria sartu ahal izateko, honetaz gain, bestelako osagai gehiago gehitzea ere ahalbidetzen du; bai kaxaren barruan, bai eta gainean ere; hala nola, kamera bat, beso robotiko bat...

Plataforma honen ezaugarriak eta hemendik abiatuta osagai ugariko robot bat eraikitzeko aukera dela eta, plataforma hau oso egokia da bai robot txapelketetarako bai eta ikerketa alorrerako proiektuetarako ere.

Sentsore ugari gehi dakizkioke eta plataforma honen kutxa moduko egiturari esker, hauek babestuta egongo dira; plataforma oso sendoa baita.



6.irudia: *Outdoor* plataforma

3.1.1 Ezaugarriak

- Korronte zuzeneko 4 motor: 3 – 12 V
- Dimentsioak: 200 mm x 170 mm x 105 mm
- Gehienezko abiadura: 90 cm/s

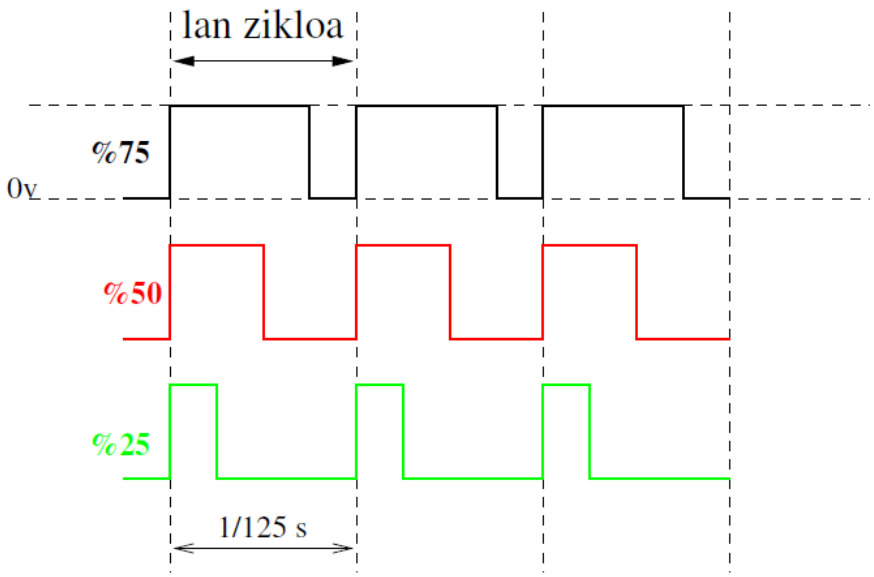
3.1.2 Motorren ezaugarriak

Outdoor plataforma honek, esan bezala, korronte zuzeneko lau motor dakartza. Gogora dezagun, mota desberdinetako motor elektrikoak daudela, baina kasu honetan korronte zuzenekoak dira erabili direnak. Motor hauek imanak, eroaleak eta korronte elektrikoa erabiliz eremu magnetikoa sortzen dute, eta eremu honek ahalbidetzen du motorraren ardatzean mugimendu errotazionala sortzea.

Gisa honetako motorrek bi terminal izaten dituzte. Tentsioa aplikatzean ardatza noranzko batean mugituko da, eta polaritatea aldatzean, berriz, beste norantzan. Abiadurari dagokionean, berriz, hau kontrolatzeko pultsu-zabaleraren bidezko modulazioa izenez ezagutzen den teknika erabiltzen da (PWM, *Pulse-width modulation*).

Teknika hau motorren elikapena aktibatu eta desaktibatzean oinarritzen da, beti tentsio maximoa aplikatuz. Beraz, abiadura aplikatzen den tentsioaren araberakoa izan ordez, tentsio maximoa aplikatzen den zikloaren zatiaren araberakoa da.

Teknika hau hobeto ulertzeko, bibliografian aipatutako *Robot mugikorrak. Oinarriak* (Aitzol Astigarraga & Elena Lazkano, 2011) liburura jo da.



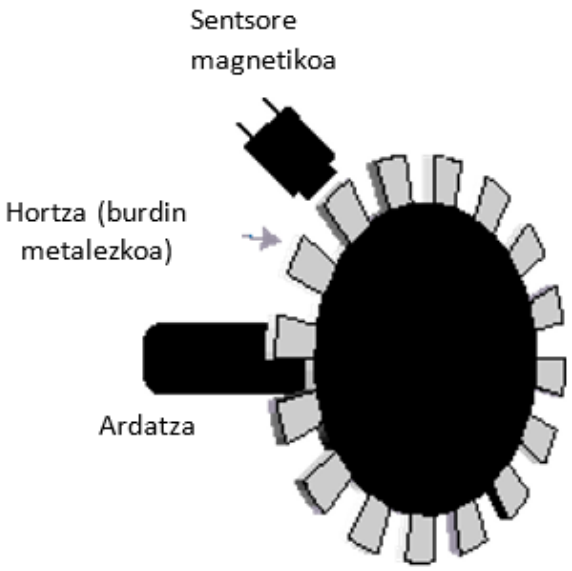
7.irudia: Pultsu-zabaleraren bidezko modulazioa

Beraz, abiadura motorra elikatuta dagoen lan-zikloaren zatiaren araberakoa izango da. Hau da, motorra lan-zikloaren %75ean elikaturik badago, azkarrago joango da %25ean aktibatuta dagoenean baino.

Behin motor hauen funtzionamendua azaldurik, jarraitu dezagun hauen ezaugarriak aztertuz. Motor hauek oso ahaltsuak dira, eta 12 Veko tentsioarekin lan egiten dute. Halere, tentsio baxuagoekin ere has daitezke biratzen.

Bestalde, motor hauetako bakoitzak, kodegailu magnetiko bana du integraturik. Kodegailu sentsore hauen bidez, robotaren posizioa kalkulatu daiteke motorren abiadura errotazionala eta norantzaren informaziotik abiatuz.

Kodegailu magnetiko hauek honako osagai hauek dituzte: iman permanente bat, sentsorea duen elementu magnetikoa eta burdinaz egindako horzdun disko bat. Disko honek biratu ahala, hortzek aldaketak eragiten dituzte eremu magnetikoan. Aldaketa hauexek dira sentsoreak jasotzen dituenak, irteera moduan pultsu segida sortuz.



8.irudia: Kodegailu magnetikoa

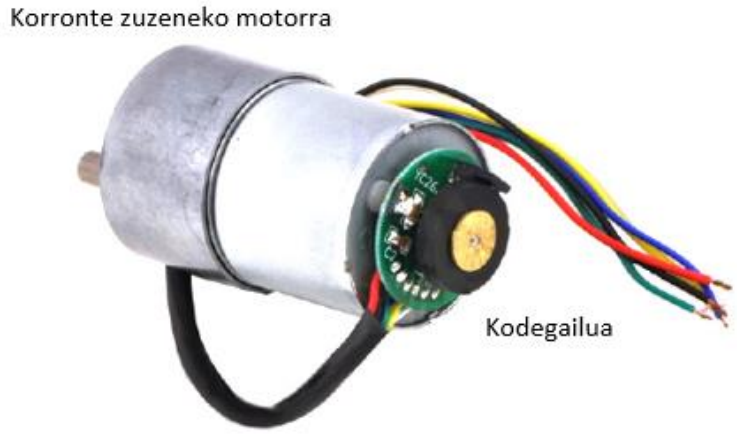
Kodegailu hau bi kanalez osatuta dagoenez, abiadura errotazionalaren informazioa emateaz gain, mugimendu errotazionalaren norantzaren berri ere ematen du. Norantza ezagutzeko, bi sentsore erabiltzen dira 90ºko desfase batekin.

Kodegailuak integratuta dituen motor hauetako bakoitzak 6 kable ditu, hona hemen bakoitzaren azalpena:

KOLOREA	FUNTZIOA
Txuria	Motorra (+)
Horia	Motorra (-)
Beltza	Kodegailua (+)
Gorria	Kodegailua (-)
Urdina	Kodegailuaren A irteera
Berdea	Kodegailuaren B irteera

4.taula: Motorren kableen azalpena

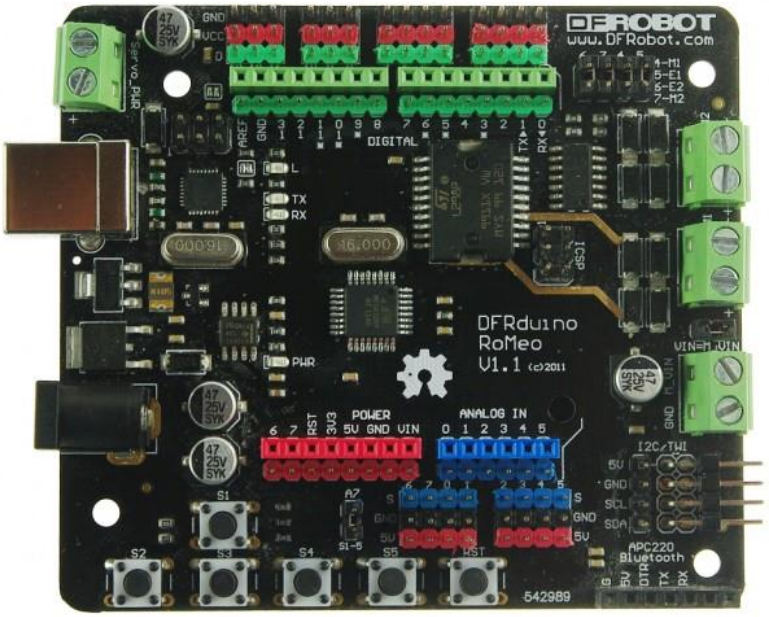
Taulan aipatzen diren A eta B irteera hauek motorraren biraketaren norantza jakiteko erabiltzen diren irteerak dira, 90º-ko desfasea dutenak.



9.irudia: Motorren osagaiak

3.2 ROMEO-ALL IN ONE MIKROKONTROLAGAILUA

Robotaren kontrola burutzeko Romeo-All in one mikrokontrolagailua erabili da. Mikrokontrolagailu hau DFRobot enpresak garatutakoa da, robot aplikazioetan erabiltzeko bereziki eta bestalde, Arduinorekin guztiz bateragarria da.



10.irudia: Romeo-All in one mikrokontrolagailua

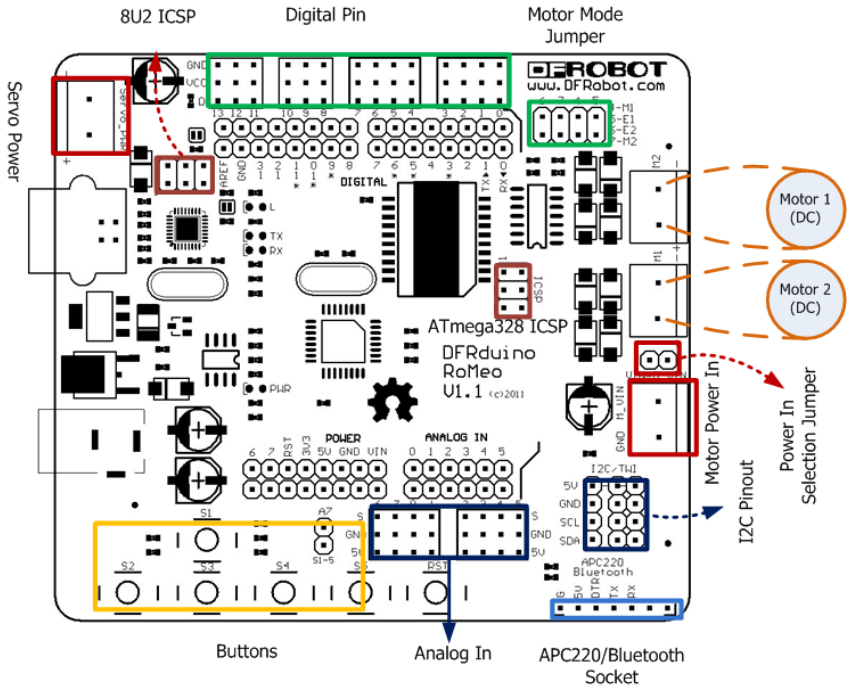
Mikrokontrolagailu honen ezaugarri nagusia korronte zuzeneko bi motor modu independentean kontrolatu ahal izateko 2 modulu dituela da; honetaz gain, haririk gabeko komunikazioa burutzeko *socket* bat ere badu. Ezaugarri hauei esker, gainerako osagairik gabe posible da proiektu txikiak egitea mikrokontrolagailu hau soilik erabilita.

Aipatu bezala, Romeo-All in one mikrokontrolagailua Arduinorekin guztiz bateragarria da; eta Arduino kontrolagailuekin bezalaxe, *shield*-ak gehitzeko aukera ematen du, bai DFRobotenak bai eta Arduinorenak ere. *Shield* hauek zer diren aurrerago azalduko da.

Hona hemen Romeo-All in one mikrokontrolagailuaren ezaugarri nagusiak:

- DC elikadura: USB bidez edo kanpo elikadura bidez 7 V – 12 V
- DC irteera: 5 V
- Pin-en mapa *Arduino Uno*-rekin bateragarria
- Sarrera/irteerako 10 bit-eko 8 kanal analogiko
- USB interfazea
- 5 sarrera botoi eta *Reset* botoia
- Serieko interfazea TTL mailan
- *Socket* integratuak: APC220 eta DF-Bluetooth modulua
- 3 I2C interfaze
- DC motorrak kontrolatzeko 2 modulu: 2 A korronte maximoarekin
- Tamaina: 90 x 80 x 14 mm
- Pisua: 60 g

11.irudian mikrokontrolagailuaren *pinout* diagrama ikus daiteke:



11.irudia: Romeo-All in one mikrokontrolagailuaren *pinout* digrama

Behin ezaugarri nagusiak eta mikrokontrolagailuaren sarrera/irteera diagrama ezagututa, honen osagaiak eta hauen ezaugarriak zein diren aztertu dira.

MIKROKONTROLAGAILUA	
Prozesadorea	ATmega32u4
Arkitektura	AVR
Erloju abiadura	16 MHz
Datu bitak	8 bit
RAM	2.5 KB
FLASH	32 KB
EEPROM	1 KB
Timer kopurua	3
Kanpo-etenak	2

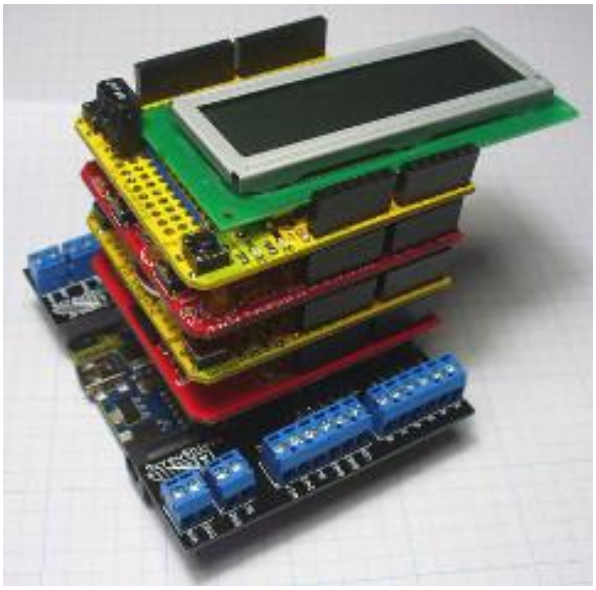
PWM kanalak	7
SARRERA IRTEERAK	
Helburu orokorreko S/I digitalak	17
Sarrera analogikoak	12
I2C portuak	3
SPI portuak	1
UART portuak	1
ERABILTZAILE INTERFAZEA	
LEDak	1
Botoiak	5
Reset botoia	1

5.taula: Romeo-All in one mikrokontrolagailuaren ezaugarri nagusiak

3.2.1 Arduino *Shields*

Arduino motako mikrokontrolagailuekin buruturiko proiektuetan, gehienetan ez da mikrokontrolagailua soilik erabiliko. Ohikoa da honen gainean beste maila batzuk gehitzea, lego piezak izango balira bezala, 12.irudian ikusten den moduan.

Mikrokontrolagailuari gehitzen zaizkion osagai hauek s*hield* bezala ezagutzen dira. *Shield* hauetako bakoitza plaka elektroniko bat da, mikrokontrolagailuaren gainean konektatu daitekeena eta bere gainean plaka gehiago konektatzeko aukera ematen duena. Shield hauei esker mikrokontrolagailuari funtzionalitate berriak gehitzen zaizkio, *shield* bakoitzak funtzionalitate jakin bat izango baitu.



12.irudia: Arduino *Shields*

Shield hauetako bat mikrokontrolagailuaren gainean gehitzean, mikrokontrolagailuaren pin guztiak *shield*-aren dagozkien pinekin konektatzen dira eta beraz, programatzerako garaian ez da ezer aldatzen.

Egun, *shield* ugari aurki ditzakegu gure proiektuen beharrak asetzeko. Batzuk aipatzearren, honako hauek dira 4 shield baliagarrienak erabiltzaileen iritziz:

- Ethernet shield
- Relay shield
- Protoshield
- LCD pantaila

Proiektu hau burutzeko *shield* desberdinak erabili dira. Hurrengo puntuan adieraziko da zein erabili diren eta zertarako.

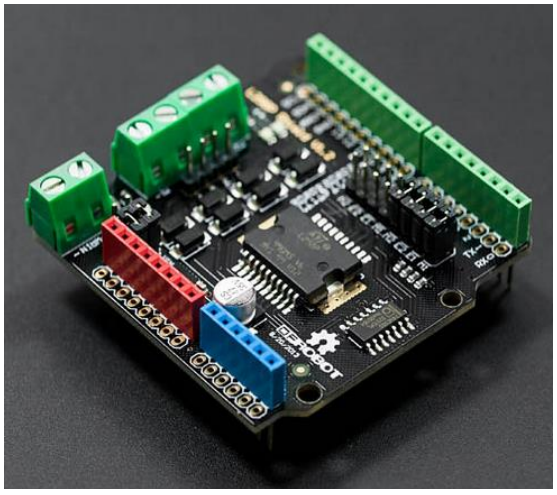
3.3 BEHARREZKO OSAGAIAK

Atal honetan, proiektuan aurrera egin ahala, robotari gehitu zaizkion osagaiak zein izan diren aipatu eta hauen ezaugarrien berri emango da. Jarraian aipatzen diren osagaiak proiektuari gehitutako ordena berean daude.

3.3.1 Motor *Shield*

Osagai hau Romeo mikrokontrolagailuari gehi dakioke, Romeok eman dezaken potentzia baino potentzia altuagoa behar duten motorrak kontrolatu ahal izateko. *Shield* honek L298P txip bat du integraturik eta honen bidez 7 – 12 V bitarteko korronte zuzeneko bi motor kontrolatzeko aukera ematen du, 2A-ko korronte maximoarekin.

Motorren abiadura kontrolatzeko aurrez aipatutako PWM teknika erabiltzen da, Romeo-ren PWM irteerako bi pin erabiliz (5 eta 6). Motor hauek aktibatu/desaktibatzeko aldiz, 4 eta 7 irteera pin-ak erabiltzen dira.



13.irudia: Motor *shield*

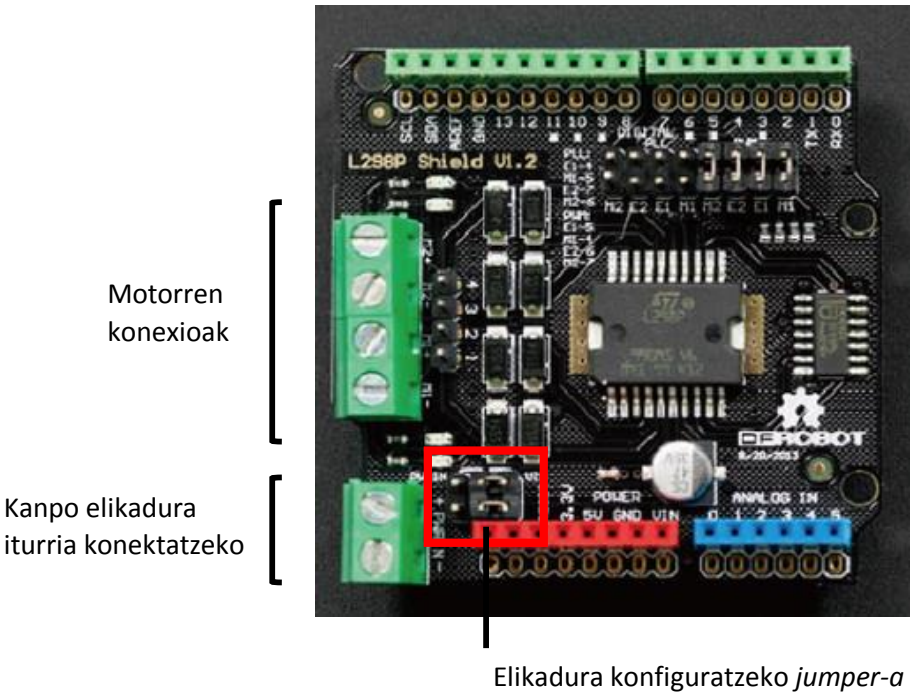
Motor Shield hau Romeo mikrokontrolagailuaren bidez elikatu daiteke, edota aparteko elikadura iturri bat erabiliz. Nahiz eta bi aukera izan, gehienetan oso komenigarria izango da aparteko elikadura iturri bat erabiltzea motorren kontrolerako jarraian ikusiko dugun arrazoia dela eta.

Elikadura iturriak konfiguratzeko, *shield* honek *jumper* bat dauka, eta hau 3 posiziotan egon daiteke 3 aukera desberdin eskainiz:

- Elikadura iturri bakarra erabiliz. Motor Shield-a Romeo-ra konektatutako elikadura iturriak elikatuko du.

- Elikadura iturri bakarra erabiliz. Baina, oraingo honetan, elikadura iturria Motor Shield-era konektatuko da, eta Romeo elikadura iturri honen bidez elikatuko da.

- Bi elikadura iturri erabiliz. Batak, Romeo mikrokontrolagailua elikatuko du eta beraz, mikrokontrolagailura konektatuko da zuzenean. Bigarren elikadura iturria Motor Shield-era konektatuko da eta honek Motor Shield-a soilik elikatuko du. Modu honetan, Romeo mikrokontrolagailua eta Motor Shield-a modu independentean elikatzen dira. Proiektu honetan aukera hau erabili da, motorrak elikatzeko 14,4 V-eko bateria bat erabili delako eta ez da komeni hau mikrokontrolagailura konektatzea honek jasan dezakeen tentsio maximoa 12 V delako eta beraz, kalteak eragin ditzakelako mikrokontrolagailuan.



14.irudia: Motor Shield-aren elikadura *jumper*-ak

3.3.2 DF-Bluetooth V3

Aipatu bezala, proiektuaren helburuetako bat robota urrunetik kontrolatzea da eta hau lortzeko *bluetooth* bidezko komunikazioa erabili da robota eta *smartphone* baten artean. Honetarako, beraz, ezinbestekoa izan da erabilitako mikrokontrolagailuari *bluetooth* modulu bat gehitzea: kasu honetan, DFRobot etxeko Bluetooth V3 modulua erabili da. Jarraian, modulu honen ezaugarriak azter daitezke.

15.irudia: Bluetooth modulua

Bluetooth modulu hau bi oholez osatuta dago, kalte elektroestatikoa ekiditeko helburuarekin. 3.5 V – 8 V bitarteko tentsio-horniketa onartzen du eta aplikazio askotarako egokia da.

Modulu honek bi funtzionamendu modu ditu: konfigurazio modua eta komunikazio modua. Konfigurazio moduan dagoela, moduluak AT komandoak onartzen ditu moduluaren konfigurazio parametroak zein diren kontsultatu edota aldatzeko. Puntu honetara iristean, komenigarria da AT komandoak (Hayes komandoak) zer diren azaltzea. Izan ere, aurrerago ere askotan aipatuko dira.

Hayes komando multzoa *Hayes Communications* konpainiak garatutako lengoaia da, eta, praktikoki modemak konfiguratu eta parametrizatzeko estandar irekia bihurtu da. Komando hauek guztiak “AT” karaktereekin hasten dira eta hauek *Attention* adierazten dute; hau dela eta, oso ohikoa da komando multzo hauei AT komando izenez deitzea. Nahiz eta hasiera batean modem komunikazioetarako sortu eta erabili ziren komando hauek, gaur egun hauen erabilera oso hedatua dago eta gailu desberdin askok eskaintzen dute AT komando bidezko konfigurazio eta komunikazio aukera.

Hona hemen bluetooth moduluaren AT komando erabilienen zerrenda:

- **AT** // Ongi funtzionatzen duela egiaztatzeko. OK erantzun behar du
- **AT + NAME** // Moduluaren izena zein den ikusteko
- **AT + ROLE** // Nagusi edo morroi gisa konfiguratuta dagoen ikusteko. Morroi gisa baldin badago 0 itzuliko du; bestela, 1.
- **AT + PSWD** //Gailu bat moduluarekin konektatzean sartu beharreko pasahitza zein den jakiteko
- **AT + UART** // Mikrokontrolagailuarekin duen komunikazio konfigurazioa zein den jakiteko. Lehenetsitakoa 9600, 0, 0 da.

Aipatutako komandoak parametroen kontsultarako dira, baina parametro hauek aldatzeko erabili beharrekoak ere berdinak dira; egin behar den bakarra komando bakoitzaren jarraian "=" ikurra eta balio berria jartzea da. Hau da, pasahitza aldatu nahi bada, esaterako, **AT + PSWD = 1234** komandoa exekutatuko beharko da; eta berdin beste komando guztiekin.

Komunikazio moduan, moduluak modu gardenean lan egiten du; hau da, moduluak jasotako datu guztiak Romeo mikrokontrolagailuari bidaltzen dizkio, eta alderantziz.

Moduluak berak bi etengailu (LED off, AT Mode) eta bi LED (STATE, LINK) ditu gainaldean, 15.irudian ikus daitekeen bezalaxe. Lehenik eta behin, LED horiek zeren adierazgarri diren azalduko da. STATE LEDak argia bi modutan egin dezake: *flicker* modua, konexio bilaketa egoera adierazten duena; edota *flashing* modua, modulua konexio egoeran dagoela adierazteko. Beste LEDari dagokionez, LINK LEDa, behin gailu batekin konektatzen denean piztuko da.

Etengailuei dagokienez, LED off etengailua LINK LEDa ez pizteko erabiltzen da; hau da etengailu hori 1 posizioan jarriz gero, LINK LEDa ez da piztuko; honen helburua bateria aurreztea izanik. Bestalde, AT Mode etengailuari dagokionez, hau moduluaren funtzionamendu modua aukeratzeko erabiltzen da; hau da, etengailua 1 posizioan badago, modulua konfigurazio moduan egongo da AT komandoak jaso eta prozesatzeko prest; 0 posizioan badago aldiz, komunikazio moduan egongo da.

Zehaztapenak

Bluetooth txipa: CSR BC417143

Bluetooth protokoloa: Bluetooth Specification v2.0 + EDR

Eragiketa maiztasuna: 2.4 ~ 2.48 GHz

Modulazioa: GFSK (Gaussian Frequency Shift Keying)

Transmisio distantzia: 20 – 30 m espazio librean

Segurtasun ezaugarriak: Autentikatzea eta zifratzea

Onartutako perfila: Bluetooth serial port

Serie portuaren komunikazio abiadura: 4800 - 1382400 / N / 8 / 1 Lehenetsitakoa: 9600

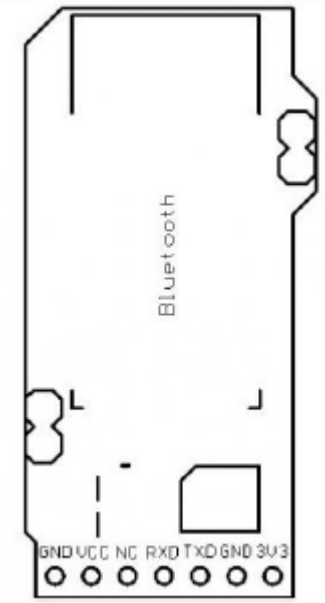
Sarrerako tentsioa: 3.5 V – 8 V

Tamaina: 40 x 20 x 13 mm

Azkenik, moduluaren ezaugarriekin amaitzeko, honek dituen pin-ak zein diren zehaztea falta da. 16.irudian ikus daitekeen bezala, honako hauek dira moduluak dituen pinak: GND, VCC, NC, RXC, TXD, GND eta 3V3. Nahiz eta 7 hanka izan, hauetatik 4 erabiliko ditut: GND eta VCC elikatzeko, eta RXD eta TXD datuak jaso eta bidaltzeko.

PIN	ESANAHIA
GND	Lurra
VCC	3.5 V – 8 V korronte zuzena
RXD	Datuak jasotzeko
TXD	Datuak bidaltzeko

6.taula: Bluetooth moduluaren pinak

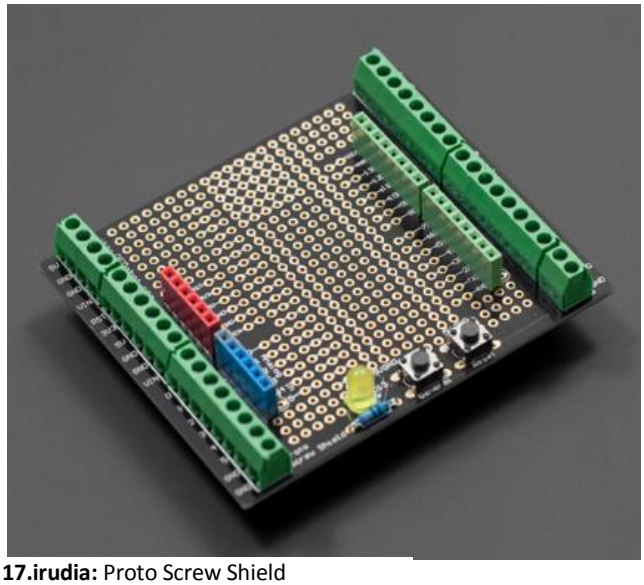


16.irudia: Bluetooth moduluaren pinak

3.3.3 Proto Screw Shield

Shield hau hasieran aipatutako 4 erabilienetako bat da eta prototipoak egiteko erabiltzen da. Honi esker modu erraz eta eroso batean erabiltzaileak zirkuitu txikiak egin ditzake kableak, eztainua eta soldagailua erabiliz.

Honetaz gain LED bat eta bi botoi ere baditu (*reset* botoia eta orokorra), mikrokontrolagailuak dituenekin bat eginik.

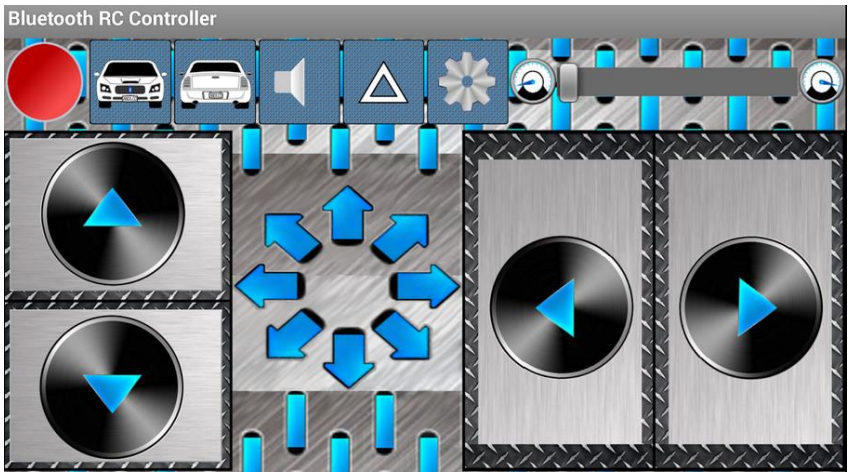


17.irudia: Proto Screw Shield

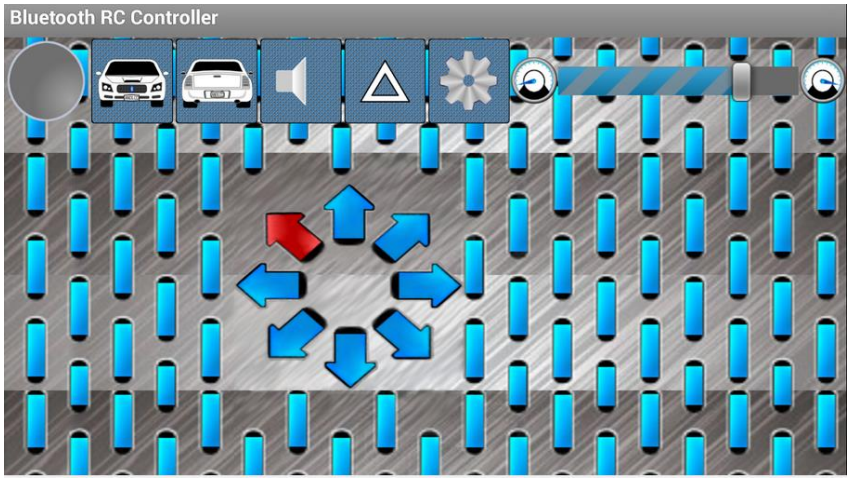
3.3.4 Bluetooth RC Car

Robota *smartphone* bat erabiliz kontrolatu ahal izateko ezinbestekoa da aplikazio egoki bat izatea eskura robotarekin bluetooth bidez komunikatzeko, kasu honetan Androiderako aplikazio bat. Bila aritu ondoren, hainbat aplikazio aurkitu dira bluetooth bidez Arduinorekin komunikatzeko; eta hauek banan-banan aztertu ondoren, proiektu honetarako egokiena *Bluetooth RC Car* (Andi.Co., 2014) dela ondorioztatu da. Aplikazio hau Play Store-n eskuragarri dago eta doakoa da.

Aplikazioa oso erraza eta sinplea da bai konfiguratzen bai eta erabiltzen ere. Auto bat kontrolatzeko dago diseinatua eta beraz, atzera, aurrera, eskuinera eta ezkerrera gisako botoiak ditu. Aipatzekoa da bi modutan erabil daitekeela: batetik botoiak zapalduz, edota bestetik, mugikorreko azelerometroak erabilita, mugikorra mugituz. Hona hemen bi aukera hauek:



18.irudia: RC Car. Botoien bidezko kontrola



19.irudia: RC Car. Azelerometro bidezko kontrola

Aplikazioa irekitzean, zuzenean mugikorreko bluetooth-a aktibatzen da eta ondoren, aplikazioko konfigurazio botoia sakatu eta *connect* aukera sakatu behar da. Bluetooth moduluarekin konektatzeko lehen aldian, pasahitza eskatuko du, baina hortik aurrera zuzenean konektatuko da.

Aplikazioaren funtzionamendua oso erraza da: botoi bat sakatzen den aldiko, bluetooth moduluari komando bat bidaltzen dio (botoiari dagokiona). Botoi ugari dituen arren, proiektu honetan ez dira denak erabili. Hona hemen kasu honetarako interesgarriak diren botoiak eta hauek sakatzean bidaltzen diren komandoak:

Botoia	*Komandoa*
Aurrera	F
Atzera	B
Ezkerrera	L
Eskuinera	R
Stop	S
Aurrera ezkerrera	G
Aurrera eskuinera	I
Atzera ezkerrera	H
Atzera eskuinera	J
Amaitu	D
Abiadura	0-9, q

7.taula: RC Car aplikazioaren komandoak

Behin hau jakinda, komandoak irakurri eta hauen arabera aginduak exekutatuko dituen programa egitea ez da lan zaila. Aipatu beharra dago, bai botoien bidez eta bai

azelerometroa erabiliz, kasu bakoitzean bidaltzen diren komandoak berdinak direla; beraz, programa bera erabiliz bi modutan erabil daiteke aplikazioa.

Bestalde, aplikazioak komandoak zein maiztasunekin bidaliko diren konfiguratzeko aukera ematen du, honetarako bi aukera eskainiz: 50 ms-an behin komando bat bidaltzea edota aldaketa dagoenean bakarrik, hots, botoi bat sakatzen denean edota askatzen denean. Kasu honetarako egokiena eta eraginkorrena bigarren aukera dela erabaki denez, modu honetan erabili da aplikazioa. Beraz, mikrokontrolagailuari komandoak erabiltzaileak botoi bat sakatu edo askatzen duenean bidaltzen zaizkio; modu honetan ez dira alferrikako komandoak bidaltzen aldaketarik ez badago.

3.3.5 Ultrasoinu sentsorea

Robota Bluetooth RC Car aplikaziotik jasotako aginduen arabera mugitzen den arren, segurtasun neurri bat hartzea ezinbestekoa izan da. Honetarako, robotak aurrealdean ultrasoinu sentsore bat dauka eta oztoporen batera gehiegi gerturatzean, bluetooth-eko aginduei gainjarri eta motorrak geldiarazten ditu. Beraz, hauxe da sentsore honen funtzioa.

Ultrasoinu sentsoreak objektuekiko distantziak neurtzeko sentsoreak dira, eta hegaldi denboraren menpeko sentsore gisa sailkatzen dira, soinu uhinak igortzen dituzten hegaldi denboraren menpeko sentsoreak, hain zuzen ere. Ultrasoinuek objektuekiko distantziak neurtzeko honako hau egiten dute: soinu uhin bat igortzen dute eta uhin honek objektu baten aurka talka egitean, errebotatu eta berriro ere ultrasoinurantz joaten da. Ultrasoinuak uhinaren oihartzuna jasotzean, uhin hori bidali duenetik jaso arte igarotako denborarekin eta uhinaren hedapen abiadura ezagututa, talka egin duen objektuarekiko distantzia neurtzen du.

$$d = \frac{Vs\ x\ T}{2}$$

Proiektu honetan erabilitako ultrasoinu sentsorea DFRobot etxeko URM37 da. Hona hemen zehaztapen nagusiak:

- Elikadura: 5 V
- Antzeman dezakeen distantzia tartea: 4 cm – 5 m
- Bereizmena: 1 cm
- Interfazeak: RS232 (TTL), PWM
- Serbomotore bat kontrolatzeko aukera
- Lan egiteko moduak: Serieko kontrol pasibo modua, modu autonomoa edo *trigger* modua

20.irudia: Ultrasoinu sentsorea

Behin sentsorearen ezaugarri nagusiak ezagututa, hurrengo urratsa sentsorearen pin-ak eta *jumper*-ak zein diren eta zein funtzio duten ezagutzea da. Hona hemen sentsoreak dituen pin-en zerrenda:

TXD: RS232,TTL komunikaziorako (*Transmitter*)

RXD: RS232,TTL komunikaziorako (*Receiver*)

NC: *Not connected*

COMP/TRIG: *Trigger* funtzioa egiten duen pin-a. Objektuekiko distantzia aurredefinitutako balio baten berdina edo txikiagoa denean HIGH egoeratik LOW egoerara pasatzen da.

MOTO: Serbomotore bat kontrolatzeko irteera pin-a

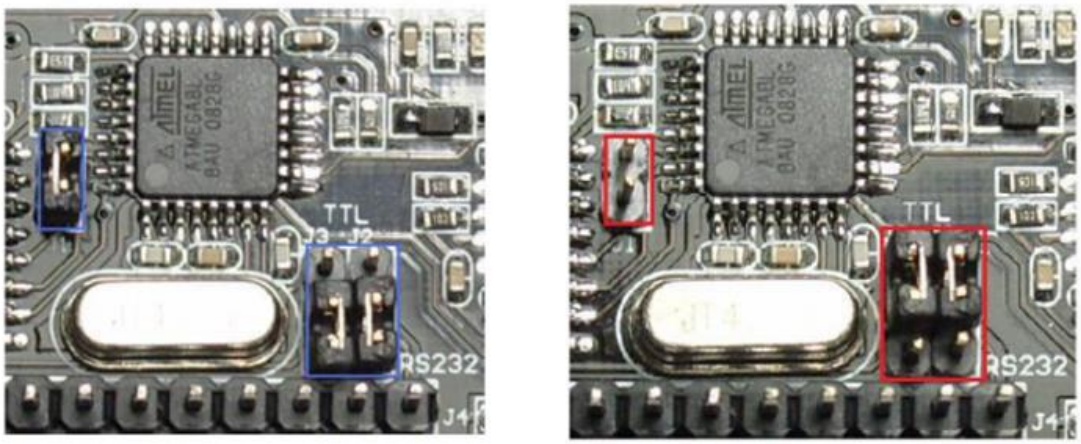
PWM: PWM irteera

RST: Reset

GND: Lurra

5V: 5V

Jumper-ei dagokienez, hiru *jumper* ditu eta hauek RS232 edo TTL komunikazioa finkatzeko erabiltzen dira. 21.irudian ikus daitezke zein diren bi aukerak:



RS232 modua

TTL modua

21.irudia: Ultrasoinuaren jumper-ak komunikazio modua aukeratzeko

Aurrerago aipatuko den arren, Romeo mikrokontrolagailuak ez du aukerarik harien bidez programatzeko eta beraz, ezin dira atazak paralelizatu. Hau dela eta, aukera onena ultrasoinu sentsorea etenen bidez erabiltzea da; hainbat denboraz behin irakurketak egiten aritu beharrean. Beraz, proiektuan ultrasoinuaren hiru pin erabiliko dira: GND, 5V eta COMP/TRIG.

Jumper-ei dagokienez, TTL modua erabili behar da kasu honetarako.

Proiektu honetarako baliagarriak diren bi funtzionamendu moduak honako hauek dira:

1. **Serieko kontrol pasibo modua**

   Modu honetan sentsorea serie lerrotik komando bat iritsi zain egoten da. Komando bat jasotzen duen aldiko, hau prozesatu eta dagokion emaitza itzultzen du. Amaitzean, berriro ere zain geratuko da.

   Lan egiteko modu hau sentsorea komando bidez konfiguratzeko erabili da.

2. **Modu autonomoa/*trigger* modua**

   Modu honetan sentsoreak distantzia neurketa bat egiten du 25 ms-an behin. Sentsoreak EEPROM memorian balio bat izango du, aurrez definitu dena distantzia minimo gisa. Beraz, irakurketa bat egiten duen aldiko, irakurritako distantzia aurredefinitutako balio horrekin alderatzen du eta berdina edo txikiagoa bada, COMP/TRIG pin-eko balioa aldatzen du HIGH baliotik LOW baliora. COMP/TRIG pin hau Romeo-ren eten pin batean konektatzen bada, eten gisa erabil daiteke; aldaketa bat dagoen aldiko etena sortuz.

Funtzionamendu modua zehazteko ere komandoak erabiliko dira. Jarraian aztertuko dugu erabil daitezkeen komandoak zein diren eta hauen formatua zein den.

Komandoak

Ultrasoinu sentsorea konfiguratzeko komandoak erabili behar dira, eta komando hauek serie lerrotik bidali behar dira. Honako hau da komandoen egitura:

Komando_id + data 0 + data 1 + sum

8.taula: Ultrasoinua konfiguratzeko komando egitura

Komando_id: exekutatu nahi den komandoaren identifikadorea

Data0, data1: Komandoaren arabera, komandoari dagozkion parametroak

Sum: Balio hau errore kontrolerako erabiltzen da (*checksum* gisa) eta komando_id + data0 + data1 batuketaren emaitzaren azken 8 bitak dira.

Komandoaren osagai bakoitza 8 bitekoa da eta hamaseitarrez adierazi behar da. 9.taulan ikus daitezke erabil daitezkeen komando desberdinak zein diren eta hauen funtzioa eta parametroak zein diren:

Komandoa	*Funtzioa*
0x11 + NC + NC + SUM	16 bit-eko tenperatura irakurketa ahalbidetu
0x22 + Degree + NC + SUM	16 bit-eko distantzia irakurketa ahalbidetu
0x33 + Add + NC + SUM	EEPROM-eko Add helbidea irakurri
0x44 + Add + Data + SUM	EEPROM-eko Add helbidean idatzi

9.taula: Ultrasoinuaren konfigurazio komandoak

Trigger modua edo modu autonomoa eta aurredefinitutako muga balioa finkatzeko EEPROM memorian dagokion posizioan idatzi behar da; beraz, taulako azken komandoa erabili behar da honetarako. Hona hemen zein helbidetan idatz daitekeen eta zertarako:

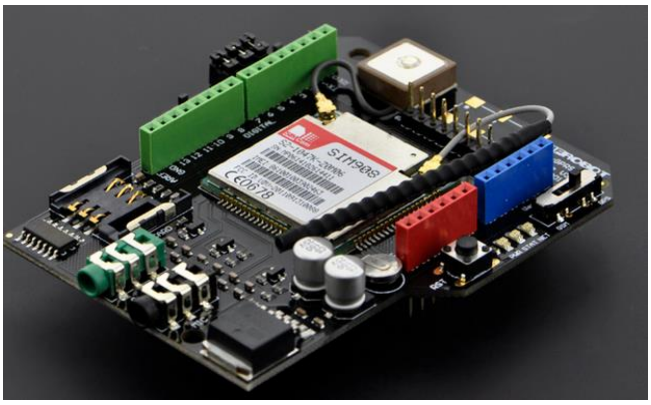
Helbideak	*Funtzioa*
0x00	*Trigger* balioaren pisu txikieneko bytea
0x01	*Trigger* balioaren pisu handieneko bytea
0x02	Lan egiteko modua (autonomoa/pasiboa)

10.taula: Ultrasoinua konfiguratzeko EEPROMeko helbideak

3.3.6 GPS/GSM/GPRS modulua

Robotaren uneko posizio globala zein den jakin ahal izateko DFRobot etxeko GPS/GSM/GPRS modulua erabili da. Nahiz eta proiektu honetan GPS moduan soilik erabili, aipatzekoa da modulu hau oso ahaltsua dela eta GSM eta GPRS teknologiak ere integratuta dakartzala.

Lehenik eta behin, azaldu dezagun, gainetik bada ere, zer diren aipatutako teknologia hauek: GPS, GSM eta GPRS.



22.irudia: GPS/GSM/GPRS modulua

GPS

GPS ingeleseko siglek *Global Positioning System* adierazten dute; euskaraz, Kokatze-Sistema Orokorra esango diogu. Sistema honen bidez, leku jakin baten posizio zehatza zein den jakin daiteke.

Informazio hau eskuratzeko sateliteak erabiltzen dira. Sateliteek etengabe mezuak bidaltzen dituzte beren kokapena eta mezua igorri den ordua zein den adieraziz. GPS gailuen bidez satelite desberdinetako mezuak jasotzen dira eta mezu hauek gordetzen duten informazioa erabiliz, GPS gailuaren posizioa zein den kalkulatzen dute. Beraz, GPSak ere hegaldi denborako sentsore gisa definitzen dira.

Aipatzekoa da GPSak kanpo-inguruneko robotentzat direla baliagarriak (barruan ez dute funtzionatzen) eta ez dela guztiz zehatza ematen duen balioa; hau da, GPSak ere errorea du.

GSM

Ingelesez *Global System for Mobile Communications* izenez ezaguna den teknologia hau telefonia mugikorrerako sistema estandar bat da. GSM bezero batek honako ekintza hauek guztiak egin ditzake: ordenagailu batera konektatu eta mezularitza elektronikoa erabiliz mezuak bidali eta jaso, Interneten nabigatu, fax-ak bidali eta baita SMS zerbitzua erabili ere.

GSM, transmisio abiadura eta beste ezaugarri batzuk direla eta, bigarren belaunaldiko estandar gisa hartzen da (2G).

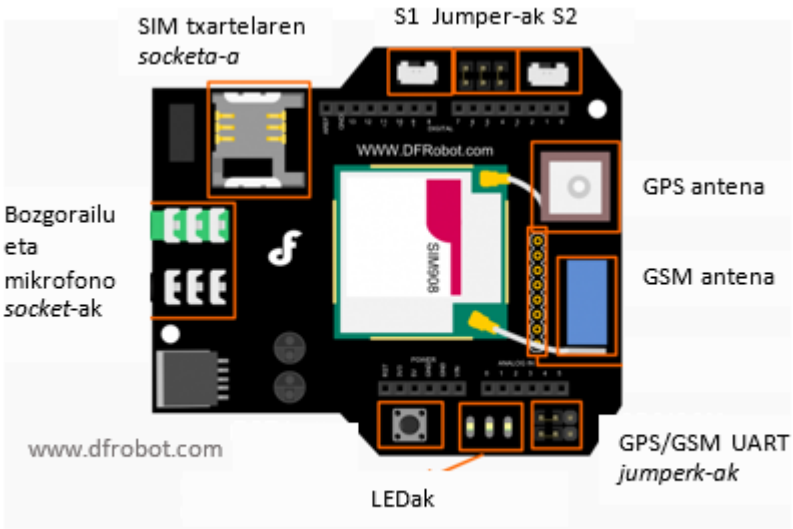
GPRS

Sigla hauek *General Packet Radio Service* adierazten dute eta GSM sistemaren hedapen gisa sortu zuten 80. hamarkadan, datu transmisioa paketeen kommutazio bidez ahalbidetuz.

Behin modulu honek eskaintzen dituen zerbitzuak zein diren azalduta, moduluaren ezaugarriekin jarraituko dugu. Modulu hau Bluetooth Shield-a bezalaxe, AT komando bidez kontrolatzen da eta komando hauek zein diren jakiteko *SIM908 AT Command Manual_V1_01* (SIMCom., 2011) eskuliburura jo da.

Modulu honek kontrolagailura edota zuzenean ordenagailura konektatzeko aukera eskaintzen du GSM eta GPS funtzioak erabili ahal izateko eta SMD antena ahaltsua integraturik dakar. Elikadurari dagokionez, 7 – 12 V inguruko elikadura behar du ongi lan egiteko.

23.irudian modu argian ikus daiteke modulu honek dituen osagaiak zein diren:



23.irudia: GPS/GSM/GPRS moduluaren osagaiak

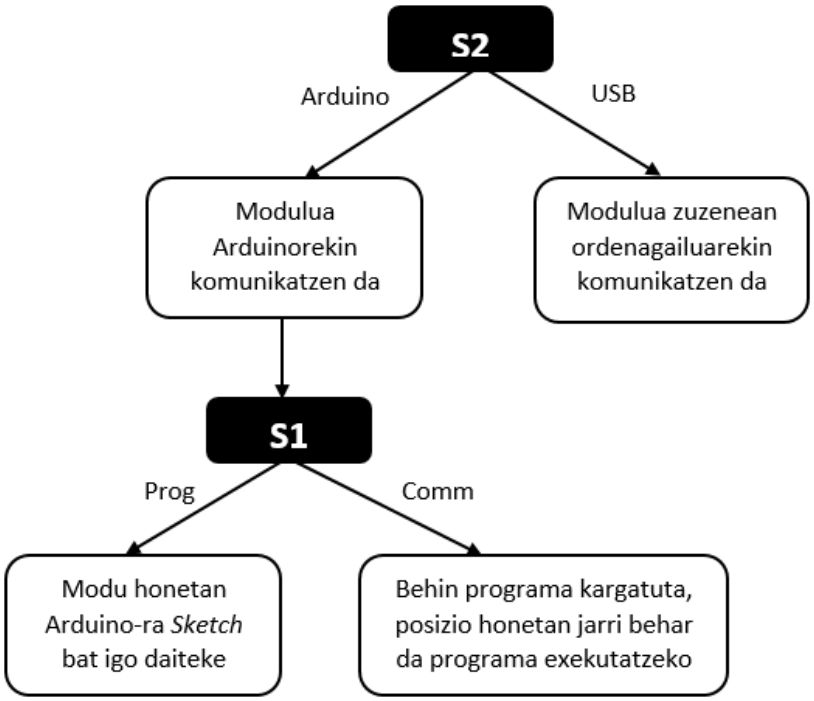
Lehenik eta behin oso garrantzitsua da modulu honek dituen *jumper* eta *switch*-ak zein diren eta zertarako diren argitzea. Modulu honek 2 switch (S1, S2) eta 2 jumper multzo (J1 eta UART SELECT) ditu.

S2 *switch*-aren bi posizioak USB eta Arduino dira. *Switch*-a USB posizioan baldin badago, USB bidez mikrokontrolagailua ordenagailura konektatzean, ordenagailutik modulu honetara zuzeneko serieko komunikazioa ahalbidetzen da; ordenagailua eta GPS/GSM/GPRS modulua zuzenean komunikatzen dira. Modu honetan mikrokontrolagailuko kodeak ez du inongo eraginik moduluan eta mikrokontrolagailua ordenagailu eta moduluaren komunikazioko bitartekari soil izatera pasatzen da. Beraz, *switch*-a posizio horretan dagoelarik, serieko terminal bat erabiliz moduluarekin komunikatu gaitezke AT komandoak bidaliz. (Honetarako Arduino softwarearen serieko monitorea ere erabil daiteke).

S2 *switch*-a USB posizioan badago, S1 *switch*-aren posizioa zein den berdin du, *switch* honen posizioak funtzionamenduan eragina izango du baldin eta S2 *switch*-a Arduino posizioan badago. Arduino posizioan egonik, modulua ordenagailuarekin komunikatu beharrean *Romeo* mikrokontrolagailuarekin komunikatuko da; beraz, mikrokontrolagailuan kargatutako programak eragina izango du moduluan. Kasu honetan, beraz, S1 *switch*-aren posizioak garrantzia izango du.

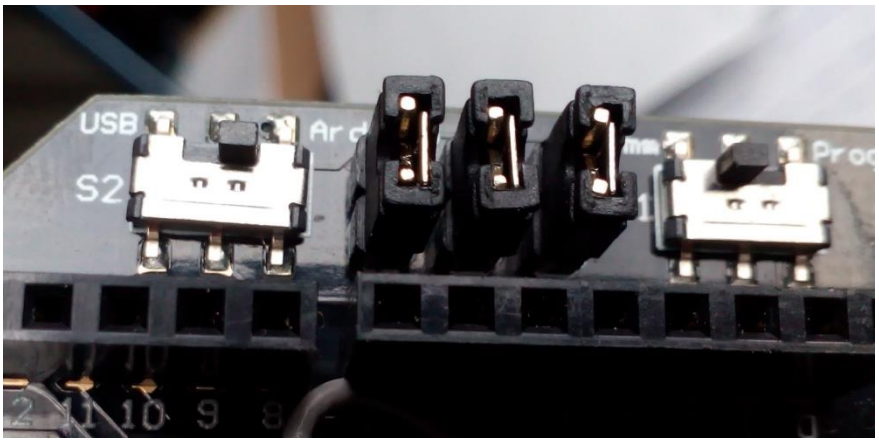
S1 *switch* honek bi posizio hauek ditu: Comm eta Prog. GPS/GSM/GPRS moduluak mikrokontrolagailutik datuak jaso nahiz bidaltzeko, RX(0) eta TX(1) hankatxoak erabiltzen ditu; programa mikrokontrolagailuan kargatzeko erabiltzen diren berberak. Honen ondorioz, arazoak sortzen dira mikrokontrolagailuan programak kargatzean. Arazo hau ekiditeko *switch* hau erabiltzen da. Mikrokontrolagailuan programa berri bat kargatu nahi badugu, *switch*-a Prog posizioan jarri behar da; eta behin kargatu ondoren, exekutatu nahi denean, Comm posizioan.

Orain artekoa hobeto ulertze aldera, hona hemen eskema bidezko azalpena:



24.irudia: GPS/GSM/GPRS moduluaren switch-en azalpena

Aipatutako *jumper* multzoei dagokienez, GPS/GSM funtzionamendu moduekin dute zerikusia. Modulua ezin da une berean GSM eta GPS gisa funtzionatzen aritu; hau da, GPS moduan edo GSM moduan egongo da, baina ez biak batera. Uneko funtzionamendu modua aukeratzeko bi modutara egin daiteke: UART *select* jumper multzoa erabiliz; edota programa bidez: 3, 4 eta 5 hankatxoak erabiliz.

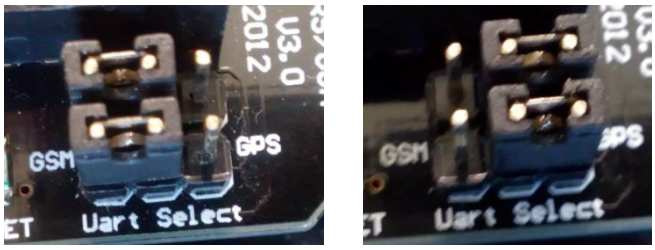


25.irudia: GPS/GSM/GPRS moduluaren J1 jumper multzoa eta switch-ak

25.irudian agertzen den *jumper* multzoa J1 da. Funtzionamendu modua programa bidez egin nahi bada, *jumper* hauek jarrita utziko dira, modu honetan mikrokontrolagailuaren 3, 4 eta 5 hankatxoak modulu honekin konektatzen dira eta programa bidez aukeratuko da funtzionamendu modua (GPS/GSM). Aldiz,

funtzionamendu modua UART *select jumper*-ekin egin nahi bada, irudiko *jumper* hauek denak kendu behar dira, eta beraz, 3, 4 eta 5 hankatxoek ez dute inongo eraginik izango moduluan eta libre geratuko dira. Beraz, ez dago posizio desberdinik: hiru *jumper*-ak kenduta edo hirurak jarrita.

UART *select jumper*-ak, beraz, funtzionamendu modua aukeratzeko erabil daitezke, eta 2 posizio posible daude: GSM (ezkerrean) eta GPS (eskuinean). Funtzionamendu modua zehazteko *jumper* hauek erabili nahi ez badira (programa bidez egingo delako), bertako 2 *jumper*-ak kendu behar dira.



GSM GPS

26.irudia: GPS/GSM/GPRS moduluaren Uart Select jumper-ak

Funtzionamendu modua 3, 4 eta 5 hankatxoen bidez egiteak abantaila nabarmen bat du: programan edozein unetan GSM modutik GPS modura pasa gaitezke, edo alderantziz. *Jumper* bidez egin nahi bada, *jumper*-ak aldatu beharra daude, eta hau proiektuaren arabera nahiko deserosoa suerta liteke.

Behin argi izanik moduluak dituen *switch* eta *jumper*-ak zertarako diren eta mikrokontrolagailuarekin komunikatzeko zein hankatxo erabiltzen dituen, modulu honen AT komandoak zein diren eta zertarako diren jakitea soilik falta da. Honen inguruko informazio osoa modulu honen AT komandoen eskuliburuan aurkitu daiteke.

Azkenik, aipatu beharra dago, GPSak ongi funtzionatu ahal izateko, kanpoan egon behar duela sateliteen mezuak jaso ahal izateko; barruan ez du mezurik jasotzen.

3.3.7 Wifi *shield*

Nahiz eta orain arte Wifi bidezko komunikazioaren inguruan ez den ezer aipatu, robotak erabili behar izan duen gailu bat izan da proiektuaren oinarrizko kalitate maila lortzeko. Irismenean aipatutako helburuetako bat robotaren kokapena zein den jakitea da, eta hori GPS bidez lortu da. Baina, alferrikakoa da robotak uneko posizioa kalkulatzea, ondoren posizio horren berri emateko gai ez bada. Beraz, derrigorrezkoa izan da robotak haririk gabeko konektibitatea izatea, erabiltzaileari informazioa bidaltzea ahalbidetzeko.

Egia da bluetooth bidezko komunikazioa baduela, baina, erabilitako aplikazioak datuak bidaltzeko soilik balio du, ez jasotzeko. Honetaz gain, bluetooth modulua aldiko gailu batekin egon liteke konektatuta, beraz, ezinezkoa da mugikor baten aginduak bluetooth bidez jasotzea eta aldi berean beste gailu batekin komunikatzea datuak bidaltzeko.

Hau horrela izanik, wifi bidezko komunikaziora jotzea erabaki da. Honela, GPS bidez lortutako balioak erabiltzaileari erakustea lortu da; eta gainera, robotari kalitate dimentsio bat gehitu zaio, GPSaren datuak bidaltzeaz gain, uneko informazioa ere bidaltzeko aukera ematen duelako. Honela, erabiltzaileak robotaren uneko egoeraren informazio osoa izan dezake eta robotak haririk gabeko komunikaziorako beste aukera bat eskaintzen du.

Wifi-aren erabileraren zergatiak azaldu ondoren, jo dezagun honetarako erabili den osagaia aztertzera: DFRobot Wifi Shield V2.2.

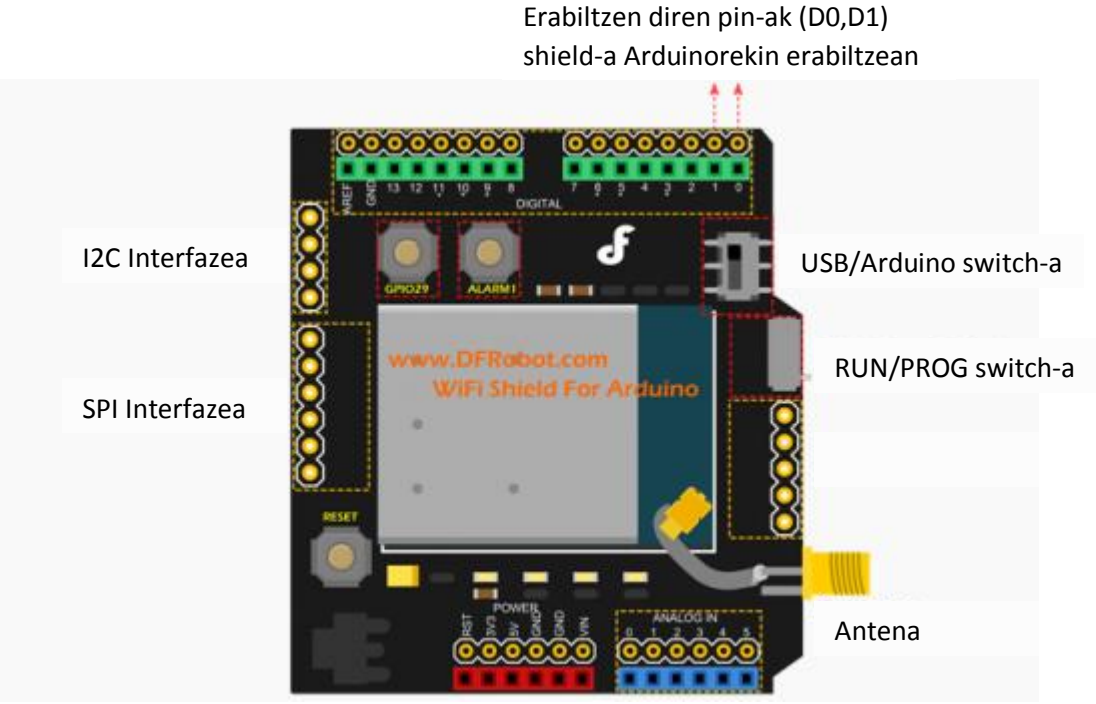
Dfrobot-en *shield* honek WizFi210 modulua erabiltzen du; eta GPSaren kasuan bezalaxe, AT komandoen bidez konfiguratu eta erabiltzen da. Modulu hau zubi gisa erabiltzen da TTL serie portuko komunikazio eta IEEE802.11b/g/n komunikazioen artean. Beraz, modulu hau TTL serie portuak dituen edozein gailuri gehi dakioke eta haririk gabeko komunikazioa ahalbidetu.

Modulu honek komunikazio protokolo eta zifratze algoritmo desberdinak ditu integraturik. Hona hemen *shield* honen ezaugarri nagusiak:

- Wifi txipa: WizFi210
- Irrati protokoloa: IEEE 802.11b/g/n bateragarria
- Komunikazio abiadura: 11, 5.5, 2, 1 Mbps (IEEE 802.11b)
- Modulazioa: DSSS eta CCK
- Antena aukerak: Txipean integratutako antena erabil daiteke edota kanpo antena bat gehi dakioke U.FL konektorearen bidez
- Sare protokoloak: UDP, TCP/IP (IPv4), DHCP, ARP, DNS, HTTP/HTTPS bezeroa eta zerbitzaria
- Kontsumo baxua
- Segurtasun protokoloak: WEP, WPA/WPA2-PSK, Enterprise, EAP-FAST, EAP-TLS, EAP-TTLS, PEAP
- Elikadura: 3.3 V
- Sarrera/Irteera interfazeak: : UART, SPI(*), I2C(*), WAKE, ALARM, GPIOs

27.irudia: Wifi shield

Hona hemen *shield* honen eskema:



28.irudia: Wifi shield-aren eskema

GPS/GSM/GPRS moduluaren kasuan bezalaxe, *shield* hau kontrolagailuarekin edota zuzenean ordenagailuarekin komunikatu daiteke AT komandoak erabiliz. Honetarako, aurreko moduluak bezala, bi switch ditu *shield* honek: USB/Arduino switch-a eta Prog/Run switch-a. Erabilera ere berdina da. Shield-a ordenagailuarekin zuzenean komunikatu nahi denean, lehenengo switch-a USB posizioan jarri behar da eta bigarren switch-a berdin du zein posiziotan dagoen. Arduinorekin komunikatzea nahi denean, aldiz, lehenengo switch-a Arduino posizioan jarri behar da eta bigarren switch-ari dagokionez, *sketch* bat kargatu nahi denean Prog posizioan jarri behar da eta *sketch*-aren exekuziorako, berriz, Run posizioan.

29.irudia: Wifi shield-aren switch-ak

LEDei dagokienez, *shield* honek 4 LED ditu: POWER, STW, SDR eta ASSOC. LED hauen bidez, *shield*-aren uneko egoeraren inguruko informazioa ematen da eta hau oso lagungarri suertatzen da lan egiterako orduan. Azter dezagun, beraz, zeren adierazgarri den bakoitza:

- POWER: *Shield*-a elikatuta dagoen bitartean piztuta egoten da.
- STW, ASSOC: *Shield*-a router batera konektatzen denean pizten dira (piztuta mantentzen dira konexioa galdu arte)
- SDR: *Shield*-ak daturen bat jaso edo bidaltzen duenean pizten da (piztu eta itzali)

Azkenik, falta den bakarra *shield*-a nola konfiguratu eta erabili jakitea da. Honetarako, hasieran aipatu den moduan, AT komandoak erabili behar dira; eta GPSaren kasuan bezalaxe, produkturen webgunean aurki daiteke AT komando hauen eskuliburua: *WizFi210/220 User Manual* (WIZnet Co. , 2011).

4. GARAPENA

4.1 Motorren kontrola

Romeo-All in one mikrokontrolagailuaren ezaugarriak aztertzerakoan ikusi den bezala, mikrokontrolagailu honek motorrak kontrolatu ahal izateko bi kanal eskaintzen ditu, eta motorren kontrola pultsu bidezko modulazioaren bidez burutzen da. Kanal hauetako bakoitzak bi terminal ditu, eta beraz, terminal horietara konektatu behar dira motorrak.

Hurrengo pausoa beraz, programa bidez motor hauek mugitzea lortzea izan da; honetarako, beharrezkoa da motorren kontrolerako kanalei dagozkien pin-ak zein diren jakitea, hortaz, mikrokontrolagailuaren pin-en eskemara joko dugu.

1.KANALA		2.KANALA	
M1	E1	M2	E2
4	5	7	6

11.taula: Mikrokontrolagailuaren motorren kontrolerako pin-ak

Kanal bakoitzaren M terminala, motorraren polaritatea finkatzeko erabiltzen da. Beraz, terminal honetan balio digital bat idatzi behar da: HIGH, aurreraka biratzeko eta LOW atzeraka biratzeko. E terminalean berriz, 0-tik 255-era bitarteko balio bat ezarri behar da, eta balio honen bidez, motorren mugimendu errotazionalaren abiadura finkatzen da.

Beraz, motorrak nora eta nola konektatu eta programatu ahal izateko erabili beharreko pin-ak ezagututa, hau praktikan jartzea soilik falta da. Baina, urrats honetan arazo batekin topo egin da; izan ere, Romeo mikrokontrolagailuak gehienez ere, 5 V-eko tentsioa eman diezaioke motor bakoitzari, eta motorren ezaugarrietan ikusi den bezala, motor hauek 12 V inguruko tentsioarekin funtzionatzen dute.

Beraz, puntu honetan eskura dauden osagaiak ez dira nahikoa motorrak mugiarazteko. Hau lortu ahal izateko, Romeo mikrokontrolagailuari Motor Shield osagaia gehitu zaio, hau indar altua behar duten motorrak kontrolatu ahal izateko diseinatua baitago. *Shield* hau erabiltzean, programan ez da ezer aldatu behar, pin-ak berberak izaten jarraitzen baitute, noski. Egin beharreko gauza bakarra, motorrak *shield* honetara konektatzea da eta *shield* honi kanpo elikadura jartzea.

Lanerako erabilitako materialen atalean aipatu bezala, *shield* hau hiru modutara elika daiteke. Motorrek tentsio altua behar dutenez, *shield* honetara konektatutako elikadurak *shield-a* soilik elikatzen du; eta beraz, Romeo mikrokontrolagailua elikatzeko

beste elikadura iturri independente bat erabili da. Kontutan izan behar da, *jumper*-ak dagokien moduan jarri behar direla. Behin dena prest izanda, jada posible izan da motorrak mugiaraztea.

12.taulan, tentsio mailekin sortutako arazoa konpontzeko hartu diren tentsio neurketa batzuk ikus ditzakegu, arazoa zein zen ezagutzeko lagungarri izan direnak. Neurketa hauek egiterakoan, motorrei abiadura maximoa ezarri zaie (255).

NEURKETA	*NEURRIA*
Romeok motorrei ematen dien tentsio maximoa	3.8 V
Motor Shield-ak, Romeoren bidez elikatuta, eman dezakeen tentsio maximoa	3.8 V
Motorrek mugitzeko behar duten tentsioa	5 V – 12 V
12 V-eko elikadurarekin Motor Shield-ak ematen duen tentsioa	10,15 V

12.taula: Motorren kontrolerako tentsio neurketak

Beraz, modu honetan 2 kanalen bidez 2 motor modu independentean mugitzea lortu da, hasierako arazoak konponduz. Nahiz eta 12 V-eko elikadurarekin 255eko balioa emanez motorrak mugitzen diren, oso geldi biratzen dute; eta proba hauek gurpilak airean daudela egin dira. Lurrean jartzean, lurrak eragiten duen marruskadura eta robotaren pisua dela eta ia ez dira bat ere mugitzen.

Hau horrela izanik, azkenean, 14,4 V-eko bateria bat erabili da. Eta honela, aipatutako arazo hau konpontzen da.

4.2 Urruneko kontrola bluetooth bidez

Robota urrunetik kontrolatu ahal izateko, bluetooth modulua erabili da, modulu honen inguruko zehaztapen guztiak Laneko Materialak izeneko atalean azaldutakoak dira.

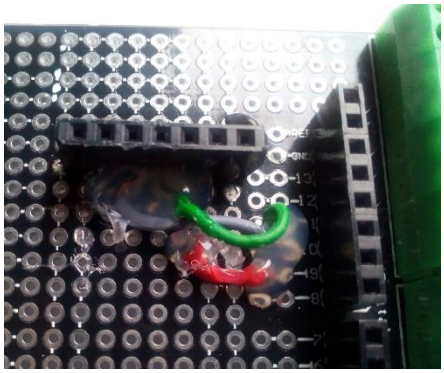
4.2.1 Konexioak egin

Eman beharreko lehenengo pausoa moduluaren hankei dagozkien konexioak egitea izan da; aipatu bezala, moduluaren lau hankatxo erabili dira: GND, VCC, RXD eta TXD. GND hankatxoa mikrokontrolagailuko GND portuetako batekin konektatu da eta

beste hirurak mikrokontrolagailuan libre dauden hiru pin digitalekin: 9, 10 eta 11. VCC ez da mikrokontrolagailuko VCC portura konektatu, 9.pin-era baizik; modu honetan, programatik piztu eta itzal daitekeelako bluetooth modulua. Bestalde, pentsa daiteke logikoena RXD eta TXD hankak mikrokontrolagailuko TX eta RX hanketara konektatzea dela. Aukera honen inguruan ere hausnartu da, baina honek arazo bat dakar: izan ere, mikrokontrolagailua serie portutik USB bidez ordenagailura konektatzean ere lerro hauek erabiltzen dira komunikaziorako; beraz, ezin dira biak batera erabili (ordenagailua eta bluetooth modulua). Programen exekuzioan ordenagailurako konexioa ezinbestekoa suertatu da *debugger* gisa erabiltzeko, programaren exekuzioaren eta uneko egoeraren berri izateko.

Hau dela eta, bluetooth moduluaren hankak beste bi pinetara konektatzea erabaki da eta ondoren, SoftwareSerial izeneko liburutegia erabili da aukeratutako bi hankatxo hauek serie portu bezala erabili ahal izateko. Liburutegi honek serie komunikazioa ahalbidetzen du mikrokontrolagailuaren edozein bi pin digital hartuz, software bidez funtzionalitate hau emulatuz.

Modulu hau ezin da zuzenean mikrokontrolagailura konektatu, pin bakoitzei dagokien portuak ez daudelako ordena berean eta elkarren ondoan mikrokontrolagailuan. Beraz, hau egin ahal izateko, DFRobot-eko Proto Screw Shield-a erabili behar izan da. Beraz, soldatzailea, eztainua eta kableak erabiliz beharrezkoak ziren konexioak egin dira. Egindakoa 30.irudian ikus daiteke. Behin konexioak eginda, *shield* hau mikrokontrolagailuaren gainean jarri, bluetooth modulua kokatu eta hurrengo urratsa emateko moduan izan gara.



30.irudia: Proto Screw Shield-ean egindako soldadurak

4.2.2 Modulua konfiguratu

Behin modulua zuzen konektaturik, hau konfiguratzea izan da hurrengo urratsa. Konfigurazio hau egin ahal izateko, modulua AT moduan jarri da (AT mode etengailua 1 posizioan) eta Realterm-en moduko aplikazioren bat edota programa sinpleren bat idatziz, AT komandoak bidali moduluari. Kasu honetan, Arduino softwarea erabiliz programa sinple bat erabili da, komandoak jaso eta hauen emaitza erakusten duena; programa hau 31.irudian ikus daiteke.

```
void loop()
{

   if (Serial.available())
    BTSerial.write(Serial.read()); //Serie lerrotik jasotakoa bluetooth moduluari bidali

   if (BTSerial.available())
    Serial.write(BTSerial.read()); //Idatzi bluetooth moduluaren erantzuna pantailan

}
```

31.irudia: Bluetooth modulura AT komandoak bidaltzeko programa

Konfigurazioan aldatu dena honako hau izan da: moduluaren izena, pasahitza, abiadura eta rola. Hauetatik garrantzitsuenak azken biak dira; lehenik eta behin abiadura 9600 balioan jarri da, ondoren komunikaziorako programa bidez ere abiadura hau erabili behar izan da. Azkenik, rola ere zehaztu da. Modulua morroi gisa konfiguraturik badago, konexioa egiten duen gailua bestea izango da; aldiz, nagusi bezala badago, modulua bera da konexioa ezartzen duena. Hau argi izanik, kasu honetan komeni dena modulua morroi gisa egotea da, beraz, honela konfiguratu da. Honela geratu da konfiguratuta modulua:

```
+NAME:ARDUINO_N
OK
+PSWD:1234
OK
+ROLE:0
OK
+UART:9600,0,0
OK
```

32.irudia: Bluetooth moduluaren konfigurazioa

Behin konfigurazioa amaituta, AT mode etengailua 0 posizioan jarri da, bluetooth bidezko komunikazio gardena izan ahal izateko puntu honetatik aurrera.

4.2.3 Programatu

Behin dena prest izanik, azken urratsa programatzea izan da, bluetooth moduluak RC Car aplikaziotik jasotzen dituen komandoak prozesatzeko eta horien araberako aginduak exekutatzeko. Programa oso sinplea da: modulua komando bat iritsi

zain egongo da eta hau iristean komando honi dagokion agindua exekutatu beharko du. Hona hemen programaren zati bat:

```
 // Itxaron komandoren bat jaso arte
if (BTSerial.available()){

    // Irakurri jasotako komandoa
    char komandoa = BTSerial.read();
    BTSerial.flush(); //Cache memoria garbitu

    //Komandoa prozesatu
    switch(komandoa){

       case 'F': //Aurrera
         motorrak.aurrera();
         break;

        case 'B': //Atzera
          motorrak.atzera();
          break;
```

13.taula: Bluetooth bidez jasotako komandoen prozesaketa

4.3 Ultrasoinu sentsorea segurtasun neurri gisa

Aurrez aipatu bezala, nahiz eta robota urrunetik kontrolatu den, komenigarria izan da segurtasun neurri gisa ultrasoinu sentsore bat izatea, objekturen batera gehiegi gerturatzen bada, robota geldiarazteko talka egin baino lehen.

4.3.1 Ultrasoinu sentsorea konfiguratu komando bidez

Ikusi dugun moduan, ultrasoinu sentsoreak lan egiteko modu desberdinak ditu, baina proiektu honetan interesatzen dena *trigger* modua da: objekturen batera aurredefinitutako distantzia batera edo gutxiagora gerturatuz gero, ultrasoinuko TRIG hankaren balioa 1-etik 0-ra pasako da.

Lan egiteko modu hau konfiguratu beharra dago, eta honetarako zenbait komando erabili dira. Honako hauek dira, hitzez azalduta, exekutatu diren komandoak:

1. Lan egiteko modua finkatu: modu autonomoa (*trigger* modua) – 0x02 @-an idatzi
2. Muga balioa idatzi, 50 cm – 0x00@ eta 0x01@ (2 komando)

Hitzez adierazitakoa komandora pasaz gero honako hauek izango lirateke exekutatu beharreko komandoak:

cmmd1 [ ] = { 0x44, 0x02, 0xaa, 0xf0 } // Modu autonomoa

cmmd2 [ ] = { 0x44, 0x00, 0x32, 0x76 } //Muga-balioaren pisu txikieneko bytea: 0x32

cmmd3 [ ] = { 0x44 , 0x01, 0x00, 0x45 } //Muga balioaren pisu handieneko bytea: 0x00

//50 = 0000 0000 0011 0010 = 0x00 0x32

Komando hauek exekutatzea nahikoa izan da ultrasoinu sentsorea konfiguratzeko, eta objekturen bat 50 cm edo gertuago izatean, TRIG hankaren balioa 1tik 0ra aldatzen da. Muga balio optimoa erabakitzerako orduan hainbat froga egin dira eta azkenean 50 cm izatea erabaki da. Robota abiadura maximoan doala, oztopo bat detektatzean frenatzen hasi eta gelditu arte egiten duen distantzia 50 cm baino gutxiago da. Balio honekin egindako frogetan robotak inoiz ez du oztopoa jo.

Komando hauek exekutatzeko, programa sinple bat idatzi da, serie lerrotik komandoak bidaltzen dizkiona ultrasoinu sentsoreari. Honetarako, Romeoren bi pin digital (10,11) aukeratu eta aurrez erabili den *SoftwareSerial* liburutegia erabili da, bi pin horiek serie pin gisa emulatzeko (TX, RX). Ondoren, *for* begizta baten bidez komando bakoitzaren osagaiak bidaltzen dira banan-banan, komando baten bidalketatik hurrengo bidalketara 2 segundo itxaronez.

33.irudian ikus daiteke programa honen egitura nagusia:

```
//EEPROM memorian idazketak

 for (i=0;i<4;i++) ULSerial.write(cmmd1[i]);

 delay(2000);

 for (i=0;i<4;i++)ULSerial.write(cmmd2[i]);

 delay(2000);

 for (i=0;i<4;i++) ULSerial.write(cmmd3[i]);

 delay(2000);
```

33.irudia: Ultrasoinu sentsorea konfiguratzeko programa

Behin hau exekutatuta, jada konfiguratuta dago ultrasoinua honek betetzea nahi den funtzioa bete dezan. Beraz, hurrengo pausoa Romeoren *interrupt* motako hankatxoak identifikatzea izan da, eta hauetako bat ultrasoinuaren TRIG hankatxoarekin erabiltzea.

4.3.2 Etenak

Romeo mikrokontrolagailuak 2 kanpoko eten ditu: int0 eta int1 izenez deituak, 2. eta 3. pinetan, hurrenez hurren. Beraz, ultrasoinuaren TRIG hankatxoa hauetako batera konektatu behar izan da, int0 aukeratu da (pin 2).

Programa bidez etenak eta hauen zerbitzu errutinak konfiguratzea lan erraza da. Honetarako Arduino IDEak *attachInterrupt()* funtzioa eskaintzen du:

attachInterrupt(etena_id, zerbitzu_errutina, eten_modua)

etena_id: etenaren identifikadorea, 0 (int0) edo 1 (int1)

zerbitzu_errutina: zehaztutako etena gertatzean zein zerbitzu errutinari deitu

eten_modua: noiz kontsideratuko den etena:

- LOW: pin horretako balioa 0 denean
- CHANGE: pin horretako balioa aldatzen denean
- RISING: pin horretako balioa 0-tik 1-ra pasatzen denean
- FALLING: pin horretako balioa 1-tik 0-ra pasatzen denean

Robotak objektu bat 50 cm baino gertuago duenean, TRIG pin-a 1-etik 0-ra pasatzen da eta urruntzean, berriz, 0-tik 1-era. Hau dela eta, erabili den eten modua CHANGE izan da; 1-tik 0-ra aldatzean, motorrak geldiarazten dira, aurrera joaten ez uzteko (atzerako mugimenduak soilik onartuko dira); eta 0-tik 1-ra aldatzean, berriro ere zerbitzu errutinara joko da, mugimendu guztiak onartzeko.

Beraz, honela idatzi da *attachInterrupt* funtzioa:

attachInterrupt(0, oztopoa, CHANGE)

Honela, Romeoren 2.pin-aren balioa aldatzen den aldiro *oztopoa()* funtzioa exekutatzen da. Funtzio honen barnean, aldaketa zein izan den ikusi behar da (0-tik 1-ra edo alderantziz), eta honen arabera motorrak gelditu eta aurrerako mugimenduak galarazi, edo aurrerako mugimenduak ahalbidetu.

Hona hemen oztopoa() funtzioa:

```
void oztopoa(){
   Serial.println("Etenaren zerbitzu errutinan!");
   int trigger=digitalRead(2);
   if (trigger==0){ //Oztopoa gertuegi
      ARRISKUAN=true;
      Serial.println("Arriskuan");
      //Motorrak gelditu
      analogWrite(esk_abiadura, 0);
      analogWrite(ezk_abiadura, 0);
   }
  else{ //Oztopoa ekidin da
    ARRISKUAN=false;
    Serial.println("Arriskutik kanpo");
    //Motorran gelditu aurreko abiaduran abiarazi
    analogWrite(esk_abiadura, abiadura);
    analogWrite(ezk_abiadura, abiadura);
  }
}
```

34.irudia: Ultrasoinuaren zerbitzu errutina

ARRISKUAN aldagaiak *true* balioa duenean, ezin dira aurrerako mugimenduak egin; honek *false* balioa hartzen duen arte.

Beraz, aurreko bertsioaren programan (motorren kontrola bluetooth bidez) ultrasoinu sentsorea gehitzeko egin beharreko aldaketak honako hauek izan dira: *attachInterrupt*() lerroa eta zerbitzu errutina gehitu eta *bluetooth* aginduetan ARRISKUAN *true* den begiratu aurrerako mugimenduak agindu aurretik, adibidez:

```
void aurrera(){
  if(!ARRISKUAN){
    Serial.println("Aurrera");
    digitalWrite(esk_norantza, HIGH);
    analogWrite(esk_abiadura, abiadura);
    digitalWrite(ezk_norantza, HIGH);
    analogWrite(ezk_abiadura, abiadura);
  }
  else{
    Serial.println("Arriskuan, ezin du aurrera egin");
  }
}
```

35.irudia: Robota arriskuan dagoen begiratu

Behin programa prest izanda, ultrasoinua konektatzea soilik falta da; honetarako konexio hauek egin behar izan dira:

5v – 5v

GND – GND

TRIG – pin 2

4.4 GPS bidezko lokalizazioa

Proiektuaren irismenean aipatu bezala, robotaren ezaugarri nagusitako bat bere uneko posizioa zein den (latitudea eta longitudea) kalkulatu dezakela da. Beraz, ezaugarri hau burutzeko, ezinbestekoa izan da GPS/GSM/GPRS moduluaren erabilera.

Aurrez aipatu den moduan, modulu hau bi modutan erabili daiteke: USB bidez ordenagailura konektatuta, serieko terminal batetik AT komandoak bidaliz; edota Romeo mikrokontrolagailuaren bidez, AT komandoak kontrolagailuko programatik bidaliz. Moduluaren funtzionamendua ikasteko, sinpleena lehenengo aukera denez, hasiera batean honela erabili da modulua, desiotako emaitza lortu arte.

4.4.1 USB bidez ordenagailura konektatuta

Modu hau aipatu bezala nahiko sinplea da. Lehenik eta behin modulua Romeoren gainean konektatu behar da (beste *shield* guztiekin egin den moduan), eta ahaztu gabe kanpo elikadura konektatu behar zaio; modulu honek 7 - 12 V inguru behar baititu ongi funtzionatzeko (USB bidezko elikadura ez da nahikoa).

AT komandoak modulua konfiguratzeko erabiltzen dira. Kasu honetan ez da oso konplexua izan GPSa konfiguratzea; izan ere, nahi izan den bakarra GPSari modu autonomoko funtzionamendua ezartzea izan da. GPSa modu honetan dagoelarik, uneoro ari da sateliteetako mezuak jaso, prozesatu eta uneko kokapena kalkulatzen.

AT komandoen eskuliburua aztertuta, ikusi da honetarako bi komando nahikoa direla: AT + CGPSPWR=1 (GPSa gaitzeko) eta AT+CGPSRST=1 (GPS modu autonomoan konfiguraturik berrabiarazteko GPSa).

Modulu honen eskuliburuan, kode zati bat ageri da, kontrolagailuan kargatu behar dena. Honen bidez, modulua hasieratu egiten da, ondoren AT komandoen bidezko komunikazioa egin ahal izateko.

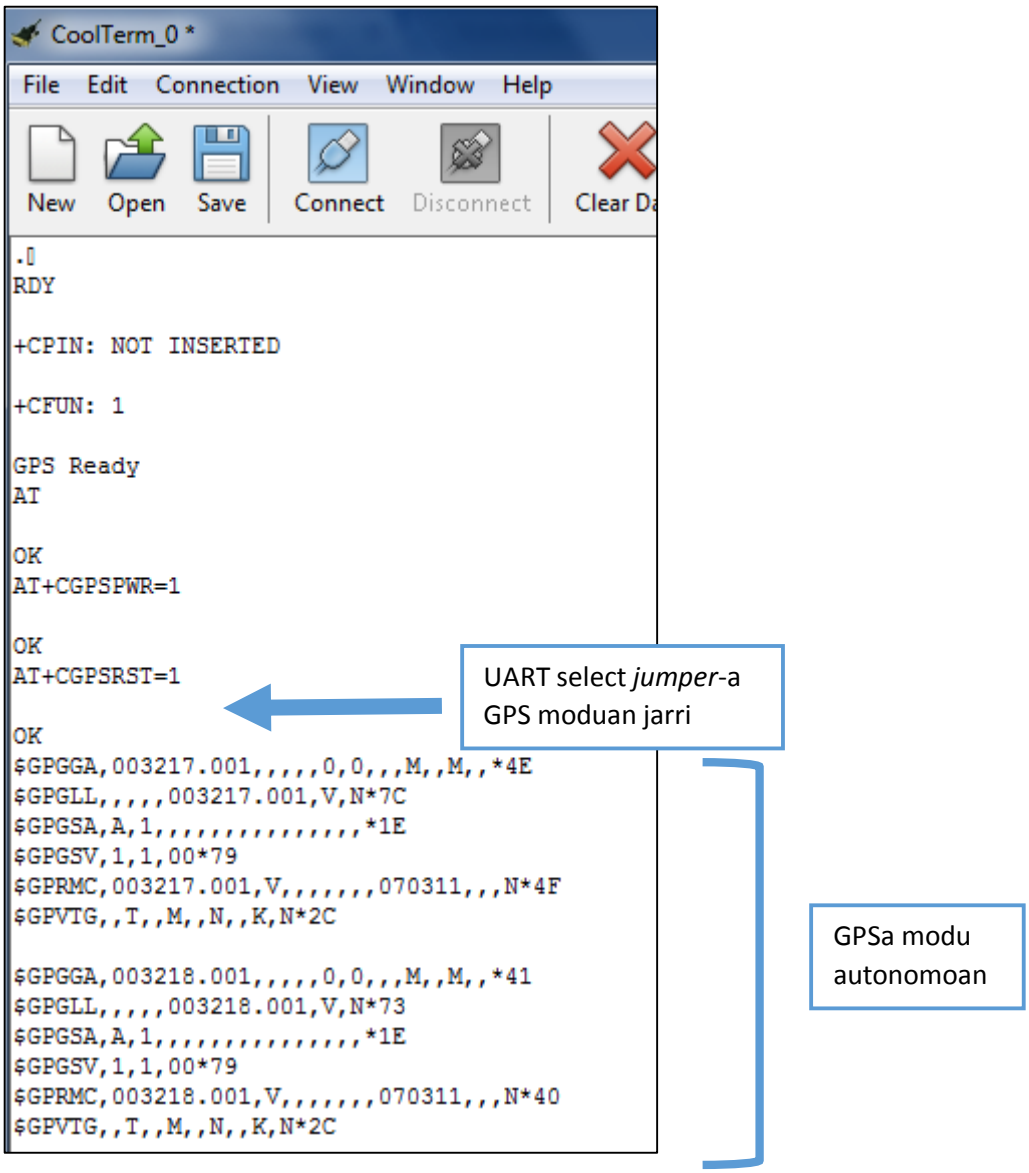
```
void setup()
 {
   //Init the driver pins for GSM function
    pinMode(3,OUTPUT);
    pinMode(4,OUTPUT);
    pinMode(5,OUTPUT);
  //Output GSM Timing
    digitalWrite(5,HIGH);
    delay(1500);
    digitalWrite(5,LOW);
 }
 void loop()
 {
    digitalWrite(3,HIGH);//disable GSM TX、RX
    digitalWrite(4,HIGH);//disable GPS TX、RX
 }
```

36.irudia: GPS/GSM/GPRS modulua AT moduan erabiltzeko konfigurazio programa

AT komandoak bidaltzen hasi ahal izateko honako urrats hauek jarraitu behar dira:

1. S1 switch-a Prog posizioan jarri
2. UART select jumper guztiak kendu
3. Goian aipatutako sketch-a kargatu kontrolagailuan
4. S1 switch-a Comm posizioan jarri
5. S2 switch-a USB posizioan jarri
6. UART select jumper-a GSM moduan jarri (GPS modua konfiguratzeko)
7. RST botoia sakatu itxaron STAT leda piztu arte
8. Ireki serieko terminal bat eta bidali aipatutako AT komandoak
9. Behin GPS modua konfiguratuta, UART select GPS posizioan jarri

Hona hemen egindakoaren emaitza:



37.irudia: GPSa modu autonomoan

Oharrak:

Adibide sinple honekin jada agerian geratu da funtzionamendu modua *jumper* bidez aukeratzearen desabantaila. GPSa modu autonomoan konfiguratu ahal izateko, GSM moduan egon behar da modulua; beraz, behin konfiguratuta *jumper*-ak mugitu behar dira (kasu honetan ezin baita programaz egin) GPS moduan jartzeko.

Bestalde, GPSak balio egokiak eman ahal izateko, kanpoan egon behar du. Honetaz gain, hasieratik ez ditu balio zuzenak ematen eta denbora bat behar du balio egokiak lortzen dituen arte. Denbora hau ez da berdina kasu guztietan, honetan eragina baitu GPSaren kokapenak eta bai eta egoera atmosferikoak ere.

Kasu honetan, esaterako, minutu bat inguru behar izan du balio egokiak eman arte. Hona hemen balio zuzenen irudia:

```
$GPGGA,092315.000,4302.972360,N,00217.204792,W,1,6,2.31,339.004,M,51.140,M,,*4A
$GPGLL,4302.972360,N,00217.204792,W,092315.000,A,A*45
$GPGSA,A,3,20,05,13,10,28,02,,,,,,,3.72,2.31,2.91*02
$GPGSV,2,1,06,20,72,193,41,05,70,286,40,13,47,288,29,10,38,145,42*7B
$GPGSV,2,2,06,28,30,132,37,02,21,219,41*7C
$GPRMC,092315.000,A,4302.972360,N,00217.204792,W,0.000,0.0,190415,,,A*7A
$GPVTG,0.0,T,,M,0.000,N,0.000,K,A*0D
$GPZDA,092315.000,19,04,2015,,*50
```

38.irudia: GPSak ematen dituen balioak

38.irudian informazio asko ageri den arren, proiektu honetarako ez da hori guztia interesatzen; behar diren datu bakarrak latitudea eta longitudea dira. Datu hauek soilik eskuratu ahal izateko programan eskuratutako datuak tratatu dira.

4.4.2 Mikrokontrolagailura konektaturik programa bidez

GPSa programa bidez konfiguratzeko egin behar izan dena oso antzekoa da; desberdintasun bakarrak honako hauek dira: Batetik, terminalaren bidez bidalitako komandoak *Serial.println()* funtzioaren bidez bidali dira modulura, eta bestetik, GPS/GSM modua aukeratzeko 3, 4 eta 5 pin-ak erabili dira (*jumper*-ak mugitzen ibili gabe).

Beraz, puntu honetatik aurrera S2 switch-a Arduino posizioan utzi da eta S1 switch-a Prog posizioan jarri programa kargatzeko, eta Com posizioan, berriz, programa exekutatu behar denean.

GPSa konfiguratzeko nahikoa izan dira 39.irudian agertzen diren programa lerroak:

```
void setup()
{
  //GPS/GSM moduak aukeratzeko defektuzko pin-ak
  pinMode(3,OUTPUT);
  pinMode(4,OUTPUT);
  pinMode(5,OUTPUT);
  digitalWrite(5,HIGH);
  delay(1500);
  digitalWrite(5,LOW);

  digitalWrite(3,LOW);//GSM modua gaitu
  delay(2000);
  digitalWrite(4,HIGH);//GPS modua desgaitu
  delay(2000);
  Serial.begin(9600);
  delay(7000);

  //AT komandoen bidez konfiguratu modulua
  Serial.println("AT");
  delay(2000);
  //GPS elikadura gaitu
  Serial.println("AT+CGPSPWR=1");
  delay(2000);
  //GPSa berrabiarazi modu autonomoan
  Serial.println("AT+CGPSRST=1");
  delay(2000);
  digitalWrite(4,LOW);//Gaitu GPS modua
  digitalWrite(3,HIGH);//Desgaitu GSM modua
```

39.irudia: GPSaren konfigurazioa programa bidez

Modulua programa bidez erabiltzearen konplexutasun handiena datuen erauzketa da; hau da, aurreko atalean ikusi dugun moduan, modu autonomoan dagoela GPSak datu ugari ematen ditu; baina, kasu honetarako interesatzen diren bakarrak kokapenaren latitudea eta longitudearen balioak dira; beraz, programa bidez datu horiek soilik hartu eta erakutsi behar izan dira.

Honetarako, *Dfrobot*-en webgunean kode proposamen bat dago argitaraturik. Hau hartu, ulertu eta hainbat moldaketa egin zaizkio proiektuaren beharretara egokituz eta behar ez dena kenduz, exekuzio denboran aurreztearren.

Idatzitako programarekin lortutako balioak 40.irudian agertzen direnak dira. Kasu honetan ere, ikus daiteke ez dituela balio zuzenak ematen hasieratik. Baliorik ez duen kasuetan *inf* inprimatzen du eta ondoren, balioak lortzen hastean hauek inprimatuko ditu. Exekuzioan aurrera egin ahala gero eta balio zehatzagoak lortzen dira.

```
Latitudea:inf,
Longitudea:inf,


Latitudea:,
Longitudea:inf,


Latitudea:inf,
Longitudea:infW


Latitudea:43.0303421N
Longitudea:2.1720566W


Latitudea:inf,
Longitudea:2.1719884W


Latitudea:43.0297546N
Longitudea:W


Latitudea:43.0298042N
Longitudea:2.1719484W


Latitudea:43.0297393N
Longitudea:2.1720113W


Latitudea:43.0297164N
Longitudea:2.1720206W
```

40.irudia: GPSaren balioak

4.4.3 Robotari kokapenaren funtzionamendua gehitu

Behin moduluaren funtzionamendua ezagututa eta hau nola programatu jakinda, modulua robotari gehitzea izan da eman beharreko azken urratsa. GPSa robotaren gainaldean jarri behar izan da, sateliteetako informazioa ongi eskuratzeko oztoporik gabe. Beraz, ez da zuzenean Romeo mikrokontrolagailuaren gainean jarri, gainerako *shield*-en kasuan bezala, ondoren robota eraikitzean mikrokontrolagailua eta gainerako *shield*-ak tapatuta doazelako. Gainera, moduluak erabiltzen dituen pin guztiak puntu honetan jada, ez zeuden libre. Batetik, 4. eta 5. pinak bi motorren kontrolerako erabili dira, eta Rx eta Tx lerroak puntu honetan ere ez dira erabili nahi izan; bluetooth-aren kasuan aipatutako arrazoi berbera dela eta.

Hau horrela izanik, modulu honentzat bost pin digital behar izan dira, eta nahiz eta puntu honetan 4 soilik izan libre, arazoa bluetooth-ak erabiltzen duen 9.pina erabiliz konpondu da. Bluetooth moduluaren elikadura programa bidez kontrolatze aldera, hasiera batean moduluaren Vcc hankatxoa mikrokontrolagailuaren 9.pinarekin konektatu zen. Baina, arazo honekin topatzean hau aldatu egin da. Vcc hankatxoa mikrokontrolagailuaren Vin hankatxoarekin konektatu da eta horrela 9.pina GPS moduluarentzat libre geratu da. Azkenik, serie komunikaziorako Rx eta Tx pin-ak erabili beharrean, 8 eta 12 erabili dira, jada aipatutako *SoftwareSerial* liburutegia erabiliz.

Laburbilduz, honako konexio hauek egin dira GPS moduluaren eta Romeo mikrokontrolagailuaren hankatxoen artean:

GPS/GSM/GPRS modulua	*Mikrokontrolagailua*
Rx	8
Tx	12
3	3
4	9
5	13

14.taula: GPS/GSM/GPRS moduluaren konexioak

4.5 Wifi bidez informazioa bidaltzen

Wifi bidez informazioa bidali ahal izateko, Wifi *shield*-a erabili da. *Shield* honen ezaugarriak eta eskaintzen dituen aukerak aztertu ondoren, honako hau egitea erabaki da: *echo* zerbitzari bat programatzea; Wifi *shield*-ak bezeroaren *rol*-a hartu du eta honen egitekoa zerbitzarira konektatu eta zerbitzariari mezuak bidaltzea izan da. Azkenik, zerbitzariak egiten duen bakarra bezerotik jasotako informazioa pantailaratzea da. Hau egin ahal izateko, *socket*-ak erabili dira, TCP/IP arkitekturan aplikazio eta garraio mailen arteko komunikazioa ahalbidetzeko interfazea eskainiz.

Garraio mailako protokoloari dagokionez, *shield* honek, aipatu bezala, TCP eta UDP protokoloak biak onartzen ditu. Aplikazioaren ezaugarriak aztertuta, UDP egokiagoa dela erabaki da; fidagarritasunak ez baitu berebiziko garrantzirik, paketeren bat galtzen bada ez duelako arazo larririk suposatuko eta horrela, TCP-k ezartzen duen gainkarga saihesten da.

Hau argi utzirik, honako hauek izan dira garapenean eman behar izan diren pausoak:

1. Echo UDP zerbitzaria programatu
2. Beharrezko AT komando guztiak aurkitu Wifi *shield*-a zerbitzari horretara konektatu eta mezuak bidali ahal izateko

4.5.1 Echo UDP zerbitzaria programatu

Atal hau burutu ahal izateko hirugarren mailako Sare Zerbitzuak eta Aplikazioak (SZA) irakasgaiko apunteak oso lagungarri suertatu dira; bertan, 2. gaian, *Socket*-ak landu baitziren.

Aplikazioa *socket* interfazearekin programatzeko C lengoaia erabili da. Aplikazioak berak ez du konplexutasun handirik, jasotako mezuak pantailan inprimatzea baita honen eginbehar nagusia.

Oinarrizko UDP zerbitzari baten egitura honako hau da:

1. *Socket*-a sortu
2. *Socket*-ari helbide bat esleitu (IP helbidea eta portua)
3. Begizta infinitua: *socket*-aren bidezko komunikazioa

Socket-ak sortzeko C lengoaian *struct sockaddr_in* egitura eskaintzen da; *struct* honek honako egitura hau duelarik:

- sin_family //*socket mota*
- sin_port //zein portutan entzungo duen
- sin_addr // IP helbidearen *struct*-a. Egitura: s_addr

Socket-aren sorrera eta helbideen esleipena honela egin da:

```
Struct sockaddr_in zezrb_helb, bez_helb;

Int sock;

Sock=socket(AF_INET, SOCK_DGRAM, 0),

Zerb_helb.sin_family = AF_INET;

Zerb_helb.sin_addr.s_addr = hton1(INADDR_ANY);  //makinako edozein helbide

Zerb_helb.sin_port = htons(PORTUA);

//helbidea esleitu socketari

bind(sock, (struct sockaddr *) &zerb_helb, sizeof(zerb_helb));
```

15.taula: *Socket*-aren sorrera eta helbide esleipena

Behin *socket*-a sortuta eta helbidea esleituta, falta den bakarra *socket*-aren bidezko komunikazioa inplementatzea da. *Socket*-a erabiliz datuak jasotzeko honako funtzio hau erabiltzen da:

int n = recvfrom (sock, buf, MAX_BUF, 0, (struct sockaddr *) &bez_helb, helb_tam);

Sock aldagaia sortu berri den *socket*-a da. Jasotako datuak buf aldagaian gordetzen dira eta jaso nahi diren byte kopuru maximoa MAX_BUF aldagaiaren bidez zehazten da. Hortik aurrerakoa, bezeroaren *socket*-aren inguruko informazioa da: helbidea eta helbidearen tamaina. Bezeroaren helbidea ondoren bezeroari mezua bidaltzeko erabiltzen da; baina, kasu honetan, zerbitzariak ez dio bezeroari ezer bidaliko; beraz, begizta nagusiaren oinarrizko egitura honako hau da:

```
while(1)
{
        n=recvfrom(sock, buf, MAX_BUF, 0, (struct sockaddr *) &bez_helb, &helb_tam);
        printf("%s",buf);
}
```

16.taula: *socket*-aren bidezko komunikazioa

Behin zerbitzaria prest izanik, UDP bezeroa martxan jartzea izan da hurrengo egitekoa.

4.5.2 AT komandoak UDP bezeroa martxan jartzeko

Aurrez aipatu bezala, produktuaren webgunean aurki daiteke AT komandoen eskuliburua. Bertan azaltzen diren komandoak aztertu ondoren, lehenik eta behin

oinarrizko komandoak zein diren azpimarratuko dira jarraian. Honako hauek izango lirateke:

KOMANDOA	*AZALPENA*
AT + WS	Eskuragarri dauden wifi sareak inprimatzen ditu
AT + WWPA = Pasahitza	Konektatu nahi dugun AP (*Access Point*) – aren pasahitza zehazteko
AT + NDHCP = 1	DHCP bidezko helbide esleipena gaitzeko (0 modu manualean egiteko)
AT + WA = AP_izena	Zehaztutako APra konektatu ahal izateko
AT + WSTATUS	Konektatutako sarearen inguruko konfigurazio informazioa lortzeko
AT + WD	Konektatutako saretik deskonektatzeko

17.taula: Wifi shield-aren oinarrizko AT komandoak

Komando hauen bidez, posible da eskuragarri dauden wifi sareak eskanetzea eta hauetakoren batera konektatu ahal izatea. Beraz, falta den bakarra UDP bezero bat martxan nola jarri jakitea da. Esan beharra dago, hau lortzeko lan handiak izan direla, eta Wizfi210 txiparen informazioa dagoen foroan *WIZnet User forum-WizFi210/220* (WIZnet Co. , 2013) idatzi eta bertan emandako erantzun azkar eta zuzenari esker lortu da, azkenik, UDP bezeroa martxan jartzea. Honako hauek dira exekutatu beharreko komandoak, agertzen diren ordena berean:

KOMANDOA	*AZALPENA*
AT + WAUTO = 0, AP_izena	Konfiguratu wifi parametroak konexio automatikorako
AT + NAUTO = 0, 0, @_zerb, Portua_zerb	Konexio automatikoan erabili beharreko parametroak zehaztu: Bezero moduan (0), UDP protokoloarekin (0), @_zerb helbideko zerbitzariarekin konektatu Portua_zerb portuan.
ATA	Hasi konexio automatikoa

18.taula: UDP bezeroa martxan jartzeko AT komandoak

Beraz, azkenik, honako hauek dira exekutatu beharreko komando guztiak "nireAP" izeneko wifi sarera konektatu nahiko balitz, "pass_AP" pasahitza duelarik, eta sare horretan aurrez programatutako *echo* UDP zerbitzaria martxan egongo balitz 192.168.1.110 helbidearekin eta 50001 portuan:

AT+WWPA=123456789

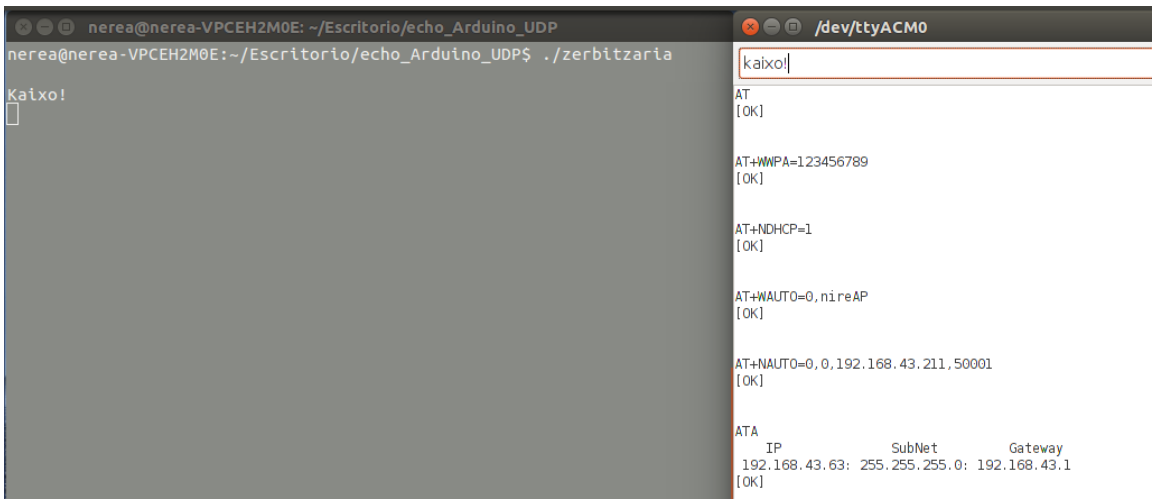
AT+NDHCP=1

AT+WAUTO=0, nireAP

AT+NAUTO=0,0,192.168.43.120, 50001

ATA

Komando hauen ondoren idatzitako guztia, aipatutako zerbitzariari bidaltzen zaio zuzenean. 41.irudian ikus daiteke honen exekuzioaren emaitza:



41.irudia: UDP aplikazioa martxan

Kasu honetan Wifi *shield*-a ordenagailuarekin komunikatu da zuzenean, komandoak ordenagailutik bidaliz. Hau programaz egiteko ez dago inongo arazorik, GPSaren kasuan ikusi den moduan.

4.5.3 Robotetik zerbitzarira informazioa bidaltzen

Gailuaren zehaztapenetan aipatu bezala, *shield* honek Rx(0) eta Tx(1) hankatxoak erabiltzen ditu komunikaziorako. Puntu honetara arte, ez dira mikrokontrolagailuko hankatxo hauek erabili nahi izan, programen exekuzioetarako komenigarria zelako ordenagailurako konexioa *debugger* gisa. Baina, hemendik aurrera exekuzioetan inprimatu beharrekoa zerbitzarira bidali da wifi bidez. Aipatutako hau lortzeko programan ez da aldaketa handiegirik egin; serieko komunikazioaren konfigurazioa soilik aldatu da (wifi bidezko komunikazioa ezarri); programan zehar dauden *print*-ak berdin mantenduz.

Beraz, honetarako konfiguratu_wifi( ) izeneko funtzio bat definitu da, eta setup() funtzioaren hasieran deitu zaio programa honi; wifi-ari dagokion konfigurazio guztia egin dezan. Honako hau da konfiguratu_wifi( ) azpiprograma:

```
void konfiguratu_wifi(){

 Serial.begin(115200);
 delay(3000);
 Serial.println("AT");
 delay(2000);
 Serial.println("AT+WWPA=123456789"); //Konektatuko garen APren pasahitza
 delay(2000);
 Serial.println("AT+NDCHP=1"); //DHCP onartu
 delay(2000);
 Serial.println("AT+WAUTO=0,nireAP"); //Konfiguratu wifi sarerako konexioa
 delay(2000);
 Serial.println("AT+NAUTO=0,0,192.168.43.211,50001");//Konektatu zerbitzariarekin UDP bezero gisa
 delay(2000);
 Serial.println("ATA");
 delay(2000);
 Serial.println("Robota martxan!");
}
```

42.irudia: Konfiguratu_wifi programa

Lehenengo mezu guztiak wifi-a konfiguratzeko AT komandoak dira. Behin sarera konektatuta eta zerbitzariarekin konektatuta, azkena dagoen Serial.println("Robota martxan!") aginduak, wifi bidez zerbitzariari bidaltzen dio mezua, honek inprima dezan.

4.5 Elikadura

Elikadurak garrantzi handia du proiektuan, beraz, honi atal bat eskaintzea erabaki da; beharrezko zehaztapen guztiak emanez. Aurrez aipatu den bezala, robotaren motorren ezaugarriak kontuan harturik, beharrezkoa izan da 14,4 V-eko elikadura bat izatea motorrak mugitu ahal izateko. Hasiera batean pentsa zitekeen mikrokontrolagailua elikatzeko ere bateria hau erabiltzea, baina aurrez aipatu bezala, bateria honek mikrokontrolagailuak jasan dezaken gehienezko tentsio maila gainditzen du (12 V). Hau horrela izanik, beharrezkoa izan da bi bateria erabiltzea: bat motorrak elikatzeko soilik, eta bestea mikrokontrolagailua elikatzeko. Bigarren bateria hau 7,5 V-ekoa da.

Bateriak gehitzeaz gain, hauen erabilera eta beharretarako erosotasunak ematea ere ezinbestekoa izan da. Honexegatik, robotari berak bateria hauek kargatzeko bi konektore jarri zaizkio atzealdean; bai eta etengailu bat ere robota piztu nahiz itzaltzeko. Bateriak kargatzeko bi konektoreak berdinak direnez, etiketa bana dute nahasketarik ez izateko.

Hurrengo atalean azalduko da aipatutako hau nola eraiki den.

4.6 Eraikuntza

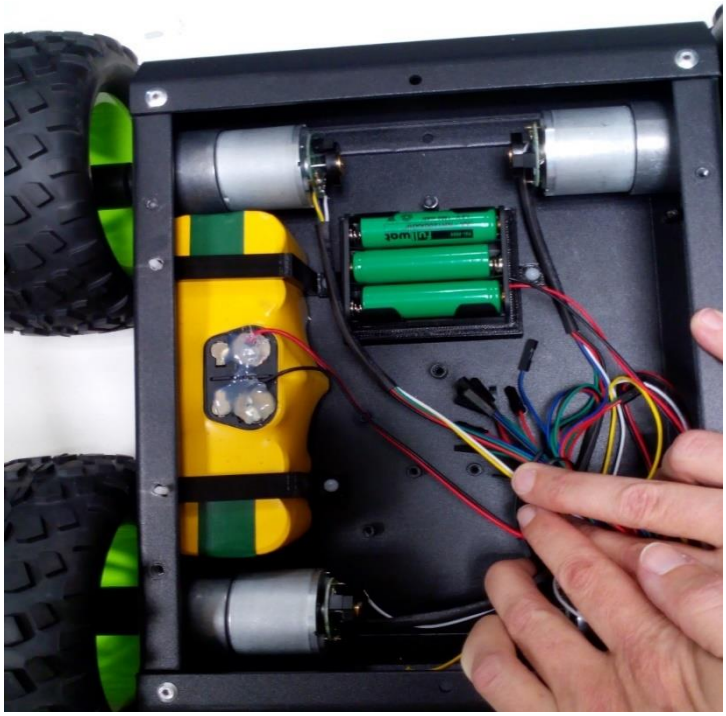
Garaturiko robotak hiru maila ditu. Lehen mailan motorrak eta bateriak daude. 2.mailan mikrokontrolagailua eta *shield* desberdinak daude motor eta elikadura iturriekin konektaturik. Honetaz gain, maila honetan bateriak kargatzeko konektoreak ere badaude, bai eta robota piztu/itzaltzeko etengailua ere. Azkenik, hirugarren mailan, ultrasoinu sentsorea eta GPS modulua kokatu dira.

Jarraian, maila bakoitzean zer egin den azalduko da irudien laguntzaz.

1. Maila

Maila honetan hasieran 4 motorrak soilik zeuden. Bertan 2 elikadura iturriak jartzea erabaki da eta honetarako fakultateko 3D inprimagailuaren bidez hainbat pieza egin behar izan dira; bateriei ongi eusteko. Pieza hauek jarri ahal izateko taladroa erabiliz zuloak egin dira eta plastikozko torlojuekin eutsi.

43.irudian ikus daiteke behe maila hau nola geratu den:



43.irudia: Robotaren 1.maila

2. Maila

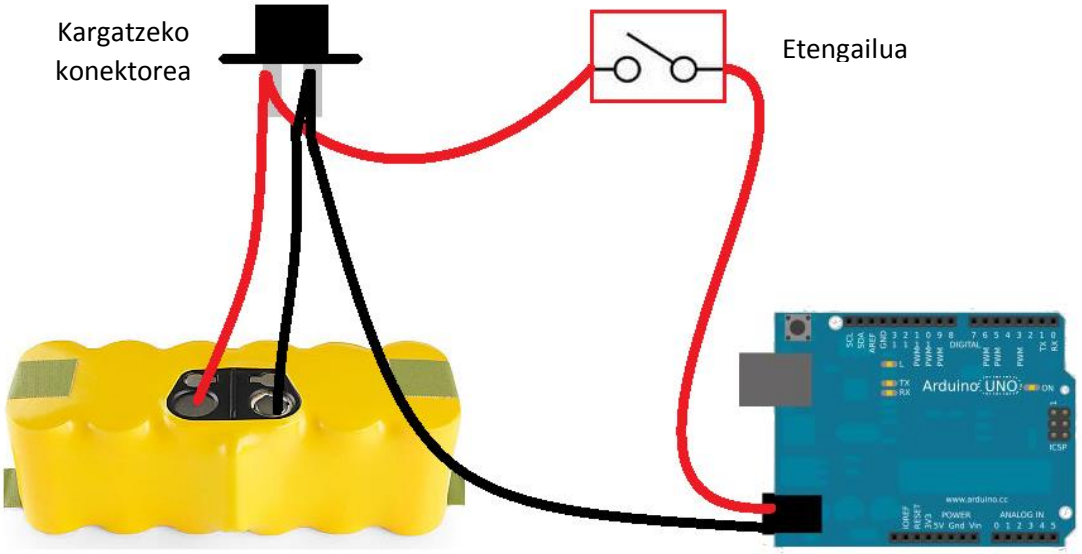
Maila honetan 2 atal nagusi bereiztu dira: batetik, Romeo mikrokontrolagailua eta bestetik, bateriekin zerikusia duen guztia (kargatzeko konektoreak eta etengailua).

Bateriak kargatzeko konektoreak eta etengailua jarri ahal izateko 3 zulo egin behar izan dira bigarren mailako egituraren atzealdean, 44.irudian ikus daitekeen bezala.

44.irudia: Baterien konektore eta etengailurako zuloak

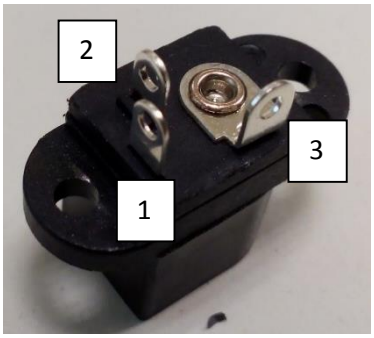
Hurrengo pausoa konektoreak eta etengailua gehitu eta dagozkien konexioak egitea izan da, kableak eta soldagailua erabiliz. Lehenik eta behin, komenigarria da argitzea nola egin diren konexioak. Egin dena hobeto azaltze aldera, 45.irudia txertatu da. Argitu beharra dago, eskema honetan bateria bakarra ageri dela, bi baterien kasuan egitura bera da eta konpartitzen duten osagai bakarra etengailua da; hau bikoitza izanik.

Hona hemen bateria bakarraren eskema:



45.irudia: Baterien elikadura eta karga konexioen eskema

45.irudian ikus daitekeen bezala, bateria bakoitzaren bi kableak zuzenean kargatzailearen konektorera doaz. Erabilitako konektorea 46.irudian ikus daitekeena da eta honen funtzionamendua nahiko sinplea da. Hiru hankatxo dituen arren, bi soilik erabili dira eta nahasketarik ez egoteko 3 zenbakidun hankatxoa kendu egin zaio. 1 zenbakidun hankatxoan bateriaren tentsio altuko kablea konektatu da eta bigarrenean, lurra. Behin hau eginik, ondoren hanka bakoitzetik kable bana atera da: lurraren kasuan, konektoretik zuzenean mikrokontrolagailura doa eta tentsio altuko kablea, berriz, etengailura. Bi baterien kasurako gauza bera egin da bi konektore erabiliz.



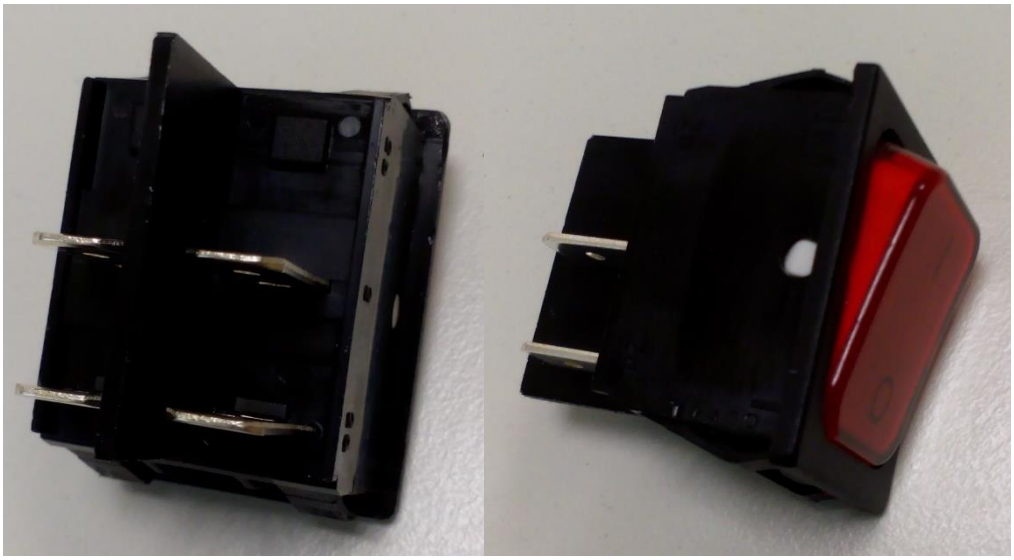
46.irudia: Bateriak kargatzeko konektorea

Etengailuari dagokionez, etengailu bikoitza erabili da, bi bateriak aldi berean piztu nahiz itzali ahal izateko. Etengailuen funtzionamendua ere oso sinplea da. Etengailu normalek bi hankatxo dituzte eta etengailua 0 posizioan dagoenean ez dago bi hanka hauen arteko konexiorik eta etengailua 1 posizioan jartzean bi hanken arteko konexioa egingo da.

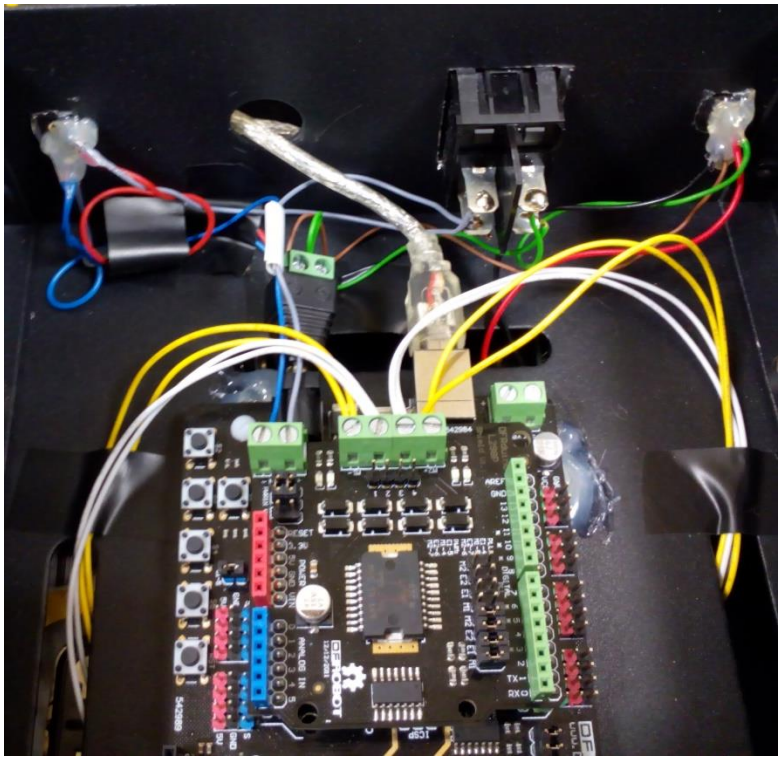
47.irudia: Etengailuen eskema

Etengailuaren hanka batera bateriaren tentsio altuko kablea konektatu da eta beste hankatik beste kable bat jarri da mikrokontrolagailura doana, bateriaren lurraren kablearekin batera.

Etengailua aipatu bezala bikoitza da, beraz, bi etengailu izango bagenitu bezala da, baina independenteak izan gabe. Alde bakoitza bateria batentzako erabili da.


48.irudia: Etengailu bikoitza

Konexioekin amaitzeko azken urratsa, Romeori eta motor *shield*-ari kableak konektatzea izan da. Bi kasutan tentsio altuko kablea etengailutik dator eta lurra, berriz, kargatzeko konektoretik. Behin hau eginda, jada amaitutzat jo da elikadurarekin zerikusia duen guztia. Honela geratu da konexioak burutu ondoren:

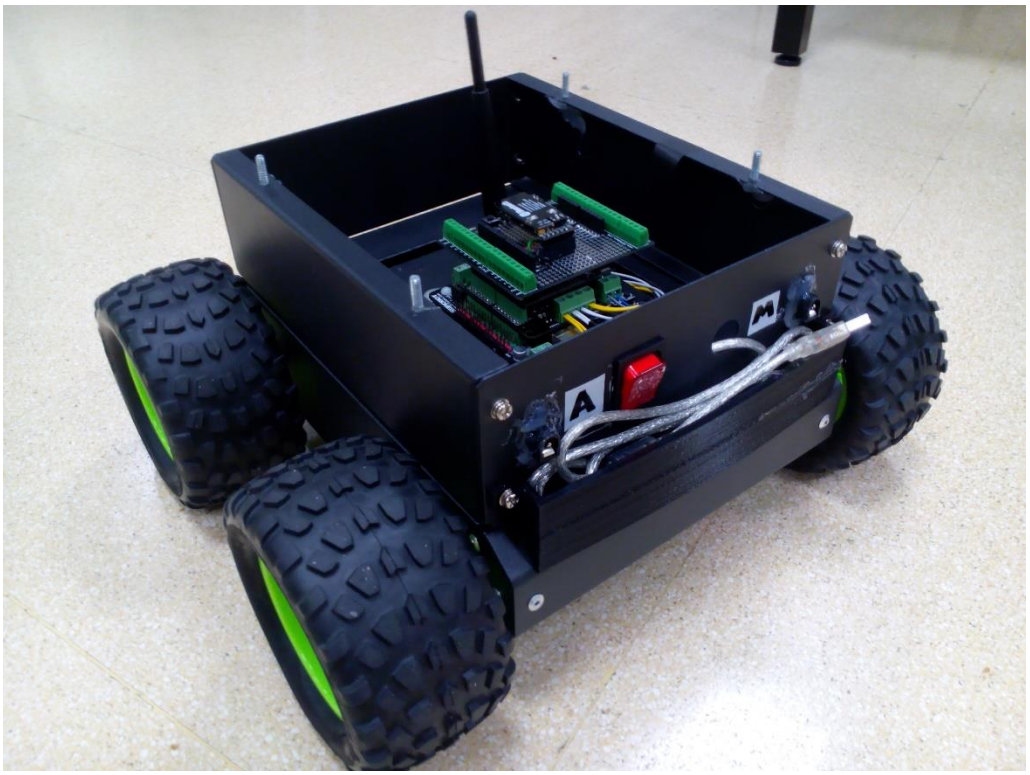


49.irudia: Robotaren 2.mailako elikaduren konexioak

Mikrokontrolagailuaren kokapena burutzeko ataza sinpleagoa izan da. Lehenik eta behin beharrezko zuloak egin dira bigarren mailaren oinarrian, eta bertara lotu da mikrokontrolagailua plastikozko torloju eta hankatxoak erabiliz. *Shield*-ak gehitzeko ez da ezer gehiago egin behar izan, Romeoren gainean baitoaz.

Azkenik, maila honek aurrez zuen zulo bat aprobetxatuz, programa mikrokontrolagailuan kargatu ahal izateko USB kablea pasa da bertatik. Honetaz gain, kable hau gordetzeko pieza bat ere egin da 3D inprimagailua erabiliz. Modu honetan oso erosoa suertatu da Romeo-n programak kargatzea, eta beharrezko osagai guztiak robotean bertan daude.

50.irudian ikus daiteke maila hau nola geratu den:



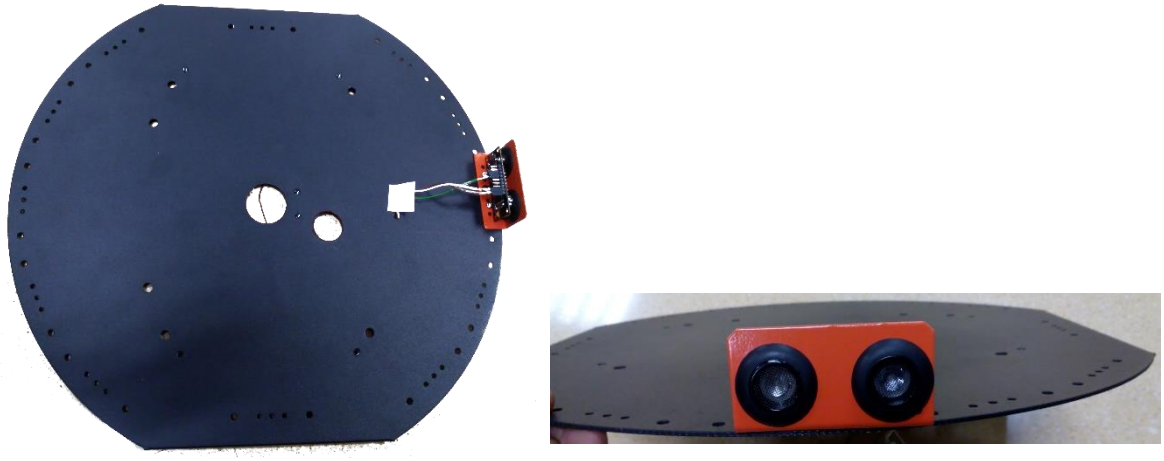
50.irudia: Robotaren 2.maila

3. Maila

50.irudian ikusten den bezala, Romeoren gainean doazen *shield* guztiak gehituta lortzen den altuerak ez du 2.mailaren egituraren altuera gainditzen, baina Wifi *Shield*-ak antena du, eta honek 3.mailaren oinarria jotzen du. Hau dela eta, maila honi dagokion lehen pausoa wifi antena pasa ahal izateko zulo bat egitea izan da.

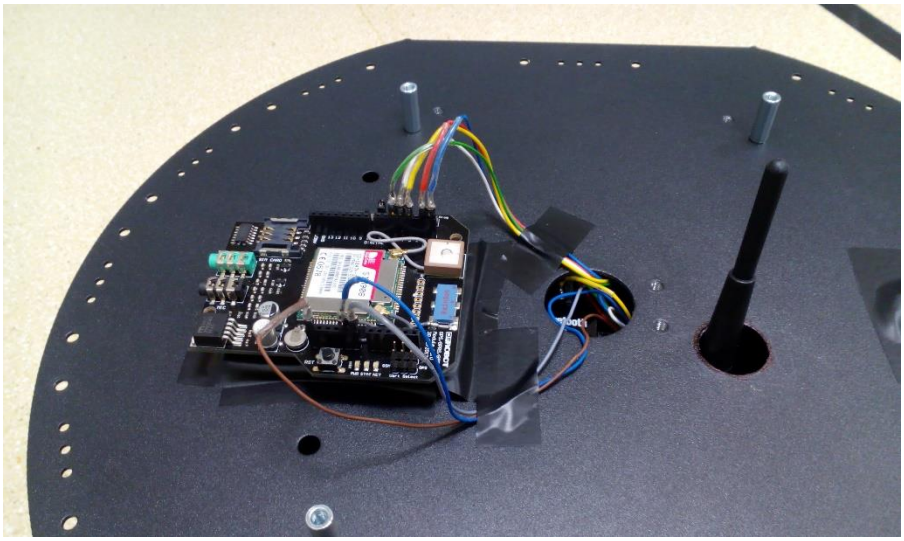
Aurrez aipatu bezala, maila honetan ultrasoinu sentsorea eta GPS modulua jarri dira, arrazoi nabarmenak direla eta: ultrasoinuak aurreko oztopoak ekidin ahal izateko kanpoan eta ikuspen oneko toki batean egon behar du, eta GPSa, sateliteetako seinaleak hartu ahal izateko, gainean egon behar da.

Ultrasoinua kokatzeko egitura sentsorearekin batera zetorren eta oinarriak zituen zuloak aprobetxatu dira bai sentsorea torlojuen bidez eusteko bai eta beharrezko kableak modu txukun batean pasatu ahal izateko ere mikrokontrolagailura.



51.irudia: Ultrasoinu sentsorearen kokapena robotean

GPSari dagokionez, moduluak berak zekarren belaki moduko osagai bat erabili da. Belaki hau robotaren gainaldean kokatu da zinta isolatzailea erabiliz, eta ondoren, GPS modulua bertan jarri da, honek dituen hankatxoak belakian sartuz mugitu ez dadin.



52.irudia: GPS/GSM/GPRS modulua robotean

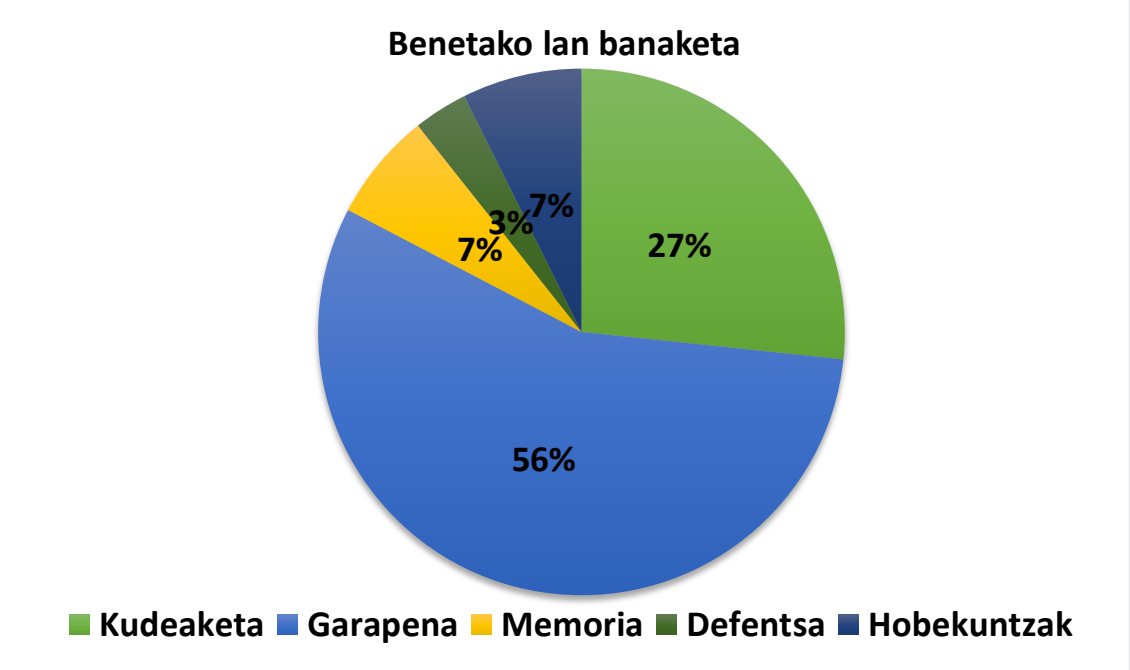
Hauxe izan da robotaren eraikuntzan emandako azken urratsa.

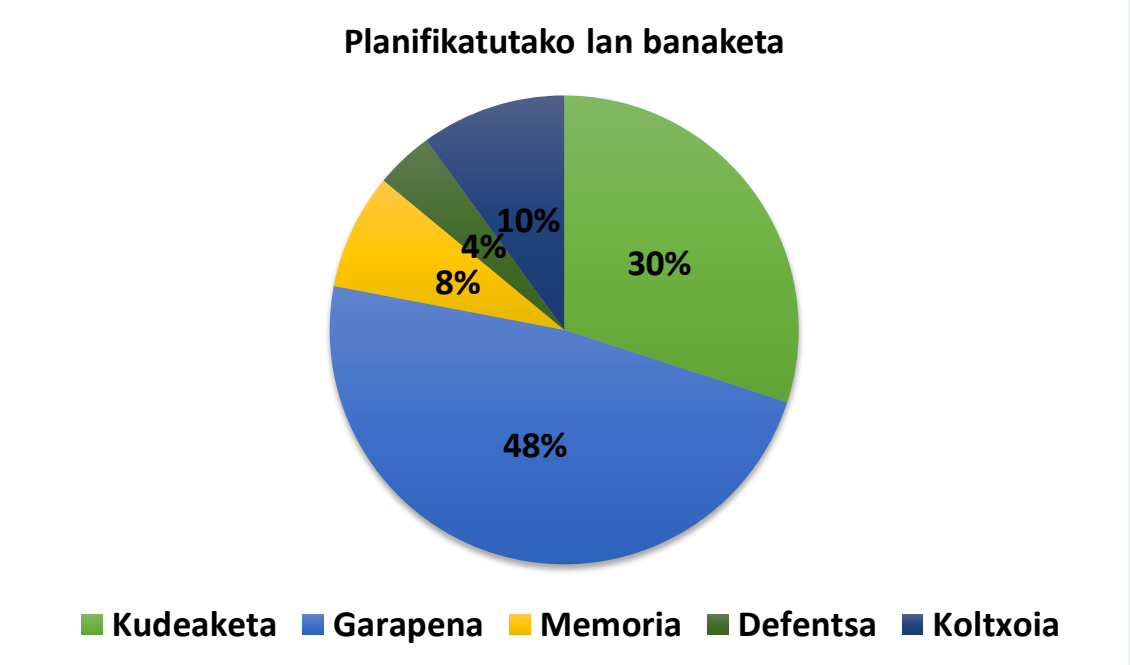
5. KUDEAKETA

5.1 JARRAIPEN ETA KONTROLA

Proiektuan aurrera egin ahala, jarraipen eta kontrol zorrotza egin da desbiderapenak antzeman, hauen aurrean erabakiak hartu eta proiektua finkatutako epeen barruan amaituko zela bermatzeko.

Jarraian, proiektuko 300 orduak zertan eman diren ikus daiteke grafiko baten laguntzaz. Grafiko honekin batera, hasieran planifikatutako lan banaketaren grafikoa ere jarri da, alderaketak egiteko eta gertatu diren desbiderapenak azaltzeko.





53.irudia: Planifikatutako eta benetako lan banaketen grafikoak

5.1.1 Desbiderapenak

Grafikoetan argi ikusten den bezala, desbiderapen nagusia garapenean gertatu da. Beste atazetan, gutxi gora behera, estimatutako denbora eman da ataza hauek burutzen, gehienetan orduren bat edo beste soberan izanik.

Garapeneko desbiderapen hau proiektuaren bizi zikloaren erdialdean hasi zen nabarmentzen eta segituan antzeman zen; beraz, uneoro proiektuaren egoera kontrolpean egon da; eta erreserbatutako orduei esker, ez du eragin negatiborik izan proiektuan.

Lehenengo desbiderapen nabarmena GPSarekin izan da. GPSa zuzen programatu eta probatu ahal izateko hainbat baldintza derrigorrezkoak izan dira: Batetik, GPSa kanpoan egon behar da sateliteetatik seinale ona jasotzen den toki batean. Bestetik, GPSak funtzionatu ahal izateko, ordenagailuak Romeori ematen dion elikadura ez da nahikoa, 7 - 12 V behar baititu. GPSarekin lanean hastean eskuragarri zegoen bateria bakarra motorrena zen, baina hau ezin zen erabili, 14,4V ematen zituelako. Mikrokontrolagailuak darabilen bateria lortu bitartean, motorren bateria erabili da piska bat deskargatuta zegoenean. Azkenik, eguraldiarekiko dependentzia ere izan da; ateri egoteaz gain, oskarbi egotea komeni zelako. Hau dena kontuan izanik, eta programatzeko garaian izan diren zailtasunekin batera, GPS bidezko lokalizazioa lortzeak uste baino lan gehiago eskatu du.

Desbiderapenen beste arrazoi nagusia eraikuntza izan da. Plangintza eta orduen esleipena egiterakoan ez zen pentsatu eraikuntzan hainbeste ordu behar izango zirenik; beraz, estimazio okerra egin zen. Azkenean, robotaren diseinua pentsatzen eta hau eraikitzen 30 ordu inguru behar izan dira. Gainera, osagai berriak gehitu ahala, aldaketak egin behar izan dira robotean. Honetaz gain, eraikuntzaren ataza honek dependentziak eragin ditu garapeneko beste atazetan. Adibidez, bi bateriak robotean jarrita izan arte, ezin izan da motorren kontrola lurrean probatu.

Bestalde, osagai desberdinen elkarketak egiten ere uste baino denbora gehiago behar izan da. Plangintzan aipatu bezala, osagai bakoitza bera bakarrik nola erabili ulertu eta ondoren programatu egin da, bera bakarrik. Azkenean, osagai bakoitza nola erabiltzen den jakinda, ez zirudien lan zaila denak bateratzea kontrol programan, eta hau ere uste okerra izan da. Izan ere, bateratzeko garaian, ordura arte ohartu gabeko arazoak nabarmendu dira; bai eta osagaien arteko bateraezintasunak ere. Honetan ere arazo gehien GPSa gehitzerakoan izan dira; neurri handi batean hau izan delako azkena gehitu den osagaia. GPSa gehituta robotak hiru serieko komunikazio burutzen ditu: Wifi bidezkoa mikrokontrolagailuaren ohiko serie lerrotik, Bluetooth komunikazioa *SoftwareSerial* liburutegia erabiliz eta GPS komunikazioa hau ere *SoftwareSerial* liburutegiarekin. Puntu honetara arte, serieko komunikazioetarako erabili den liburutegi honek duen murriztapena ezezaguna izan da: ezin dira aldi beren bi serieko komunikazio martxan izan liburutegi hau erabiliz; beraz, idatzitako programak ez zuen ondo egiten.

Arazoa zein zen konturatzen eta hau zuzentzen denbora dezente behar izan da. Azkenik, honela konpondu da: Bluetooth bidez komando baten zain dagoenean robota, bluetooth bidezko komunikazioa martxan jarri; ondoren, GPS bidezko komunikazioa behar denean hau martxan jarri (automatikoki bluetooth-arena gelditu egiten da) eta behin GPS bidez beharrezko datuak eskuratzean, berriro ere bluetooth bidezkoa abiarazi. Honen desabantaila nagusia GPSa erabili behar den aldiro hau konfiguratzea da; honetarako denbora piska bat behar baita. Gainera, berriro ere balio zuzenak lortzen denbora bat emango du. Baina, ez da hau baino irtenbide hoberik topatu.

Azkenik, oinarrizko kalitate maila lortu ahal izateko Wifi *shield*-a erabili da eta hau ez zegoen aurreikusia. Egia da bai, GPS bidezko lokalizazioari ordu esleipena egiterakoan ez zela GPStik datuak eskuratzea soilik hartu kontuan; datu horiek bidaltzea ere ataza honetan sartu zen. Baina, ez zen pentsatu wifi bidezko konektibitatera jo beharko zenik. Hau horrela izanik, wifia gehitzeko GPS atazak zituen orduez gain ordu gehiago behar izan dira; beraz erreserbatutako orduak honetan ere erabili dira; baina ordu hauek hobekuntzarako ordu gisa ere har daitezke.

5.1.2 Erreserbatutako orduen erabilera

Proiektuaren plangintzan orduen banaketa egiterakoan 30 ordu erreserbatu ziren badaezpada ere. Oinarrizko maila lortu ahal izateko, badaezpadako 10 ordu erabili behar izan dira; gertatutako desbiderapenei aurre egin ahal izateko.

Hona hemen oinarrizko maila lortzeko behar izan diren orduak:

MEMORIA
20

DEFENTSA
12

GARAPENA	
Motorrak	20
Bluetooth	20
Ultrasoinu	8
Elikadura	15
GPS	35
Wifi	25
Prog. Elkarketak	15
Eraikuntza	30
DENERA	**168**

KUDEAKETA	
Hasieraketa	3
Plangintza	70
J&K	7
DENERA	**80**

DENERA
280

KOLTXOIA
20

19.taula: Oinarrizko maila lortzeko behar izan diren orduak

Beraz, hobekuntzak eta azken xehetasunak finkatu ahal izateko 20 ordu geratu dira.

5.2 KALITATE PLANA

Proiektuan aurrera egin ahala oinarrizko maila lortzeko bidean, egin litezkeen hobekuntzen inguruan hausnartu da. Hona hemen, behin oinarrizko maila lortuta identifikatutako hobekuntza posibleak:

- **Kodea txukundu**

  Robotaren kontrol programa ez da oso txukun geratu. Gainera, hasieran aipatu bezala, gerora begira proiektu hau oinarri gisa hartuz gauza gehiago erantsiko zaizkio; beraz, oso garrantzitsua izango da kontrol programa txukuna, garbia eta ulergarria izateaz gain, aurrerago aldatzen erraza izatea. Hau lortzeko komenigarria litzateke robotaren osagai nagusien liburutegiak garatzea: motorrena, GPSarena eta wifiarena.

- **GPS lokalizazio bidezko nabigazioa**

  Garaturiko robotak GPSa darabil uneko posizioaren berri emateko erabiltzaileari; beraz, lokalizazioaren ataza burututa dago. Ahalmen honek aukera eman diezaioke robotari nabigatzeko beste modu bat emateko: koordenatu bidezko nabigazioa, hain zuzen ere. Oinarrizko mailan robota erabiltzaileak ematen dituen aginduen bidez mugitzen da (aurrera, atzera, ezkerrera, eskuinera) eta bigarren nabigazio honen bidez, robotari koordenatu batzuk eman eta robota puntu horretara joatea izango litzateke helburua.

- **Android aplikazioa garatu**

  Robota mugitzeko erabili den aplikazioa, *Bluetooth RC Car* ez da proiektuan garatua. Beraz, uste da kalitate maila altua emango liokela proiektuari robota kontrolatzeko aplikazioa ere proiektuan bertan garatzeak.

5.2.1 Erabakiak

Behin oinarrizko maila lortuta 20 ordu geratu dira erreserbatutako orduetan hobekuntzak burutzeko. Beraz, ordu kopuru hau eta egin litezkeen hobekuntzak kontuan izanik honako hau egitea erabaki da:

Identifikatutako hobekuntzen artean lehentasun osoa kode txukunketak izan du; beraz, lehenengo hobekuntza hau egitea erabaki da. Erreserbatutako 20 ordu horietatik, 10 ordu inguru beharko direla aurreikusi da liburutegi guztiak idatzi eta kodea moldatzeko.

Behin hau lortuta denbora izanez gero, identifikatutako bigarren hobekuntza egitea erabaki da: GPS bidezko nabigazioa. Hirugarren hobekuntzari dagokionez, ez da uste denborarik izango denik hau egiteko; bigarren hobekuntza osorik burutzea ere zail da.

5.2.2 Egindako hobekuntzak

- **Kode txukunketa**

  Kontrol programa argiago eta ulerterrazago egite aldera, 3 liburutegi garatu dira hobekuntza gisa: Motorrak.h (Motorren kontrolerako), Wifi.h (Wifi *Shield*-a erabiltzeko) eta GPS.h (GPS modulua erabiltzeko). Jarraian liburutegi hauek gainetik azalduko dira garatutako funtzioak zein diren aipatuz.

Motorrak.h

Liburutegi hau motorren kontrolerako garatu da. Honen bidez 2 motor (edo 2 motor multzo) modu independentean kontrolatu daitezke ohiko eragiketak aginduz: aurrera, atzera, eskuinera, ezkerrera... Erabiliko den mugimendu modua diferentziala izango da. Jarraian ikus daiteke liburutegi honen definizio fitxategia:

```
class Motorrak
{

  public:
    Motorrak(int esk_norantza, int esk_abiadura, int ezk_norantza, int ezk_abiadura, int abiadura);
    void aurrera();
    void atzera();
    void ezkerrera();
    void eskuinera();
    void aurreraEskuinera();
    void aurreraEzkerrera();
    void atzeraEskuinera();
    void atzeraEzkerrera();
    void gelditu();
    void aldatuAbiadura(int abiadura);

  private:
    int _esk_norantza;
    int _esk_abiadura;
    int _ezk_norantza;
    int _ezk_abiadura;

    int _abiadura;

};
```

54.irudia: Motorrak.h

Funtzio guztien izenak nahiko adierazgarriak direnez, ez dira banan-banan azalduko. Bestalde, ikus daitekeen bezala, funtzio eraikitzaileak parametro asko jasotzen ditu; parametro hauek motorrak konektatuko diren hankatxoei dagozkie, bi alde bakoitzeko motor bakoitzeko: M1, E1 eta M2, E2. Bestalde, motorren hasierako abiadura ere finkatuko da azkena jasotako parametroarekin.

Wifi.h

Liburutegi hau DFRoboteko Wifi *shield*-a erabiltzeko garatu da. Ikusi den moduan *shield* hau AT komando bidez erabiltzen da eta komando hauek gogoratzea zaila denez, funtzio desberdinak idatzi dira komando hauei dei egiteko.

Shield honek serieko komunikazioa darabil, eta honetarako mikrokontrolagailuaren berezko Rx eta Tx hankatxoak erabiltzen ditu. Beraz, ondoren programa nagusitik edota beste liburutegi batetik zerbait inprimatu nahi bada, nahikoa izango da ohiko Serial.print() funtzioa erabiltzea.

Hona hemen liburutegi honen funtzioen zerrenda:

```
class Wifi
{
      public:
            Wifi();
            void AP_izena(String izena);
            void AP_pass(String pass);
            void dhcp(int n);
            void helbidea_esleitu(String helbidea, String maskara, String gateway);
            void socketa_sortu(String rol, String protokoloa, String helbidea, String portua);
            void hasi_komunikazioa();
            void bidali(String mezua);
            void deskonektatu();

};
```

55.irudia: Wifi.h

GPS.h

Liburutegi hau DFRoboten GPS/GSM/GPRS modulua erabili ahal izateko garatu da. Bertan, modulua GPS modu autonomoan erabili ahal izateko eta honek bidaltzen dituen datuak prozesatzeko beharrezko funtzio guztiak daude.

Hona hemen GPS klaseko funtzioak:

```
class GPS
{
      public:
            GPS(int p1, int p2, int p3);
            void konfiguratu();
            void UTC();
            void latitudea();
            void latitudea_norantza();
            void longitudea();
            void longitudea_norantza();
            void altitudea();

      private:
              int _p1;
              int _p2;
              int _p3;

};
```

56.irudia: GPS.h

GPS objektuaren hiru atributuak GPS/GSM moduak aukeratzeko erabiliko diren hiru hankatxoak dira; hau da, GPSko 3, 4 eta 5 hankatxoak mikrokontrolagailuko zein hankatxorekin konektatuko diren hurrenez hurren.

Konfiguratu() funtzioaren bidez modulua GPS modu autonomoan konfiguratuko da eta gainontzeko klaseko funtzioak GPSak lortzen dituen datuak eskuratzeko dira: ordua, latitudea eta honen orientazioa eta longitudea eta honen orientazioa. Bestalde, GPS.cpp fitxategian nahi diren datuak eskuratzeko beharrezko funtzio guztiak daude.

Liburutegi hauek idazten 10 ordu behar izango zirela aurreikusi zen, baina, azkenean 2 ordu gehiago behar izan dira, liburutegi guztiak bateratzerako orduan sortu diren arazo txikiak direla eta. Honen ondorioz, hurrengo hobekuntzari aurre egiteko 8 ordu soilik geratu dira. Denbora honetan ez da posible identifikatutako beste bi hobekuntza horietako bakar bat ere egitea; beraz, geratzen diren ordu hauek honetarako erabili dira: 6 ordu memoria eta robotaren azken xehetasunak burutzeko eta geratzen diren 2 orduak defentsarako erreserbatu dira badaezpadako ordu gisa.

5.3 ARAZOAK

Proiektuan zehar ez da arazo larririk izan; beraz, atal honetan aipatuko direnak arazo baino gehiago izandako oztopoak dira.

Desbiderapen handienak eragin dituzten arrazoiak jada aipatu dira jarraipen eta kontrolaren atalean: eraikuntzaren estimazio okerra eta dependentziak, GPSa erabiltzeko beharrezko baldintzak, osagai guztiak bateratzerako garaian izandako arazoak eta serieko komunikazio ugari erabiltzearen ondorioak (*software serial* liburutegia).

Lan gehiago eragin duen beste arrazoi bat informazio kontua izan da. Erabilitako osagai guztiak DFRobot etxekoak ziren, ez Arduinorenak. Hau dela eta, osagai guztiekin egoera antzekoa izan da: askoz ere informazio gehiago zegoen Arduinoren osagaientzako DFRoboteko osagaientzako baino; gainera, kasu gehienetan Arduinoren *shield*-ek, erraztasunak emate aldera, liburutegiak dituzte. Kontu hau hasiera batean oztopo gisa hartu zen arren, azkenean zerbait positibo bezala ikusi da. Izan ere, erabilitako osagai bakoitzaren inguruan asko ikasi behar izan da, eta derrigorrezkoa izan da funtzionamendua ongi ezagutu eta ulertzea programatu ahal izateko. Azkenik, osagai hauentzako liburutegiak ere idatzi dira.

Azkenik, arazo nagusietako bat paralelizazioa izan da. Robotaren garapenarekin hastean osagai desberdinekin hari desberdinak erabiltzea pentsatzen zen, hau da: hari bat motorren kontrolerako, beste hari bat ultrasoinu sentsorearentzat eta beste bat

GPSarentzat, uneoro balioak bidaltzen aritzeko. Baina, honen inguruko informazioa aurkitzean erabiltzen dudan mikrokontrolagailuak ez duela honelako aukerarik eskaintzen jakin da; beraz, ezinezkoa izan da paralelizatzea.

6. ONDORIOAK ETA EMAITZAK

6.1 ONDORIOAK

Ikusi den bezala, proiektuaren hasieran finkatutako helburuak lortu dira: Kanpoan ibiliko den robot bat eraiki da honako ezaugarri hauekin:

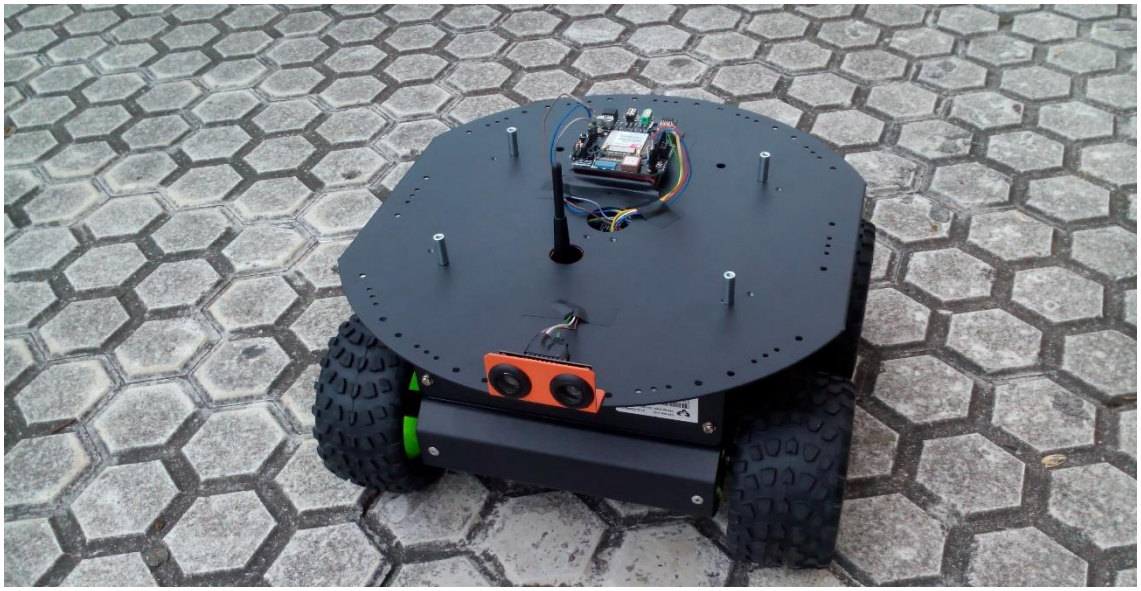
- Bluetooth bidezko konektibitatea robota kontrolatu ahal izateko
- Ultrasoinu sentsorea oinarrizko segurtasun neurri gisa
- GPSaren bidezko lokalizazioa

Honetaz gain, hainbat hobekuntza ere egin ahal izan dira; batetik, robotaren osagaientzako liburutegiak; eta bestetik, wifi bidezko konektibitatea. Hobekuntza gehiago egin ez badira ere, hauen inguruan hausnartu da, bai eta hainbat proposamen burutu ere. Hasieran aipatu bezala, robot hau oinarri gisa harturik, etorkizunean hainbat hobekuntza egingo zaizkio; beraz, honi esker, egin litezkeen zenbait hobekuntza posible jada proposatuta daude.

Produktuaren helburuak ez ezik, proiektuarenak ere lortu dira. Robotika mugikorraren inguruan gehiago sakontzea lortu da proiektu honen bidez; bai eta gai honekiko interesa areagotzea eta etorkizunean gai honen inguruan ikasten jarraitzeko gogoa sustatzea ere.

6.2 EMAITZAK

57.irudian robotaren azken emaitza ikus daiteke:



57.irudia: Robota

Bestalde, robotaren funtzionamendu zuzena frogatze aldera, bideo bat grabatu da, non robotaren funtzionalitate eta ezaugarri guztiak ikus daitezkeen.

7. BIBLIOGRAFIA

Orokorrak

Aitzol Astigarraga eta Elena Lazkano. (2011). *Robot Mugikorrak. Oinarriak*

John-David Warren, Josh Adams, Harald Molle. (2011). *Arduino Robotics*

Robotika & sistema autonomoen ikerketa taldea (RSAIT)
Webgunea: http://www.sc.ehu.es/ccwrobot/seccion/sarrera

DFRobot. Drive the future (2008)
Webgunea: http://www.dfrobot.com

Arduino (2005)
Webgunea: http://www.arduino.cc/

Memoriaren ezaugarri eta betebeharrak

Webgunea: http://www.ehu.eus/eu/web/informatika-fakultatea/ikasketak/gradu-amaierako-proiektua

Txantiloia KAPen txostena egiteko

Kudeaketa

Project Management Institute. (2008). *A guide to the project management body of knowledge (PMBOK guide).*

Romeo-All in one

DFRobot. (2014). *Romeo-All in one*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=656#.VX8rFvntmko

DFRobot. (2012). *Romeo-All in one. Wiki*
Webgunea: http://www.dfrobot.com/wiki/index.php?title=DFRduino_Romeo-All_in_one_Controller_V1.1(SKU:DFR0004)

Ro-botica. *Controladora DFRobot Romeo-Todo en una*
Webgunea: http://ro-botica.com/es/Producto/DFRobot-Romeo-1.1-compatible-ARDUINO/

Motorrak

DFRobot. (2014). *2A Motor Shield for Arduino*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=69#.VX_bqvntmkp

DFRobot. (2015). *Arduino Motor Shield. Wiki*
Webgunea:
http://www.dfrobot.com/wiki/index.php?title=Arduino_Motor_Shield_(L298N)_(SKU:DRI0009)

Hobbyist. *Motor Shield Tutorial*
Webgunea: http://www.hobbyist.co.nz/?q=motor-shield-tutorial

Bluetooth

Andi.Co. (2014). *Arduino Bluetooth RC Car*
Webgunea:
https://play.google.com/store/apps/details?id=braulio.calle.bluetoothRCcontroller&hl=es

DFRobot. (2014). *DFRobot bluetooth V3*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=360#.VX_fN_ntmkp

DFRobot. (2013). *DF-BluetoothV3 Bluetooth module. Wiki*
Webgunea: http://www.dfrobot.com/wiki/index.php?title=DF-BluetoothV3_Bluetooth_module_(SKU:TEL0026)

Techbitar. (2013). *Modify The HC-05 Bluetooth Module Defaults Using AT commands*
Webgunea: http://www.instructables.com/id/Modify-The-HC-05-Bluetooth-Module-Defaults-Using-A/

Jose Antonio Luceño Castilla. (2013). *Conectar Android con Arduino por Bluetooth Capítulo 1*
Webgunea: https://www.youtube.com/watch?v=Is6pJyEv8s0

Jose Antonio Luceño Castilla. (2013). *Conectar Android con Arduino por Bluetooth Capítulo 2*
Webgunea: https://www.youtube.com/watch?v=7ciWMxi4sSU

Ultrasoinua

DFRobot. (2014). *URM37 V4.0 Ultrasonic Sensor*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=53&search=ultrasonic+sensor&description=true

DFRobot. (2015). *URM37 V4.0 Ultrasonic Sensor. Wiki*
Webgunea:
https://www.dfrobot.com/wiki/index.php?title=URM37_V4.0_Ultrasonic_Sensor_(SKU:SEN0001)#Introduction

GPS/GSM/GPRS modulua

DFRobot. (2014). *GPS/GPRS/GSM Shield V3.0 (Arduino Compatible)*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=673#.VX_hm_ntmkp

DFRobot. (2015). *GPS/GPRS/GSM Module V3.0. Wiki*
Webgunea:
http://www.dfrobot.com/wiki/index.php/GPS/GPRS/GSM_Module_V3.0_(SKU:TEL0051)

SIMCom. (2011). *SIM908 AT Command Manual_V1.01*
Pdf dokumentua:
http://www.dfrobot.com/image/data/TEL0051/3.0/SIM908_AT%20Command%20Manual_V1.01.pdf

Wifi shield

DFRobot. (2014). *Wifi Shield V2.2 for Arduino (802.11 b/g/n)*
Webgunea:
http://www.dfrobot.com/index.php?route=product/product&product_id=548#.VX_i-_ntmkp

DFRobot. (2013). *WiFi Shield V2.2 For Arduino. Wiki*
Webgunea:
http://www.dfrobot.com/wiki/index.php/WiFi_Shield_V2.2_For_Arduino_(SKU:TEL0047)

WIZnet Co. (2011). *WizFi210/220 User Manual (Version 1.12)*
PDF dokumentua: http://www.dfrobot.com/image/data/TEL0047/WizFi210-User_Manual_EN_V1.12.pdf

WIZnet Co. (2011). *WIzFi210 Quick Start Guide (Version 1.0)*
PDF dokumentua: http://www.dfrobot.com/image/data/TEL0047/WizFi210-QuickStartGuide_EN_V1.0.pdf

WIZnet Co. (2013). *WIZnet User forum. WizFi210/220*
Webgunea: http://wizwiki.net/forum/viewforum.php?f=31

8. ESKERTZAK

Proiektu honen arrakastaren atzean faktore ugari daude; baina garrantzitsuenak ondoan izandako laguntzaileak izan dira. Beraz, ez nuke nahi proiektu hau amaitutzat jo hauei emandako laguntza eskertu gabe.

Lehenik eta behin, eskerrik asko Txelo Ruiz eta Elena Lazkano proiektu osoan emandako laguntza guztiagatik eta behar izan dudan guztian ondoan egoteagatik. Bestalde, eskerrak eman nahiko nizkioke Igor Rodriguezi ere robotaren eraikuntzan emandako laguntzagatik 3D inprimagailuarekin piezak eginez. Azkenik, Borja Gamecho ere aipatu nahiko nuke bluetooth kontuekin lagundu izanagatik.

Proiektu polit hau testuinguru eder batean aurrera eramateak asko erraztu du egin beharreko lana. Mila esker hau eragin duzuen guztioi.

9. ERANSKINAK

A) GIDALIBURUA

Atal honetan robota martxan jarri eta funtzionalitate guztiak aprobetxatuz erabiltzeko gidalerro guztiak daude, pausoz pauso azaldurik.

1. Konfiguratu

Robotean lehenengo aldiz programa kargatu baino lehen, hainbat konfigurazio parametro zehaztu behar dira kontrol programan. Konfigurazio parametro hauek zein diren eta hauen esanahia 20.taulan ikus daiteke:

Parametroa	*Esanahia*	*Adibidea*
WIFI		
AP_izena	Wifi sarearen AP izena	"nireAP"
AP_pass	Wifi sarearen pasahitza	"123456789"
zerb_helb	Echo zerbitzariaren IP@	"192.168.43.211"
zerb_portua	Echo zerbitzariaren portua	"192.168.43.211"
GPS		
Irakurketa_kop	Zenbat GPS irakurketa egin lokalizazioa egiten denean	30

20.taula: Programako parametro konfiguragarrien azalpena

Ikus daitekeen bezala konfiguratzeko parametro guztiak Wifi eta GPS funtzionalitateetarako dira. GPSaren kasuan komenigarria da ongi azaltzea parametro honen garrantzia. Robota lokalizazioarekin hastean, irakurketak egiten hasten da sateliteetatik jasotako mezuen bidez. Hasierako irakurketen emaitzek ez dute baliorik ematen, eta irakurketa gehiago egin ahala balioak inprimatuko ditu eta hauek gero eta zehatzagoak izango dira irakurketa gehiago egitean.

Parametro hau finkatzerako orduan hainbat proba egin dira eta azkenean 30 irakurketetan utzi da. Halere, proba guztietan 20-23 irakurketa nahikoa izan dira balio zuzenak lortzeko. Baina, egoeraren eta lekuaren arabera balio optimoa aldatu egiten denez, parametro konfiguragarri gisa uztea erabaki da. Kontuan izan behar da irakurketa gehiago egiteak robota denbora gehiagoz geldi egotea suposatzen duela.

Konfigurazio parametro hauek zehaztu ahal izateko, programaren fitxategiaren hasieran egin behar dira aldaketak; 58.irudian ikusten den tokian:

```
/*PARAMETRO KONFIGURAGARRIAK*/

//GPS parametroa
int irakurketa_kop=30;

//Wifi parametroak
String AP_izena="nireAP"; //Konektatu nahi dugun AParen izena
String AP_pass="123456789"; //AParen pasahitza
String zerb_helb="192.168.43.211";//echo zerbitzaria dagoen makinaren IP helbidea
String zerb_portua="50001"; //echo zerbitzariaren portua
```

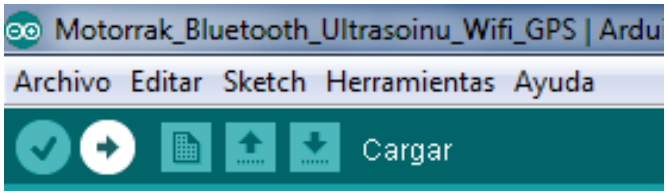
58.irudia: Programako parametro konfiguragarriak

Behin parametro hauek zehaztuta, lehenengo pausoa amaitutzat joko da.

2. Programa konpilatu eta kargatu

Behin konfigurazio parametroak zehaztuta, programa konpilatu eta mikrokontrolagailuan kargatu behar da. Programa kargatu ahal izateko, lehenik eta behin, robotaren wifi *shield*-aren *switch* bat aldatu behar da; Prog/Run *switch*-a hain zuzen ere. Gogora dezagun, *shield* honek Rx eta Tx serie lerroak erabiltzen dituenez, programak kargatzerakoan arazoak sortzen direla. Hau dela eta, programa kargatzerakoan *switch* hau Prog posizioan jarri behar da.

Behin hau eginda, programa ireki Arduino IDE bidez, eta kargatzeko botoia sakatu behar da:



59.irudia: Arduino IDE-Programa kargatzeko botoia

Programa kargatutakoan, wifi *shield*-eko Prog/Run *switch*-a berriro ere Run posizioan utziko da, programaren exekuzio zuzena burutzeko.

3. Konexioak egin

Ultrasoinu sentsorea eta GPS/GSM/GPRS modulua kableen bidez daude konektaturik mikrokontrolagailura. Kable hauetakoren bat gaizki jartzeak robotaren exekuzio okerra eragingo du. Beraz, garrantzitsua da kable hauek nola jarri behar diren jakitea eta ondo jarrita daudela ziurtatzea.

Hona hemen kasu bakoitzerako zein koloretako kableak dauden eta zein bi hankatxorekin konektatu behar diren:

Ultrasoinua:

Kableak	*Mikrokontrolagailua*	*Ultrasoinu sentsorea*
Txuria	GND	GND
Marroia	5V	5V
Berdea	2	Comp/Trig

21.taula: Ultrasoinu sentsorearen konexioak

GPS/GSM/GPRS modulua

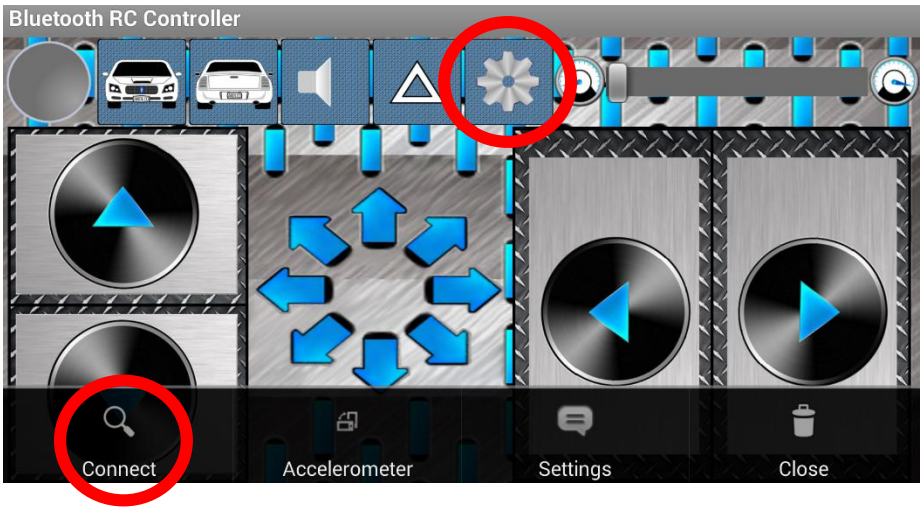
Kableak	*Mikrokontrolagailua*	*GPS/GSM/GPRS*
Grisa	Vin	Vin
Urdina	GND	GND
Marroia	RST	RST
Urdina	8	Rx
Gorria	12	Tx
Horia	3	3
Txuria	9	4
Berdea	13	5

22.taula: GPS/GSM/GPRS moduluaren konexioak

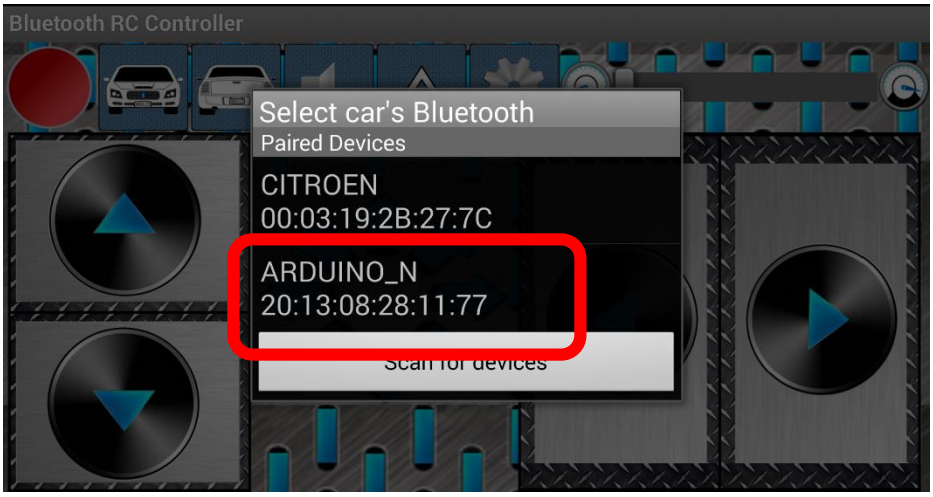
4. Martxan jarri

Robota martxan jartzeko honako pauso hauek eman behar dira:

1. Ordenagailua erabiliko den wifi sarera konektatu
2. Ordenagailu horretan echo zerbitzaria martxan jarri
3. Robota piztu (etengailua aktibatu)
4. Bluetooth RC Car aplikazioa ireki eta robotarekin konexioa ezarri: *Zehaztapenak – Konektatu – Arduino_N*

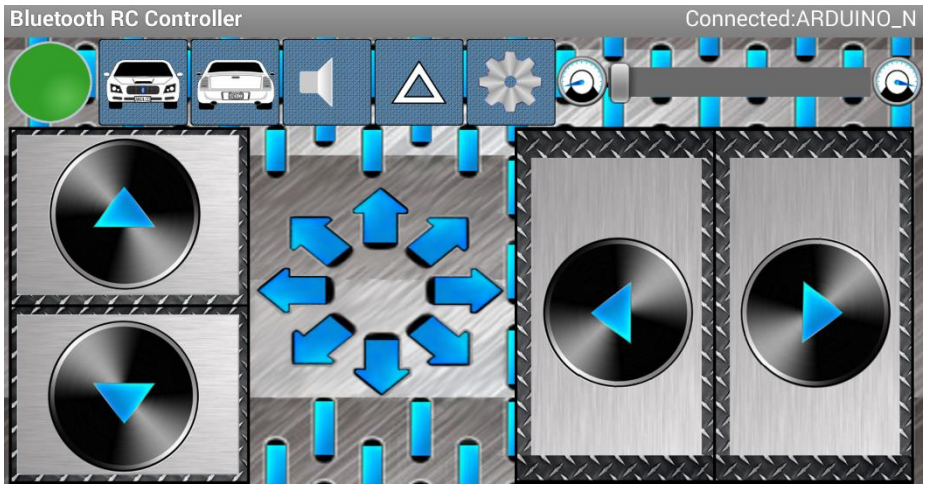


60.irudia: Robotarekin bluetooth bidezko komunikazioa ezarri 1.pausoa



61.irudia: Robotarekin bluetooth bidezko komunikazioa ezarri 2.pausoa

Lehen aldiz konektatzean, bluetooth moduluaren pasahitza eskatzen da eta hau "1234" da. Konexioa ongi ezartzean, pantailan goian ezkerretara dagoen biribila berde kolorekoa jartzen da; komunikazio zuzenaren adierazgarri:



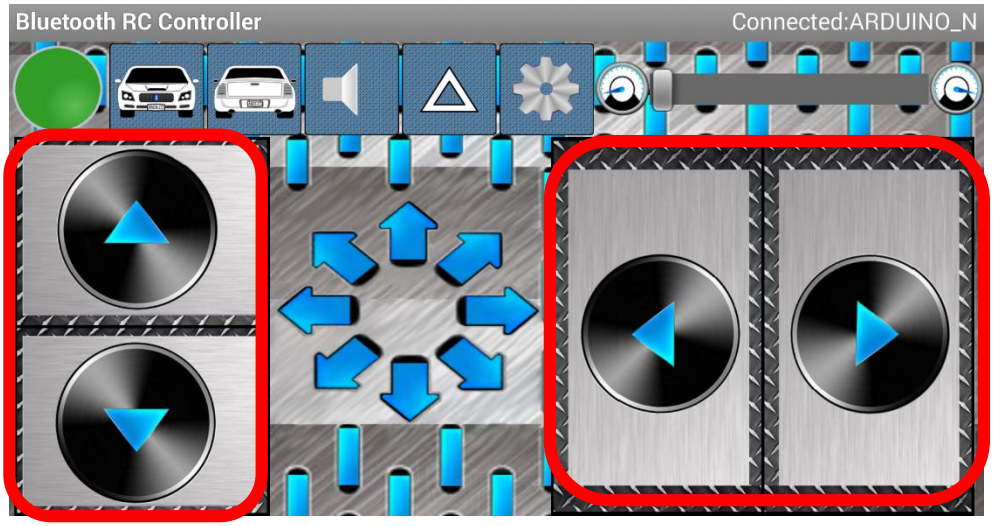
62.irudia: Bluetooth RC Car-Bluetooth komunikazioa finkatuta

Lau urrats hauek emanda, jada robota martxan eta erabiltzeko prest egongo da.

5. Erabili

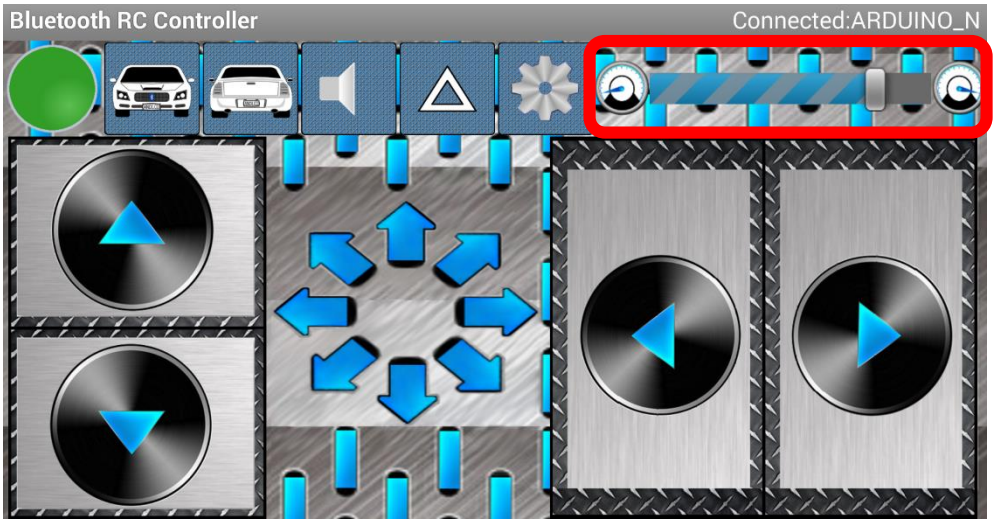
Behin dena prest izanda, geratzen den bakarra robota erabiltzea da. Honetarako aplikazioak bi aukera eskaintzen ditu aurrez aipatu bezala: botoien bidez edo azelerometro bidez. Bi modutara egin daitezkeen ekintzak berdinak dira:

- Norantza gezien bidez



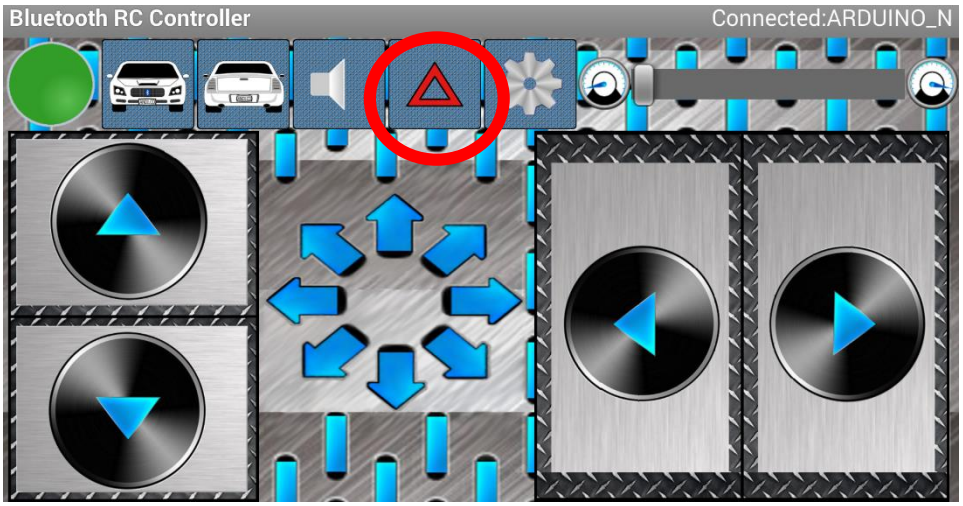
63.irudia: Bluetooth RC Car-Gezien bidez robota mugiarazi

- Abiadura aldaketa



64.irudia: Bluetooth RC Car-Abiadura aldatu

- GPS bidezko lokalizazioa



65.irudia: Bluetooth RC Car-GPS bidezko lokalizazioa

Robota martxan dagoen bitartean, bere egoeraren berri emateko mezuak bidaliko ditu echo zerbitzarira. Beraz, uneoro jakingo dugu zein den robotaren egoera. Azkenik, exekuzioarekin amaitzeko nahikoa izango da aplikazioa ixtea edota deskonektatzeko aukera sakatzea (konektatu garen bezalaxe).

B) ROBOTAREN KONTROLERAKO FITXATEGIAK

Proiektuan zehar robotaren kontrolerako hainbat fitxategi garatu dira: kontrol programak, konfigurazio programak, liburutegiak eta *echo* zerbitzaria. Hasieran aipatu bezala, proiektu honetatik abiatuz, aurrerantzean gauza gehiago egingo dira robot honekin; beraz, prozesu horretan lagungarri suerta daitezkeen fitxategi guztiak memoriarekin batera entregatu dira.

Eranskin honetan, fitxategi hauek zein diren eta bakoitzarekin zer egin behar den eta egin daitekeen azalduko da.

Kontrol programak

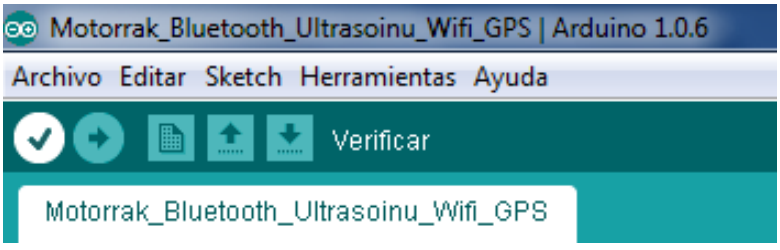
Aipatu bezala, robotari osagaiak pixkanaka-pixkanaka gehitu zaizkio; bai eraikitzerakoan bai eta programatzerakoan ere; bi ataza hauek paraleloan eramanez. Hortaz, azken emaitza lortu arte hainbat kontrol programa idatzi dira, eta urrats bakoitzeko programak gorde dira. Beraz, azkenean, lau kontrol programa daude erabilgarri:

- Motorrak_Bluetooth
- Motorrak_Bluetooth_Ultrasoinu
- Motorrak_Bluetooth_Ultrasoinu_Wifi
- Motorrak_Bluetooth_Ultrasoinu_Wifi_GPS

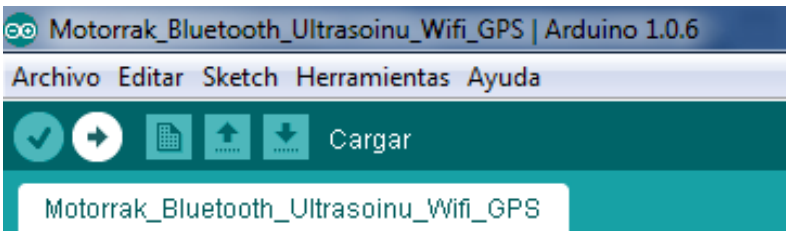
Modu honetan, robota osagai batzuekin bakarrik erabiltzeko aukera eskaintzen da eta hau robotaren erabileraren arabera, erosoa suerta liteke.

Programa hauek idatzi, konpilatu eta mikrokontrolagailuan kargatzeko Arduino softwarea erabili da; software hau Arduinoren webgune ofizialean bertan dago eskuragarri eta doakoa da.

Kargatu nahi den programa Arduino IDErekin ireki eta hau konpilatu eta kargatzeko honako botoi hauek sakatu behar dira:



66.irudia: Arduino IDE-Programa konpilatzeko botoia



67.irudia: Arduino IDE-Programa kargatzeko botoia

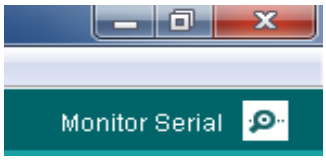
Beraz, erabili nahi diren robotaren osagaien arabera kontrol programa bat aukeratu, ireki, konpilatu eta USBa ordenagailura konektatuz, mikrokontrolagailuan kargatu behar da.

Konfigurazio programak

Konfigurazio programei dagokienez, bi konfigurazio programa entregatu dira: bat bluetooth modulura komandoak bidaltzeko eta bestea ultrasoinuaren muga balioa konfiguratzeko.

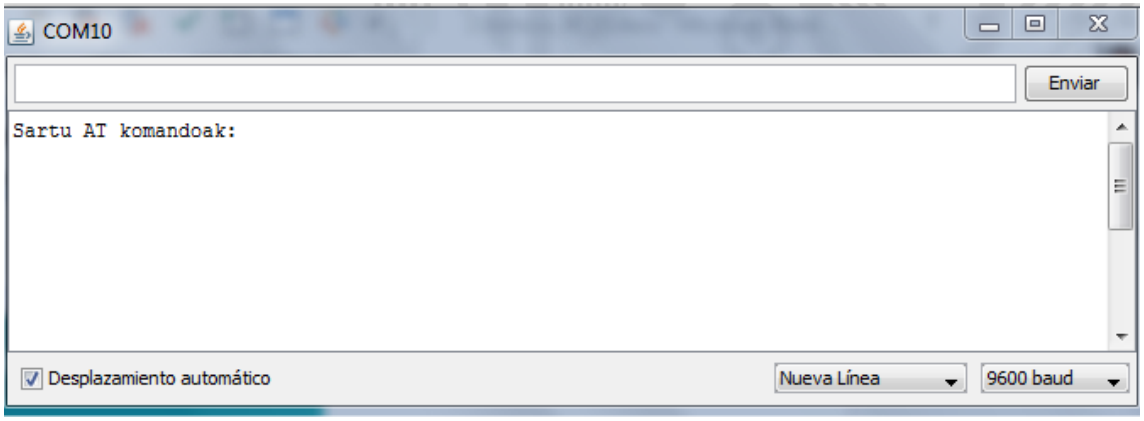
Bluetooth-ari dagokionez, aipatu bezala, AT komandoen bidez konfiguratzen da. Hau, programarik kargatu gabe ere egin daiteke, serieko terminal bat ireki eta zuzenean komandoak bidaliz. Baina, hau egin ahal izateko, bluetooth-aren serieko hankatxoak mikrokontrolagailuko Rx (0) eta Tx (1) hankatxoekin lotuta egon behar dira; eta robotean ez daude horrela. Hau dela eta, beharrezkoa da programa bat idaztea, komandoak bluetooth-aren serie lerrora bideratzeko.

Mikrokontrolagailua ordenagailura konektatu, programa hau kargatu eta serieko monitorea ireki behar da komandoak bidali ahal izateko.



68.irudia: Arduino IDE-Serieko monitorea

Ondoren, serieko komunikazio abiadura finkatu beharko dugu bluetooth moduluaren abiadura berbera jarria: 9600. Behin hau eginda, komandoak bidaltzen has gaitezke, modulua guk nahi bezala konfiguratuz.



69.irudia: Arduino IDE-Serie komunikazioa bluetooth moduluarekin

Ultrasoinuaren konfigurazioari dagokionez, hau ez da bluetooth-arena bezain erosoa; ultrasoinua konfiguratzeko honen serie lerroak erabili behar direlako komandoak bidaltzeko eta lerro hauek robotak erabiltzen ez dituenez ez daude konektaturik.

Beraz, ultrasoinua konfiguratu nahi izanez gero, honen Rx eta Tx lerroak 11 eta 10 pin-etara konektatu behar dira, urrenez urren. Behin hau eginda, entregatutako konfigurazio programa hartu eta bertan dauden komandoak aldatu behar dira muga balio berria ezartzeko ultrasoinuari. Honetarako, ultrasoinuaren laneko materialen atalean eta garapeneko atalean zehaztutako pausoak jarraitu behar dira.

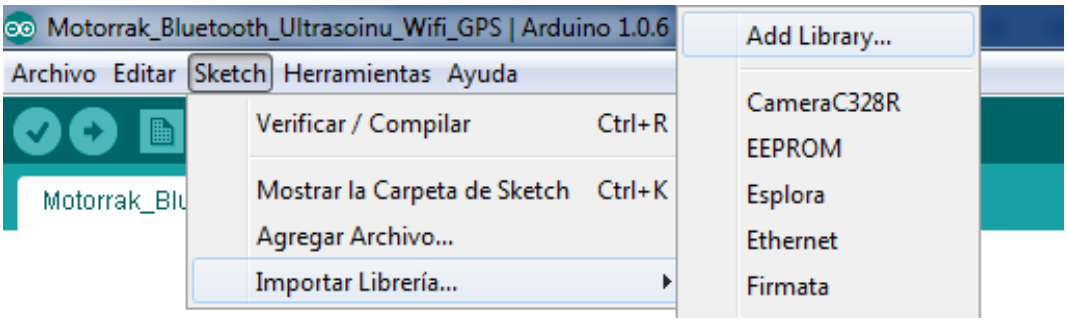
Behin dena prest izandakoan, programa konpilatu, kargatu eta exekutatu behar da. Hau eginez gero, jada konfiguraturik izango dugu ultrasoinu sentsorea muga balio berriarekin.

Ultrasoinua konfiguratzeko bi programa entregatu dira: bat 16 cm-ko muga balioa ezartzeko; eta bestea, 50 cm-koa.

Liburutegiak

Hobekuntzen atalean esan bezala, hobekuntza gisa hiru liburutegi idatzi dira: motorrak.h, wifi.h eta GPS.h. Entregatutako kontrol programak mikrokontrolagailuan kargatu ahal izateko, beharrezkoa da liburutegi hauek eskura izatea; eta kontrol programarekin batera, mikrokontrolagailuan kargatzea.

Honetarako, honako hau egin behar da:



70.irudia: Arduino IDE-Liburutegi bat inportatu

Behin *Add Library…* sakatuta nahi dugun liburutegia aukeratuko dugu eta inportatutako liburutegia jada *Importar Librería…* aukeran agertuko da; nahi dugunean erabili ahal izateko.

Echo zerbitzaria

Azkenik, wifi bidezko komunikaziorako *echo* zerbitzaria ere entregatu da. Zerbitzari hau, gogora dezagun, wifi bidezko komunikazioa izan nahi bada, robota konektatuko den sare berean jarri behar dela martxan eta zerbitzari honen IP helbidea eta portua konfiguratu kontrol programan. Zerbitzari hau konpilatu eta martxan jartzeko honako bi komando hauek erabiltzea nahikoa da:

```
gcc  –o  zerbitzaria   echo_udp_zerb.c    // Konpilatzeko

./zerbitzaria                             // Exekutatzeko
```

23.taula: *echo* zerbitzaria konpilatu eta exekutatzeko komandoak