eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Konputagailuen Ingeniaritza

Gradu Amaierako Proiektua

Arduino plataforma mugikor baten erabilera Raspberry Pi eta ROS bitartez

---

Egilea

Goiatz Irazabal Rodriguez

Tutoreak

Elena Lazkano eta Txelo Ruiz

informatika fakultatea
facultad de informática

2015eko abuztuaren 30a

Eskertzak

Lau urte pasa dira dagoeneko unibertsitate munduan sartu nintzenetik eta lehenengo garai honi amaiera emateko ordua iritsi da. Asko izan dira lau urte hauetan ikasitakoak, bai informatikaren inguruan eta bai informatika mundutik kanpo. Asko izan dira bizitako esperientziak, ezagututako lekuak eta ezagututako lagunak.

Lehenik eta behin eskerrak eman nahi dizkiet Txelo Ruiz eta Elena Lazkanori, proiektu hau burutzeko gonbita luzatzeagatik eta proiektu osoan zehar eskainitako laguntzagatik. Nola ez, baita ere eskerrak eman RSAIT taldeko kideei, bai proiektu honetan eta aurretik egindakoetan lagundu izanagatik, bereziki Aitzol Astigarraga eta Igor Rodriguezi.

Nola ez, nire guraso eta familiari, urte guzti hauetan eskainitako laguntzagatik eta ikasketak burutzeko erraztasunak emateagatik, bai hemen eta bai atzerrian.

Urte guzti hauetan nire egunerokoaren zati izan diren ikaskide, irakasle, AZP eta unibertsitate bizitza osatu duten gainerako pertsona guztiei ere, bai Donostian eta bai Erroman, eskerrak eman nahi dizkiet: Aitor, Alex, Aratz, Asier, Davide, Eneko, Ezekiel, Francesca, Guillermo, Lierni, Maider, Manuel, Nerea, Xabi... eta bereziki, Oihane Parrari, lau urte hauetan batera jarraitutako ibilbidean ikaskide, bidaialagun, pixukide eta batez ere lagun izateaz gain une oro eskainitako laguntzagatik.

Ezin ahaztu unibertsitate bizitzako lehenengo bi urtetan nire egunerokoaren zati handi bat bete zuten IKD Gazte taldeko kideei eta nola ez bertan sartzeko aukera eman zidatenei. Ikaragarria izan zen bi urte haietan ikasitakoa. Bertan guk sortu genuen proiektuak urtez urte aurrera jarraitzen duela ikusteak izugarrizko poztasuna ematen du.

Bestalde, eskerrak eman azken bi urteetan Informatika Fakultateko Ikasle Kontseiluaren parte izateko nigan konfiatu zuten guztiei eta nirekin batera kontseilua osatu duten pertsonei. Batez ere Ikubo taldeko kideei: Adrian, Alex, Mikel eta Uxueri, besteak beste, eta batez ere guzti honetan parte hartzera bultzatu zidan Gorka Maiztegiri.

Eskerrak eman ere, nola ez, RITSI elkartean ezagututako pertsona guztiei (gehiegi dira zerrendatu beharreko pertsonak). Bakoitza iberiar penintsulako leku ezberdin batean bizi arren, talde bat osatzen dugulako. Beraiengandik asko ikasi eta Informatika Ingeneritzako ikasleen eskubideen alde lan egiten jarraitzen dugulako.

Garatu Sistemas Informáticos enpresari eta bertako lankideei ere eskerrak eman behar dizkiet, praktikak egiten pasatako lau hilabeteetan erakutsi eta lagundutako guztiagatik, lan mundua ezagutzeko aukera eman eta ningan konfiatzeagatik.

Azkenik, mila esker zuri, proiektuarekiko interesa izan eta ondorengo orrialdeak irakurtzeagatik.

Laburpena

Txosten honetan azaltzen den proiektua, auto itxurako robot baten eraikuntzan oinarritzen da. *Arduino* plataforman oinarritutako robot mugikor baten sorkuntza burutuko da hutsetik, honen kontrola *Raspberry Pi* ordenagailu txikiaren bitartez eginez. Gainera, azken gailu honi ahalmen handiagoa emateko asmoz, *ROS* plataforma instalatuko da bertan. Duten kostu baxua dela eta, gaur egun izugarrizko arrakasta lortu dute plataforma guzti hauek, baina proiektu honetan beraien ahalmena neurtu nahi dugu ezaugarri zehatz batzuk dituen robota sortuz.

Gaien aurkibidea

Irudien aurkibidea

Taulen aurkibidea

1. KAPITULUA

Aurkezpena

Aurkibidea

1.1 Sarrera

Gaur egun, jakina da robotikaren munduak duen berebiziko garrantzia. Izan ere, hainbat lekutan, robotek helburu sozialak dituzte eta gizakiei laguntza eskeintzeko balio duten gailuak dira. Bere garrantzia handia izanik, ez da erraza robot bat eskuratzea, izan ere duten kostua nahiko altua izaten da gehienetan.

Beste aukera bat, robota norberak egitea da, plataforma merkeez baliatuz robotaren kostua oso altua izan ez dadin. Jarraian azalduko den proiektuan, *Arduino* [1] eta *Raspberry Pi* [2] izan dira hardwarea osatu duten kode irekiko bi plataformak.Hasiera batean irakaskuntzarako sortuak izan baziren ere, zer jasateko gai dira plataforma hauek? Nahiko ote dira behar bezalako robot bat sortzeko?

1.2 Motibazioa

Jarraian azalduko den proiektua, *Donostiako Informatika Fakultateko RSAIT Robotika eta Sistema Autonomoen Ikerketa taldeak* [3] proposatutako proiektu bat da, talde honen ikerkuntza alor nagusia robotika mugikorra izanik.

Beraien esku dituzten robot guztiak komertzialak izanik, hau da, robotak dagoeneko eraikita erosi direnez, norberak hutsetik sortutako eta kode irekiko hardwarean oinarritutako robot baten boterea ikusi nahi da. Honela, aurrerago, proiektu hau aurten burutu diren antzeko beste proiektu batzuekin uztartuz robot ahalmentsuago bat sortzeko.

Bestalde, ezin ahaztu ikaskuntzaren aldetik datorren motibazioa. *Arduino*, *Raspberry Pi* eta *ROS*-ekin lan egiteak, gaur egun gori gorian dauden plataformekin lan egitea suposatzen du, beraien inguruko informazioa bilatuz, lan egiteko era ulertuz eta beraien ahalmena neurtuz.

1.3 Proiektuaren deskribapen orokorra

Aipatu bezala, dokumentu honetan azaltzen den proiektua robot baten eraikuntzan oinarritzen da. Robotak auto itxura izango du eta kode irekiko plataformatan oinarrituko da. Hardwareari dagokionez *Makeblock* etxeko *Starter Robot Kit V1.0* (*Arduino* txartela eta hainbat piezaz osatua) eta *Raspberry Pi* bat erabiliko dira. Softwarearen aldetik, *Ar-*

duino eta *Raspberry Pi* plataformek erabiliko duten oinarrizko softwareaz gain, roboten munduan aski ezaguna den *ROS* plataforma erabiliko da.

Robotak ohiko ezaugarriak izango ditu eta oinarrizko funtzioak bete, izan ere proiektuaren helburua kode irekiko plataformen ahalmena neurtzea da. Bereziki, *Raspberry Pi* ordenagailu txikiaren ahalmena neurtu nahi da bertan *ROS*-ekin lan eginez.

Alde batetik, *Arduino* txartelaren bidez infragorri eta motorren kontrola burutzeko programa bat egingo da. Honi esker, autoak autonomoki marra beltz bat jarraituko du. Ondoren, *Arduino* txartelari *Raspberry-Pi* txartelaren ahalmena gehituko zaio. Horrela, *Raspberry Pi*-a izango da erabiltzailearen aginduak jasoko dituena teklatuaren bidez eta agindu hauek *Arduino* txartelera bidali motorrak nahi ditugun norabidean mugitzeko, auto telegidatu bat sortuz.Azkenik, *USB* bidezko kamera bat gehituko diogu robotari eta *ROS* softwarearen bitartez irudi horiek jaso eta irudikatuko ditugu exekuzio garaian. *Raspberry Pi* txartelaren errendimendua hobea izan dadin, *ROS*-en lan karga *Raspberry Pi* eta honekin sarearen bidez konektatuta egongo den ordenagailu baten artean banatuko da.

Memoria honek proiektuaren nondik norakoak azaltzen ditu, hainbat ataletan antolatuta. 2. kapituluan proiektuaren plangintza aurki daiteke, proiektuaren kudeaketa nola burutu den azalduz. Jarraian, 3. kapituluan, proiektua burutzeko erabili diren teknologia eta azpiegituren inguruan hitz egiten da. Ondoren, garapenaren prozesua eta egiaztapena nolakoak izan diren azaltzen da 4. kapituluan. Azkenik proiektuaren ondorioak irakur daitezke 5. kapituluan. Dokumentu honen eranskinetan, sortutako robotaren erabilpen gida eta bileren aktak biltzen dira. Bukatzeko, proiektua aurrera eramateko erabilitako bibliografia aurki daiteke.

2. KAPITULUA

Proiektuaren plangintza

Aurkibidea

Proiektuaren plangintzan, garatu den proiektua azaltzen da. Alde batetik proiektuaren deskribapen eta helburuak, lanaren deskonposaketa egitura eta lantzen diren atazak azaltzen dira. Bestalde proiektuaren kudeaketaren inguruko kronograma eta dedikazio denborak aurki daitezke. Azkenik,helburuak zehazten dituen kalitate plana eta hauek lortzeko lan metodologia, komunikazio plana eta arriskuen plana aurki daitezke.

2.1 Irismena

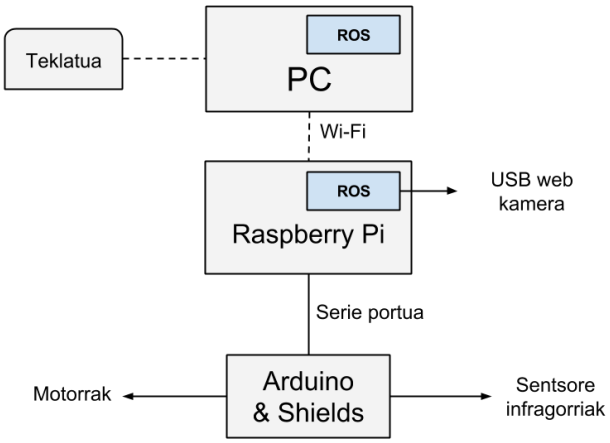
Jarraian, produktuaren eta proiektuaren irismena azalduko dira. Alde batetik, produktuan lortu beharreko ezaugarriak eta bestetik produktu hori arrakastaz lortzeko aurrera eramango den prozesua zehaztuz.

2.1.1 Produktua

Proiektu honen helburu nagusia *RSAIT* ikerketa taldearentzako auto itxurako robot bat burutzean datza. Hala ere, robot honek zenbait funtzionalitate betetzeaz gain, baldintza batzuk bete beharko ditu erabiliko duen hardwareari dagokionez. Izan ere, hardware libreko plataformez egongo da osatuta, zehazki *Arduino* mikrokontrolagailua eta *Raspberry Pi* ordenagailu txikiaren bitartez. Bestalde, robotari beharko duen adimena eman ahal izateko *ROS* ingurunea instalatuko da *Raspberry Pi*-an eta oinarri bezela *USB* kamera bat gehituko zaio honekin jolasteko.

Robotaren eraikuntza zerotik burutuko da. Behin materiala eskuratuta, bai motor, gurpil, sentsore infragorri eta egitura nagusia, hauek elkarrekin lotuko dira auto itxura emanez. Ondoren, *Arduino* mikrokontrolagailuaren bidez aipatutako periferikoak kontrolatuko dira. Gainera, beharrezko kasuetan, *Arduino* plaka *Raspberry Pi*-aren plakarekin konektatuko da beharrezko informazioa batetik bestera pasatzeko, robotari dagokion inteligentziaren zatirik handiena *Raspberry Pi*-an geratuz. Azkenik, esan bezala, *Raspberry Pi*-an *ROS* instalatuko da, inteligentzia artifiziala ezartzeko garaian erraztasun handiagoak ematen ditu eta.

Bestalde, robotaren funtzionalitateei dagokienez, hauek dira bete beharko dituen eginbeharrak: Autonomoki, marra beltz baten jarraipena; urrutiko agintearen bidezko autoaren gidatzea; eta web kamera baten irudiak jaso eta irudikatzea.



2.1 Irudia: Produktuaren eskema orokorra

Proiektuaren muga nagusia denbora denez, funtzionalitate bakoitzaren bikaintasun maila honen arabera egongo da baldintzatuta. Bestalde, denbora soberan izanez gero, funtzionalitate gehiago gehitzeko aukera egongo da, robotaren inteligentzia artifiziala handituz.

Kontuan izan beharreko ideia da, guzti honek oinarrizko maila bakarrik osatzen duela eta etorkizunera begira hobekuntza asko egin ahal izango direla eta funtzionalitate asko gehitu.

2.1.2 Proiektua

Proiektua arrakastatsua izan dadin, plagintzan idatzitakoa ahalik eta neurri handienean bete beharko da. Plangintzan azaldutako kalitate mailara iristea dagokion denboran, eta lehenago esan bezala, denbora soberan izanez gero, kalitate maila handiagoa lortzea da xedea.

Bestalde, proiektuaren helburu da ere plataforma berrien ezagutza: *Arduino*, *Raspberry Pi* eta *ROS*. Hauen inguruko informazioa bilatu, ezagutu eta beraien ahalmena zein den ikustea.

2.2 Lanaren deskonposaketa egitura

Proiektua arrakastaz burutzeko, bost atal nagusi bereizi dira. Alde batetik, kudeaketari dagokionez, bere barne daude proiektuaren plangintza, tutoreekin izango diren kudeaketa bilerak eta denbora desbiderapenak kontrolpean izateko jarraipen eta kontrola.

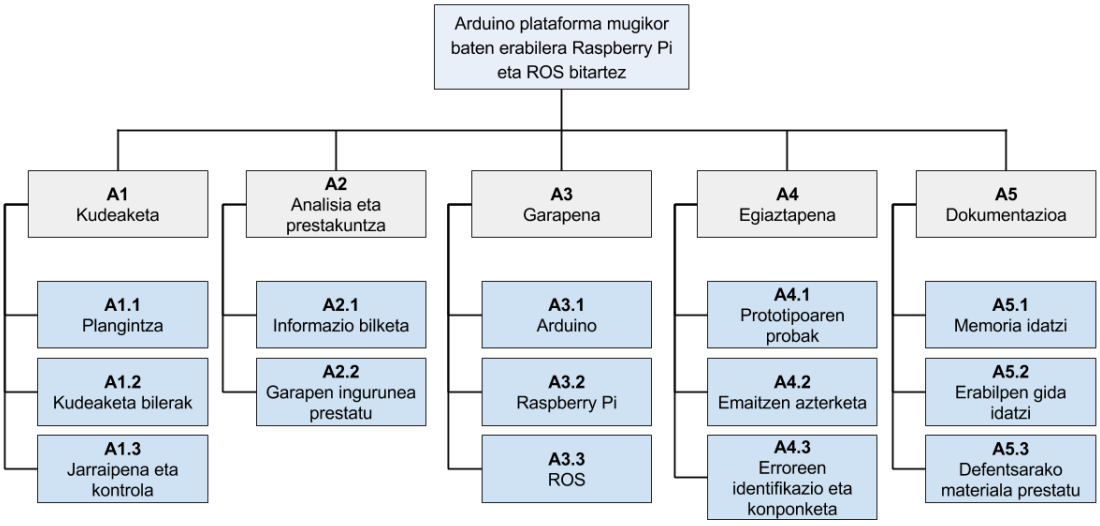
Bestalde, produktuaren garapenarekin hasi baino lehenago, beharrezkoa da erabiliko diren teknologien ezagutza. Horregatik, bigarren atala analisia eta prestakuntza dira. Hau da, beharko den informazioaren bilketa eta erabiliko diren teknologien garapen ingurunearen prestakuntza.

Proiektuaren mamia da hurrengo atala, garapena. Bertan hiru plataformak banatu dira, bakoitzak lan pakete bat osatzen duelarik: *Arduino*, *Raspberry Pi* eta *ROS*.

Arrakasta lortzeko, beharrezko da egindakoa ondo dagoela ziurtatu eta beharrezko konponketak burutzea. Horretarako, egiaztapena deituriko atala. Bertan, prototipoaren probak, emaitzen azterketa eta erroreen identifikazio eta konponketa egingo direlarik.

Azkenik, proiektua dokumentatzea ezinbestekoa denez, memoria, erabilpen gida eta defentsarako materiala prestatuko dira.

Guzti hau, 2.2 diagraman ikus daiteke.



2.2 Irudia: Lanaren deskonposaketa egitura (LDE diagrama)

2.3 Atazak eta ezaugarriak

Jarraian, proiektua garatzeko beharrezkoak izan diren atazak zerrendatu dira, aurreko ataleko lanaren deskonposaketa egiturako lan-paketeetatik abiatuz.

A1. Kudeaketa

A1.1 Plangintza

A1.1.1 Irismena zehaztu

A1.1.2 *LDE* diagrama burutu

A1.1.3 Atazak definitu

A1.1.4 Kronograma burutu

A1.1.5 Dedikazio estimazioa kalkulatu

A1.1.6 Kalitate plana burutu

A1.1.7 Arriskuen plana burutu

A1.1.8 Eskuraketen plana burutu

A1.1.9 Lan metodologia zehaztu

A1.2 Kudeaketa bilerak prestatu eta gauzatu

A1.2.1 Konstituzio bilera

A1.2.2 Lehenengo bilera

A1.2.3 Bigarren bilera

A1.2.4 Hirugarren bilera

A1.2.5 Laugarren bilera

A1.2.6 Bosgarren bilera

A1.2.7 Seigarren bilera

A1.2.8 Itxiera bilera

A1.3 Jarraipena eta Kontrola

A1.3.1 Desbideraketak kalkulatu

A1.3.2 Desbideraketen arrazoiak bilatu

A2. Analisia eta prestakuntza

A2.1 Informazio bilketa

A2.1.1 *Arduino (Meduino)*

A2.1.2 *Raspberry Pi*

A2.1.3 *ROS*

A2.2 Garapen ingurunea prestatu

A2.2.1 *Meduino* liburutegiaren funtzionamendua ulertu

A2.2.2 *Raspbian*-en instalazioa

A2.2.3 *ROS*-en instalazioa

A2.2.4 *ROS*-en funtzionamendua ulertu tutorialak jarraituz

A3. Garapena

A3.1 *Arduino*

A3.1.1 Marra jarraitzailea

A3.1.2 Kontrol telegidatua

A3.1.3 Programen integrazioa

A3.2 *Raspberry Pi*

A3.2.1 *Arduino*-rekin komunikazioa

A3.2.2 Programen integrazioa

A3.3 *ROS*

A3.3.1 *USB* web kamera

A4. Egiaztapena

A4.1 Prototipoaren probak

A4.1.1 Marra jarraitzailea

A4.1.2 Auto gidatua

A4.1.3 *USB* web kamera

A4.2 Emaitzaren azterketa

A4.3 Erroreen identifikazio eta konponketa

A5. Dokumentazioa

A5.1 Memoria idatzi

A5.2 Erabilpen gida prestatu

A5.3 Defentsarako materiala prestatu

2.4 Emangarrien identifikazio eta ezaugarriak

Proiektu hau burutzeko orduan entregatu beharreko emangarriak ondorengoak dira:

1. **Produktua**
   Alde batetik robota bera, dagozkion periferiko guztiekin modu egokian kokatuta. Bestetik, bere barnekaldean funtzionamendu egoki baterako sortu eta garatuak izan diren programa guztiak. Robota defentsaren ostean entregatuko bada ere, prest egon behar du irailaren 2an.

2. **Memoria**
   Proiektuaren kudeaketa eta garapena nola burutu diren azaltzen duen dokumentua da. Dokumentu hau idazteko, *Donostiako Informatika Fakultateak* eskeintzen dituen txantiloietaz baliatu da eta bertan jasota dauden bete beharreko ezaugarriak jarraitu dira. Dokumentu hau *ADDI* plataformara igo behar da irailaren 2a baino lehenago.

3. **Gidaliburua**
   Memorian instalazio eta garapenak azalduko diren arren, komenigarria da robotaren gidaliburu labur bat burutzea, etorkizunean beste pertsona batek robota hartzean nola erabili behar den jakin dezan. Izan ere, oinarri den robota izanik badu etorkizunera begira hobekuntza lana. Gidaliburua produktuarekin batera entregatuko da, defentsaren ostean.

4. **Defentsarako materiala**
   Proiektuaren azken fasea da defentsa. Bertan egindakoa modu txukun eta argian azaltzeko beharrezko materiala sortu beharko da. Defentsa Irailaren 16 eta 18 bitartean burutuko da. Beraz, beharrezko izango da egun batzuk lehenago bukatuta egotea.

2.5 Denboraren plangintza

2.5.1 Kronograma

Hau da hasiera batean planteatu den denbora kronograma.

	Martxoa					Apirila				Maiatza				Ekaina					Uztaila				Abuztua					Iraila	
	2	9	16	23	30	6	13	20	27	4	11	18	25	1	8	15	22	29	6	13	20	27	3	10	17	24	31	7	14
A1.1. Plangintza															-														
A1.2. Kudeaketa bilerak															-							-	-	-	-				
A1.2. Jarraipena eta kontrola															-														
A2.1. Informazio bilketa															-														
A2.2. Garapen ingurunea prestatu															-														
A3.1. Arduino															-														
A3.2. Raspberry Pi															-														
A3.3. ROS															-														
A4.1. Prototipoaren probak															-														
A4.2. Emaitzaren azterketa															-														
A4.3. Erroreen identifikazio eta konponketa															-														
A5.1. Memoria idatzi															-														
A5.2. Erabilpen gida prestatu															-														
A5.3. Defentsarako materiala prestatu															-														

2.3 Irudia: Kronograma

Lehenik eta behin, kudeaketa bilera bat egon da tutoreekin proiektuaren planteamendua azaltzeko. Bilera hauek, hiru astean behin izatea aurreikusten da, batez ere proiektua bide onetik doan jakiteko eta momentuko zalantzak argitzeko. Helburu berdinarekin ere, proiektuaren jarraipen eta kontrola egin beharko da une oro.

Ezer egiten hasi aurretik, lehenik eta behin, informazioa bildu da. Izan ere, erabili beharreko teknologiak berriak izanik, beharrezkoa izan da hauen inguruko ikerketa txiki bat plangintza egoki burutu ahal izateko. Behin informazio hori jasota, plangintza burutzeari ekin zaio.

Garapena hiru zatitan banatuko dela aurreikusten da eta zati bakoitzak hiru pauso beteko ditu: Informazio bilketa eta garapen ingurunearen prestakuntza burutuko dira lehenik, ondoren garapena egingo da eta azkenik probak, emaitzen azterketa eta erroreen identifikazio eta konponketa. Prozesu bera errepikatuko da *Arduino*, *Raspberry Pi* eta *ROS*-en kasuetan, bakoitzarekin ordu kopuru ezberdina aurreikusten delarik.

Azkenik memoria idatziko da, erabilpen gida garatu eta defentsarako materiala prestatu.

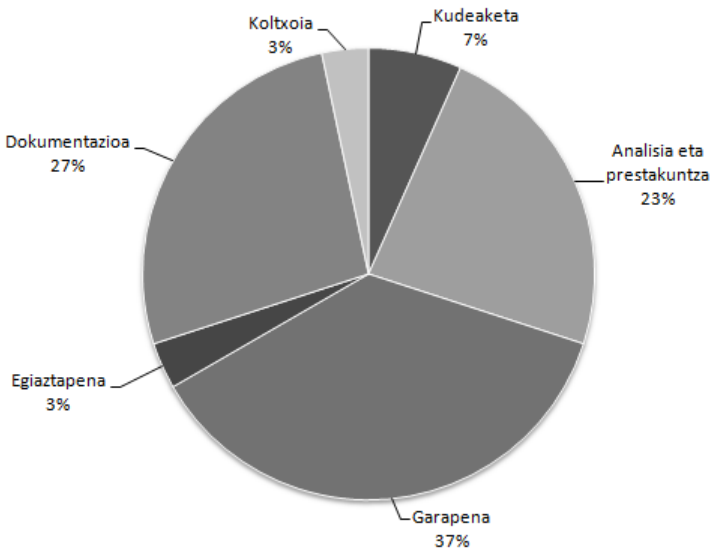
2.5.2 Dedikazio estimazioa

2.1 taulan proiektua amaitzeko egindako lanaren dedikazio estimazioa ikus daiteke, lan-paketeka antolatuta. Informazio hori laburbiltzeko, sektore-diagrama bat ere sortu da (Ikus 2.4 irudia) lan-pakete bakoitzari dedikatzea aurreikusitako pisua argi ikusteko.

Argi ikus daiteke bai taulan bai sektore diagraman, proiektuaren pisurik handiena garapenak eta hau burutzeko beharrezko den analisi eta prestakuntzak hartzen dutela.

Lan-paketea	Dedikazio estimazioa (ordutan)
A1. Kudeaketa	**20**
A1.1. Plangintza	15
A1.2. Kudeaketa bilerak	3
A1.3. Jarraipena eta Kontrola	2
A2. Analisia eta prestakuntza	**70**
A2.1. Informazio bilketa	40
A2.2. Garapen ingurunea prestatu	30
A3. Garapena	**110**
A3.1. Arduino	20
A3.2. Raspberry Pi	40
A3.2. ROS	50
A4. Egiaztapena	**10**
A4.1. Prototipoaren probak	3
A4.2. Emaitzaren azterketa	3
A4.2. Erroreen identifikazio eta konponketa	4
A5. Dokumentazioa	**80**
A5.1. Memoria	65
A5.2. Erabilpen gida	5
A5.3. Defentsarako materiala	10
Koltxoia	**10**
GUZTIRA	**300**

2.1 Taula: Dedikazioaren estimazioa



2.4 Irudia: Dedikazio estimazioaren sektore-diagrama

Pisu nahiko du baita ere dokumentazioaren zatiak, azalpen guztiak biltzen ditu eta. Azkenik, kudeaketak eta egiaztapenak hartzen dute zatirik txikiena.

Sektore diagraman ikus daiteke badagoela zati bat koltxoi moduan definitu dena. Izan ere, denbora hori aurreikusi ez diren ezustekoak konpontzeko erabili ahal izango da.

2.5.3 Dedikazio erreala

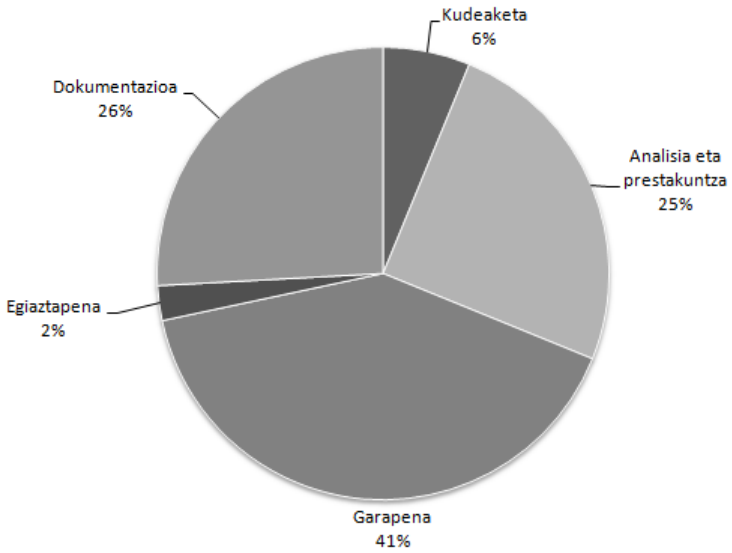
2.2 taulan proiektua amaitzeko egindako lanaren dedikazio erreala ikus daiteke, lan-paketeka antolatuta.

Lan-paketea	Dedikazio erreala (ordutan)
A1. Kudeaketa	**20**
A1.1. Plangintza	15
A1.2. Kudeaketa bilerak	3
A1.3. Jarraipena eta Kontrola	2
A2. Analisia eta prestakuntza	**80**
A2.1. Informazio bilketa	40
A2.2. Garapen ingurunea prestatu	40
A3. Garapena	**130**
A3.1. Arduino	20
A3.2. Raspberry Pi	40
A3.2. ROS	60
A4. Egiaztapena	**8**
A4.1. Prototipoaren probak	2
A4.2. Emaitzaren azterketa	2
A4.2. Erroreen identifikazio eta konponketa	4
A5. Dokumentazioa	**83**
A5.1. Memoria	70
A5.2. Erabilpen gida	3
A5.3. Defentsarako materiala	10
GUZTIRA	321

2.2 Taula: Dedikazio erreala

Taula laburbilduz, sektore-diagrama bat ere sortu da (ikus 2.5 irudia) lan-pakete bakoitzari dedikatutako pisua argi ikus dadin.

Aurreikusi bezala, proiektuaren pisurik handiena garapenak eta lehenago egin den analisi eta prestakuntzak hartzen dute. Dokumentazioak, baita ere, pisu handia hartzen du. Azkenik, proiektuaren kudeaketa eta produktuaren egiaztapena burutzen erabilitako ordu kopurua txikienak direla ikus daiteke.



2.5 Irudia: Dedikazio errealaren sektore-diagrama

2.5.4 Ondorioak

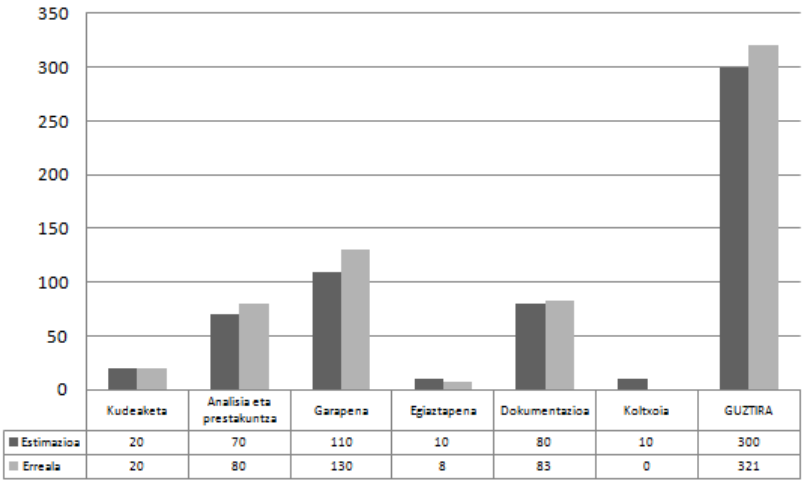
2.6 irudian ikus daiteke estimatutako orduen eta ordu errealen arteko aldea. Orokorrean, proportzioak mantendu diren arren, ordu kopuru handiagoa sartu behar izan da atal bakoitzean, egiaztapenean eta kudeaketan izan ezik.

Gainera, estimazioan utzitako 10 orduko koltxoia erabili behar izan da eta baita beste 20 ordu inguru ere. Horrela, proiektua burutzen emandako ordu kopurua 321 ordukoa izan da 300 orduko estimazioa eginda zegoenean.

2.6 Kalitate plana

Atal honetan, proiektuaren kalitate plana azalduko da. Proiektuaren kalitate maila finkatzeko orduan, lehenik eta behin, honek bete beharreko oinarrizko maila zein den zehaztu da. Hau da, proiektua amaitutzat emateko bete beharreko maila minimoa. Behin oinarrizko maila hori igarota, denboraren arabera, kalitate maila handiago bat lortzeko aukera egongo da.

Oinarrizko maila lortzeko irizpideak jarraian azalduko dira. Esan beharra dago, irizpide horiek irismenean aipatutako ezaugarriekin bat datozela orokorrean.



2.6 Irudia: Dedikazioen diagrama

1. **Autonomoki, marra beltz baten jarraipena.**
   Robotak bere kabuz, marra beltz baten jarraipena egin beharko du aurrekaldean dituen bi sentsore infragorriei esker. Marra beltza jarraitzeko gai izan beharko da ahalik eta denbora luzeenean, azken helburua marra beltzak dirauen bitartean zirkuitua jarraituz aurrera joatea izanik.

2. **Urrutiko agintearen bidezko autoaren gidatzea.**
   Autonomoki gidatzeaz gain, autoa norberak gidatzeko aukera eman behar da. Horretarako teklatu moduko urrutiko aginte bat erabiliko da, *Raspberry Pi*-ari informazioa bidaliko diona, eta honek *Arduino* txartelari bidaliko dizkio motorrei bidali beharreko aginduak.

3. **Web kamera baten irudiak transmititzea.**
   Aldi berean, *USB* konexio bidezko web kamera baten irudiak jaso beharko dira *ROS* plataforma erabiliz. Gainera hau *Raspberry Pi*-an egon beharko da instalatuta eta irudiak ikusi egin beharko dira.

Oinarrizko ezaugarri guzti hauek, Uztailaren 24rako bukatuta egon beharko dira. Hemendik aurrera, edo lehenago bukatuz gero eta denbora soberan izanez gero, hobekuntzei utziko zaie tartea. Hala ere, hobekuntza hauek ez dira planifikatu, izan ere proiektuak aurrera egin ahala hobeto antzemango direla uste da.

2.7 Lan metodologia

Proiektua aurrera eramateko erabilitako metodologia jarraian azaltzen dena izan da. Horretarako, komeni da proiektuaren bizitza zikloa fase ezberdinetan banatzea.

Lehenik eta behin, ezer egin baino lehenago, informazio bilketa sakon bat burutu da. Izan ere, erabili beharreko teknologia guztiak berriak izanik, ez da posible plangintza erreal bat egitea erabilitako teknologiaren ezaugarriak ezagutu gabe.

Behin nondik norakoak gainetik ezagututa, plangintza bat burutu da, non lortu beharreko helburuak eta hauek lortzeko jarraitu beharreko pausoak azaltzen diren.

Ondoren, garapenari ekingo zaio. Garapen honek, noski, informazio bilketa sakonagoak eskatuko ditu, tutoreekin bilerak arazoak edo zalantzak dauden bakoitzean eta lanaren jarraipen sakon bat egitea bidea egokia den ala ez ikusteko. Horretarako, egunero egindako lanaren ostean, eguneroko bat idatziko da. Egun horretan landutakoa azalduz, erabilitako bibliografia idatziz eta igarotako orduak ezarriz.

Bi astero, plagintzan ezarritako orduak eta ordu errealak alderatuko dira desbiderapenak dauden ikusi eta hauek kudeatzeko.

Bestalde, noizean behin, proiektuaren segurtasun kopia bat egingo da *Google Drive* plataforman.

Behin proiektuaren oinarrizko maila lortuta, memoria idazteari ekingo zaio. Memoria idatzita dagoenean, denbora soberan egongo balitz, hobekuntzak burutuko dira.

Azkenik, erabilpen gida eta defentsarako materiala prestatuko dira.

2.8 Komunikazio plana

Edozein arazo edo galdera burutzeko, proiektuaren tutoreekin harremanetan jarri beharko da eta ahal izanez gero bilera data bat finkatu. Horretarako, *UPV/EHU*-ko posta elektronikoaz baliatuko da.

Bilera guztietan hitzegin eta adostutakoa, bilera horien aktetan jasoko da. Akta horiek, memoriaren bukaeran eranskin moduan agertuz.

2.9 Arriskuen plana

Proiektua behar bezala aurrera eraman eta bere arrakasta ziurtatzeko, komeni da izan daitezkeen arrisku nagusi eta kritikoenak identifikatzea, hauei aurre egin ahal izateko. Jarraian, arrisku hauek bi multzotan banatuko ditugu. Alde batetik produktuaren inguruan identifikatutako arriskuak ditugu, produktuak behar bezala lan egin dezan kontuan izan behar direnak. Bestalde, proiektuaren arriskuak ditugu, hau da, produktua, bete beharreko epean behar bezala garatzeko sor daitezkeen arriskuak.

2.9.1 Produktua

1. **Bateriak bukatzea**
   Bateriak amaitzearen eraginez, robotaren motorrek futzionatzeari utziko baliote, robotaren mugimendua desagertuko litzateke. Arazo honi aurre egiteko, ziurtatu beharko da pilak beti kargatuta daudela eta egin beharreko konexio guztiak behar bezala daudela. Beti ere, motorren demoa exekutatu daiteke beraien funtzionamendu zuzena egiaztatzeko.
   Probabilitatea: Txikia
   Eragina: Larria

2. **Infragorriek ez funzionatzea.**
   Infragorriek funtzionatzeari uzten badiote, eragina soilik marra jarraitzailean izango luke. Arazo honi aurre egiteko, ziurtatu beharko da konexio guztiak zuzen egin direla eta infragorri bakoitzaren goikaldean dauden led-ak behar bezala pizten direla.
   Probabilitatea: Txikia
   Eragina: Ez da larria

3. **Kamerak ez funzionatzea.**
   Kamerak funtzionatuko ez balu, irudiak jasotzearen funtzioa bakarrik galduko litzateke. Arazo honi aurre egiteko, ziurtatu beharko da ordenagailuak gailua ondo detektatzen duela.
   Probabilitatea: Txikia
   Eragina: Ez da larria

4. ***Raspberry Pi*-aren bateria bukatzea.**
   Raspberry Pi-aren bateria bukatuz gero, robotaren inteligentzia guztia desagertuko

litzateke, eta bere gaitasun bakarra *Arduino* plakaren gain dauden funtzionalitateak aurrera eramatea izango da. Honi aurre egiteko, bateria behar bezala kargatuko da, eta bat ez ezik bi bateria edukiko dira prest une oro.
Probabilitatea: Ertaina
Eragina: Oso larria

5. **Wifiak ez funzionatzea.**
   Wi-Fi-ak funtzionatzeari utziko balio, *Raspberry Pi*-aren kontrola ezingo litzateke sarean dagoen beste ordenagailu batetik kontrolatu, ondorioz *HDMI* kable bat jarri beharko litzaioke bertan pasatzen dena ikusteko eta mugikortasuna galduko luke. Honi aurre egiteko, IP helbide estatiko bat posible zaio esleitu gailuari konexioa nora egin jakin ahal izateko, eta bien arteko komunikazioa egiaztatu beharko dugu lehenik eta behin *"ping"* komandoaren bidez. Lortuko ez balitz, *HDMI* kable bat prest izatea komeni da, sare txartelaren konfigurazioa begiratu ahal izateko eta nahi den moduan aldatzeko.
   Probabilitatea: Ertaina
   Eragina: Ertaina

2.9.2 Proiektua

1. **Egindako lana galtzea.**
   Egindako lana, bai kode eta bai memoria galduz gero, izugarrizko akatsa izango litzateke. Honi aurre egiteko, memoriaren atal guztiak *Google Driven* daude gordeta. Bestalde, garatzen den kodea noizean behin plataforma horretara ere igotzen da.
   Probabilitatea: Oso txikia
   Eragina: Oso larria

2. **Denbora falta.**
   Erabili beharreko teknologia berria denez, gerta liteke irismenean zehaztutako guztia egiteko denbora ez izatea. Honi aurre egiteko, lehenik informazio apur bat bildu da plangintza burutu aurretik. Oinarrizko mailara iritsitakoan memoria burutuko da, proiektua bukatutzat eman daitekeelarik. Denbora izanez gero, orduan egingo dira hobekuntzak.
   Probabilitatea: Txikia
   Eragina: Ez da larria

3. **Elkartzeko bateraezintasuna.**
   Uda tartean izanda, kontuan izan behar da tutoreen ordutegi aldaketa. Honi aurre egiteko, bilera beharrak daudenean, ahalik eta azkarren jarri beharko da harremanetan beraiekin. Beraien egutegia kontuan izan beharko da plangintza burutzerakoan.
   Probabilitatea: Ertaina
   Eragina: Ez da larria

3. KAPITULUA

Erabilitako teknologia eta azpiegitura

Aurkibidea

3.1 Starter Robot Kit

Robotaren egituraren zatirik handiena osatzen du *Starter Robot Kit*-ak. Izan ere, *Makeblock* [4] etxeko kit honek dakartza autoaren txasisa osatzeko osagaiak, gurpilak, motorrak, sentsore infragorriak, eta guzti hori kontrolatuko duen *Arduino* txartel baten bertsio propio bat shield eta liburutegi batez lagundurik.

Jarraian, aipatutako gailu guzti horiek zerrendatu eta azalduko dira banan banan:

1. **Txasisa**
   Autoaren hezurdura osatzen duten osagaiak dira hauek. Metalezko piezak eta hauek lotzeko torlojuak dakartza kit honek "H" itxurako egitura bat osatuz.

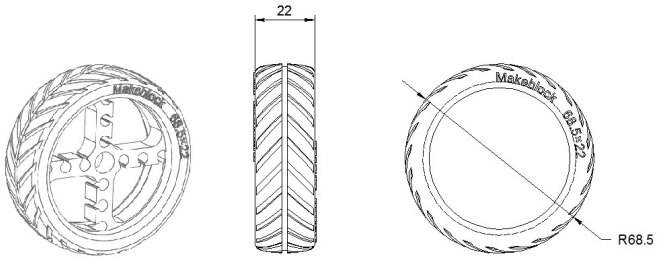
2. **Gurpilak**
   Guztira, 4 gurpil dakartza kit honek. Alde batetik, autoaren atzekaldean motorrei lotuta joango diren bi gurpilak. Bestalde, autoaren aurrekaldean inongo konexiorik gabe, atzeko gurpilen mugimenduaren eraginez mugituko diren bi gurpil txiki.

   Aurreko gurpilak, 25 mm-ko diametroa duten plastikozko gurpiltxoak dira (Ikus 3.1 irudia).

3.1 Irudia: Autoaren aurreko gurpilak

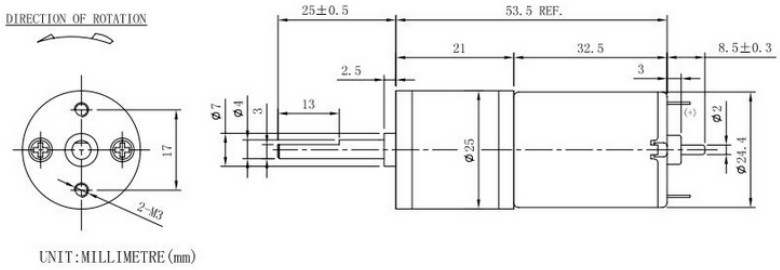
   Atzeko gurpilak, berriz, 68,5 mm-ko diametroa duten gurpil handiagoak dira (Ikus 3.2 irudia).

3. **Motorrak**
   Atzeko gurpil bakoitzari loturik, motor bana joango da konektatuta. Motor hauek



3.2 Irudia: Autoaren atzeko gurpilak

25 mm-ko diametroa duten korronte zuzeneko motorrak dira (Ikus 3.3 irudia).



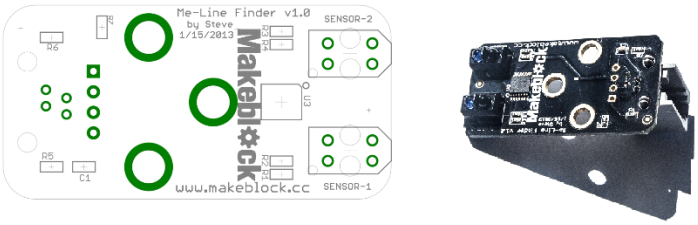
3.3 Irudia: Motorren egitura

Motorraren eredua: LT25GA34-370T
Tentsioa: 6.0 V(DC)

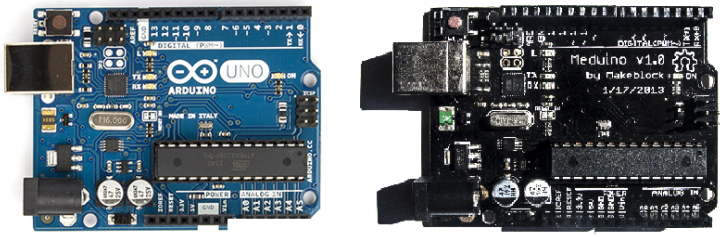
4. **Me-Line Finder V1.0**
   Autoaren aurrekaldean ezarriko den bi sentsore infragorriz osatutako txarteltxoa da. Sentsore hauek marra beltza detektatzeko erabiliko dira (Ikus 3.4 irudia).

5. **Meduino V1.0**
   Meduino txartela *Makeblock* etxeak sortutako *Arduino* txartelaren bertsio propio bat da. Bereziki, proiektu honetan erabilitako *Meduino* txartela, *Meduino V1.0* txartela da eta *Arduino UNO R3* txartelarekin guztiz bateragarria da (Ikus 3.5 irudia).

   Arduino, kode irekiko plataforma bat da, mikrokontrolagailudun plaka eta garapen ingurune batean oinarritzen dena [5].



3.4 Irudia: Sentsore infragorriak



3.5 Irudia: Arduino UNO eta Meduino V1.0 txartelak

Hardwarea, *Atmel AVR* motako mikrokontrolagailu batek eta sarrera/irteerako portuek osatzen dute. Mikrokontrolagailu erabilienak *ATmega168*, *ATmega328* eta *ATmega8* dira, beraien sinpletasun eta koste baxuarengatik.

Bestalde, softwareari dagokionez, *Processing*-en oinarritzen den *Arduino* garapen ingurune bat erabiltzen du eta *Wiring*-en oinarritutako *Arduino* programazio lengoaian programa daiteke. Kode irekiko *Arduino*-ren *IDE*-a web orrialde ofizialetik deskarga daiteke.

Arduino-k inguruko informazioa jaso dezake plakan dituen sarrera/irteerako pin analogiko eta digitalen bitartez. Horrela, hainbat periferiko kontrolatuz: led argiak, motorrak, txirrinak, etab. Behin plaka programaturik dagoenean, hau da sortutako programa *Arduino*-an kargatzen denean, ez du ordenagailura konektatuta egon beharrik.

Hauek dira *Meduino V1.0* txartelaren ezaugarriak:

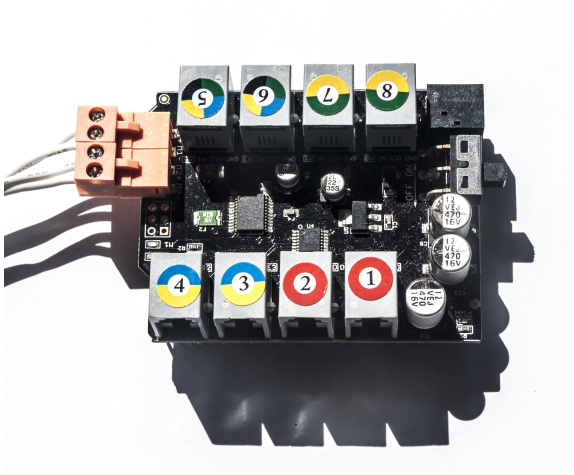
- *ATmega328* mikrokontrolagailua
- 7-12 V-eko tentsioa
- Sarrera/Irteerako 14 pin digital (Hauetatik 6 *PWM*)

- Sarrerako 6 pin analogiko
- 32 KB-eko flash memoria
- 2 KB-eko *SRAM* memoria
- 1 KB-eko *EEPROM* memoria
- 16 MHz-eko erloju abiadura

6. **Me-Base Shield V1.0**
   Me-Base Shield-a *Arduino* motako shield bat da, *Meduino* ala *Arduino* txartelaren pin gehienak *RJ25* motako zortzi konektoreetara konektatzen dituena (Ikus 3.6 irudia). Gainera, bi motor kontrolatu ahal izateko zirkuitu integratu bat dakar. Bien arteko konexioa burutzeko, nahikoa izango da *Me-Base Shield*-a lehenago aipatutako *Meduino* txartelaren gainean jartzearekin.

   Hauek dira *Me-Base Shield V1.0* txartelaren ezaugarriak:

   - 6-12 V-eko tentsioa
   - *RJ25* motako 8 konektore beheko txartelaren pinekin konexioa burutzeko
   - Bi motor kontrolatzeko zirkuitu integratua
   - Konexio intuitiboa ahalbidetzen duen kolore eta zenbaki bidezko markaketa
   - *Makeblock* liburutegia erabiltzeko aukera



3.6 Irudia: Me-Base Shield V1.0 txartela

7. **Arduino IDE**
   Aipatu bezala, *Arduino* txartelaren programazio errazerako existitzen da jada etxe

bereko softwarea. Instalazio erraza du eta guztiz askea da bere deskarga. *Arduino*-ren orrialde ofizialean aurki daiteke hau, jarraian azaltzen den estekan alegia: `https://www.arduino.cc/en/Main/Software`

Deskarga burutu bezain laister, exekutagarria ireki eta instalazioa hasiko da. Instalatuta, prest egongo da *Arduino* txartelarekin lan egiteko.

3.7 Irudia: Arduino IDE

8. **Makeblock liburutegia**
   Liburutegi honi esker, erabiliko diren periferikoak oso modu errazean programatuko dira. Esaterako, liburutegi honetan sentsore infragorrien informazioa jasotzen duen programa bat dago garatuta, zeinean esaten den sentsore bakoitza marra beltzaren barruan dagoen ala ez.

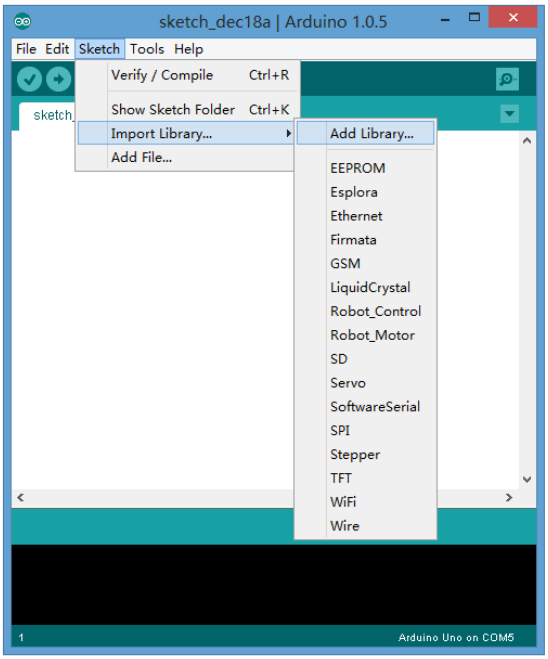
   Liburutegi hau erabiltzeko, web orri ofizialetik jeitsi eta instalatu egin beharko da. Instalazioa burutzeko, jarraitu beharreko pausoak ondorengoak dira.

   - Deskarga burutu: `https://github.com/Makeblock-official/Makeblock-Library/archive/master.zip`
   - Deskargatutako *ZIP* fitxategia erauzi
   - *Arduino*-ren *IDE*-a ireki eta nabigazio barran hurrengoa sakatu: *Sketch* > *Import Library* > *Add Library*. Bertan deskargatu eta erauzitako fitxategia gehitu (Ikus 3.8 irudia).

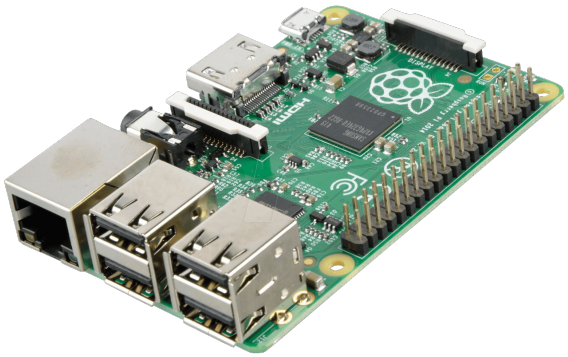
3.2 Raspberry Pi

3.2.1 Informazio orokorra

Raspberry Pi-a, *Raspberry Pi Foundation* etxeak sortutako zirkuitu-plaka bakarreko ordenagailu oso txiki bat da [6], irakaskuntzan konputazio zientziak sustatzeko asmoarekin sortua (Ikus 3.9 irudia). Elektronika proiektuetan erabiltzeko eta ordenagailu arrunt batek egin ditzakeen funtzio gehienak betetzeko prest dago: kalkulu-orriak, testu prozesua, bideoen erreprodukzioa, etab.



3.8 Irudia: Liburutegia gehitu



3.9 Irudia: Raspberry Pi B+ txartela

Gailu honen lehenengo belaunaldia, *Broadcom* etxeko *BCM2835* txip bakarreko sisteman *(System on a chip)* oinarritzen da. *ARM1176JZF-S* 700 MHz-eko prozesadorea dauka, *VideoCore IV GPU*-a, eta 256 eta 512 MB-eko *RAM* memoria ereduaren arabera. *Raspberry Pi*-ak ez dauka disko gogorrik, izan ere, sistema abiarazteko eta memoria iraunkor moduan *SD* edo *MicroSD* txartela erabiltzen du. Elikadurari dagokionez, nahikoa da gaur egun mugikor gehienek erabiltzen duten 5 V-eko tentsioko *MicroUSB* kargagailu bat konekatzearekin.

Bestalde, sistema eragileari dagokionez, *Raspberry Pi Foundation*-ek, *ARM* prozesadoreentzat egokitutako *Linux* banaketa ezberdinak jarri ditu eskuragarri: *Raspbian*, *RISC OS 5*, *Arch Linux ARM*, etab. Programazioari dagokionez, hainbat dira jasan ditzaketen programazio-lengoaiak: *Python*, *C*, *C++*, *Java*, *Perl*, *Scratch* eta *Ruby* esaterako.

Raspberry Pi txartelaren lehenengo belaunaldiari dagokionez, 4 eredu ezberdin daude: A (23€ inguru), A+ (18,5€ inguru), B (32,5€ inguru) eta B+ (32,5€ inguru). Itxura aldetik, A, B eta B+ txartelek 85.5 x 56.5 mm-ko tamaina dute eta 45 g pisatzen dute, bestalde A+ txartelak 65 x 56.5 mm-ko tamaina du eta 23 g besterik ez ditu pisatzen. Hardwareari dagokionez, lau txartelek *Broadcom* etxeko *BCM2835* txipa daukate integratuta eta 700 MHz-eko *ARM1176JZF-S* prozesadorea erabiltzen dute. Irudien prozesamendurako *Broadcom* etxeko *VideoCore IV* multimedia prozesadorea dute eta honez gain A eta A+ txartelek 256 MB eta B eta B+ txartelek 512 MB-eko *SDRAM*-a.

A eta A+ txartelek *USB 2.0* portu bakarra dute, B txartelak 2 eta B+ txartelak 4. Txartel guztiek bideo sarrera eta irteera berdina dute. Sarrerari dagokionez 15 pineko *MIPI CSI* kamera interfazea, *Raspberry Pi camera* edo *Raspberry Pi NoIR camera*-rekin erabiltzeko. Bideoaren irteerarako berriz, *HDMI*, *RCA* konektorea eta *DSI* interfazea *LCD*-rentzat. Audio irteera ere berdina da txartel guztietan, analogikoa 3.5 mm-ko *TRRS* konektore bitartez eta digitala *HDMI* bitartez.

Barne biltegirako, lehenago aipatu bezala, txartel bat sartzea beharrezkoa izango da. Txartel hori *SD*, *MMC* edo *SDIO* motakoa izan daiteke A eta B ereduetan. A+ eta B+ ereduetan berriz, *MicroSD* motakoa izan beharko da.

Sare konexioa bakarrik B eta B+ ereduek dute, 10/100 Mbit/s-ko Ethernet konexioa (*RJ-45* konektorea).

Azkenik, behe mailako periferikoak daude eta hauek ezberdinak dira txartel bakoitzean. A ereduak 8 *GPIO (General Purpose Input/Output)*, gehi *GPIO* bezala erabil daitezkeen *UART*, *I2C* busa, *SPI* busa chip select birekin, *I2S* audio, +3.3 V, +5 V eta lurra ditu. A+ eta B+ ereduek 17 *GPIO*, gehi funtzio espezifiko berdinak, eta *HAT ID* busa. Az-

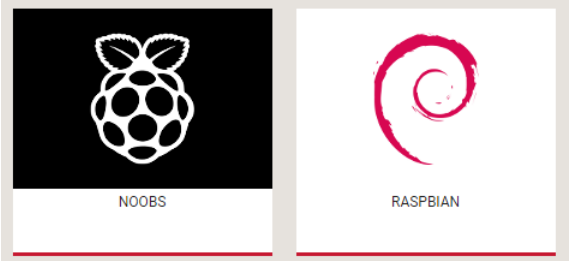
kenik, B ereduak 8 *GPIO*, gehi *GPIO* bezala erabil daitezkeen *UART*, *I2C* busa, *SPI* busa chip select birekin, *I2S* audio, +3.3 V, +5 V eta lurra, gehi beste 4 *GPIO* erabiltzaileak soldadurak eginez gero.

Proiektu honetan erabiliko den txartela, *Raspberry Pi B+* ereduko txartela izango da. Softwareari dagokionez, bertan *Raspberry Pi* fundazioak txartelarekin lanean hasteko gomendatzen duen *Raspbian* sistema eragilea instalatuko da. *Raspbian*, *Debian*-en oinarritutako sistema eragilea da, kode irekikoa eta gailu honen hardwarearentzat bereziki sortua izan zena 2012an.

3.2.2 Instalazioa

Lehenik eta behin, *Raspberry Pi*-an instalatuko den sistema eragilea deskargatu beharko da. Proiektu honen kasuan, *Raspbian* izango da sistema eragilea eta esteka honetan deskarga daiteke: `https://www.raspberrypi.org/downloads/`.

Bertan, bi aukera daude. Alde batetik, *NOOBS* izenekoak, sistema eragileak saretik instalatzen ditu. Bestalde, *Raspbian* izenekoa, sistema eragilea bera da, ondorioz ez du sare konexiorik behar.



3.10 Irudia: NOOBS eta Raspbian

Behin deskarga burutu dela, ondoren *Raspberry Pi*-an sartuko dugun *MicroSD* txartelean erauziko dugu dena. *HDMI* kablearen bidez, pantaila batera konektatu eta korrontea jarri bezain laister martxan jarriko da txartela. Beharrezko izango da teklatu eta xagu bat konektatzea honen kontrolerako. Nahi izanez gero, ethernet bidezko konexioa ezarriko diogu.

Piztu eta berehala, *Raspbian* aukeratu eta instalazioa abiatuko da. Behin instalatu denean, berarekin lan egiten hasi ahal izango da, saioa hasteko lehenetsitako kredentzialak, izena *"pi"* eta pasahitza *"raspberry"* izanik.

Erosotasuna lortze aldera, lehenik eta behin, interneteko konexioaren konfigurazioa burutuko da. Horrela, hurrengo konexioetan ez da beharrezko izango ez *HDMI* kablerik, ez pantailarik ezta teklatu eta xagurik, *Raspberry Pi*-aren gaineko kontrola norberaren ordenagailutik egin ahal izango da eta. Alde batetik, ethernet bidezko konexioa burutu daiteke, beste aukera bat, *USB* bidezko *Wi-Fi* txartel bat gehitzea da. Azken hau beharrezkoa izango da robotari aurrerago mugimendua eman nahi bazaio.

Wi-Fi-aren konfigurazioa burutzeko, jarraian dagoen agindua idatzi beharko da:

```
# sudo nano /etc/network/interfaces
```

Agertuko den fitxategia, sare interfazearen konfigurazioa izango da. Bertan honakoa idatzi beharko da:

```
auto lo

iface lo inet loopback
iface eth0 inet dhcp

allow-hotplug wlan0
auto wlan0

iface wlan0 inet dhcp
      wpa-ssid "Wi-Fi-aren izena"
      wpa-psk "Wi-Fi-aren pasahitza"
```

Azkenik, txartela berrabiarazi beharko da konfigurazioak eragina izan dezan. Horretarako:

```
# sudo shutdown -r now
```

Piztu bezain laister, *"ifconfig"* komandoari esker ea konexioa ondo burutu den eta sare txartelak zein IP helbide duen jakin ahal izango da, bai ethernet bidezko eta bai *Wi-Fi* bidezko konexioetan. Helbide hori beharrezkoa izango da beste ordenagailu batetik konexioa burutzeko, hortaz komenigarria da estatikoa izatea eta beti berdin mantentzea. Helbide finko bat esleitzeko, pauso hauek jarraitu behar dira.

Komando lerroan honakoa idatzi beharko da:

```
# sudo nano /etc/network/interfaces
```

Ethernet bidezko konexioa baldin bada erabili nahi dena, *"eth0"* interfazean egin beharko dira aldaketak. *Wi-Fi* bidezko konexioa bada, *"wlan0"* atalean egin beharko dira. Hauek dira aldatu eta gehitu beharreko lerroak, norberak jarri beharreko helbideak jartzen dituelarik:

```
iface wlan0 inet static
address 192.168.1.141
netmask 255.255.255.0
gateway 192.168.1.1
```

Konfigurazioa eginda, ordenagailutik *Raspberry Pi*-a atzigarri dagoela ziurtatu beharko da. Horretarako *"ping"* komandoaz baliatu daiteke. Konexioak funzionatzen duela ikusita, *Raspberry Pi*-ra konektatzeko aukera egongo da agindu honen bidez, IP helbidea norberaren *Raspberry Pi*-arena izanik:

```
# ssh -X pi@192.168.1.141
```

Dagoeneko prest egongo da txartela berarekin lan egin ahal izateko.

3.3 ROS

3.3.1 Informazio orokorra

ROS (Robot Operating System), *Ubuntu*-n oinarritzen den roboten software garapenerako sortutako framework bat da [7]. Sistema eragile estandar baten funtzioak betetzen ditu, hardware abstrakzioa, maila baxuko gailuen kontrola eta prozesuen arteko mezu trukea besteak beste. Honez gain, konputagailu anitzen arteko kodea lortu, eraiki, idatzi eta exekutatzeko tresnak eta liburutegiak eskuragarri jartzen ditu. Bestalde, *ROS*, lengoaiekiko independentea da eta hainbat lengoaiekin egin dezake lan: *C++*, *Python*, *Java*, etab.

Bere funtzionamenduari dagokionez, *ROS* grafo sistema batean oinarritzen da. Prozesuek *peer-to-peer* motako sare bat osatzen dute eta beraien arteko komunikazioa *ROS*-en komunikazio azpiegituran oinarritzen da. Komunikazio hau mota askotakoa izan daiteke, zerbitzu bidezko komunikazio sinkronoa edo *topic* bidezko komunikazio asinkronoa esaterako.

Jarraian, *ROS*-en lan egiteko modua hobeto ulertzeko kontzeptu batzuk azalduko dira.

3.3.2 Fitxategi-sistema

1. **Paketeak**
   Paketeak *ROS*-en softwarea antolatzeko unitate handienak dira. Pakete batek exekuzio garaiko prozesuak (nodoak), liburutegiak, datu multzoak, konfigurazio fitxate-

giak, edo elkar ondo antolatzea merezi duen edozer biltzen du. Paketeak dira *ROS*-en sortu daitezkeen gauza atomikoenak eta berrerabilpena ahalbidetzen dutenak.

2. **Metapaketeak**
   Pakete espezializatuak dira, erlazionatutako pakete talde bat adierazteko bakarrik balio dutenak.

3. **Pakete manifestuak**
   Manifestuak *XML* motako dokumentuak dira, non paketeen inguruko metadatuak gordetzen dituzten. Bertan, izena, bertsioa, deskribapena, lizentziaren inguruko informazioa, dependentziak, etab. daude jasota.

4. **Biltegiak**
   VCS (Version control system) elkarbanatzen duten pakete multzoak dira.

5. **Mezu motak**
   Mezuen deskribapena gordetzen duten fitxategiak dira, *ROS*-en bidaltzen diren mezuen egiturak azaltzen dituzte.

6. **Zerbitzu motak**
   Zerbitzuen deskribapena gordetzen duten fitxategiak dira, *ROS*-en zerbitzuetan bidaltzen diren eskaera eta erantzunen datu egiturak azaltzen dituzte.

7. ***Launch* fitxategiak**
   Nodo bat baino gehiago aldi berean exekutatzeko edo parametroak zehazteko balio duten fitxategiak dira.

3.3.3 Konputazio-grafoa

Lehen aipatu bezala, konputazio grafoa elkarrekin datuak prozesatzen dituzten *ROS*-eko prozesuek osatzen duten *peer-to-peer* sarea da. Jarraian, modu ezberdinetan bada ere, grafoari datuak pasatzen dizkioten konputazio-grafoko hainbat kontzeptu azalduko dira (Ikus 3.11 irudia):

1. **Nodoak**
   Nodoak konputazioa burutzen duten prozesuak dira. Robot batek normalean hainbat nodo izaten ditu, eta horietako bakoitzak zeregin zehatz bat izaten du. Esaterako,

nodo batek gurpilen motorrak kontrola ditzake, beste batek kokalekua eta beste batek guzti hori bistara dezake grafikoki. Nodo bat idazteko *roscpp* edo *rospy* moduko *ROS* bezero liburutegi bat erabili behar da.

2. **Gainbegiralea edo Master**
   Nodoen izen erregistroa ahalbidetzen du. Honi esker, nodoek elkar ikusi dezakete eta bata bestearekin komunikatu, mezuak trukatuz edo zerbitzuei deituz.

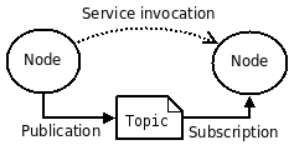
3. **Parametro zerbitzaria**
   Datuak kokaleku zentral batean gordetzea ahalbidetzen du, nodoek exekuziodenboran behar dituzten parametroak gorde edo eskuratzeko.

4. **Mezuak**
   Nodoak elkarrekin komunikatzeko erabiltzen dira mezuak. Mezu bat hainbat motako eremuak biltzen dituen datu egitura bat da. Oinarrizko motak (osokoak, boolearrak, errealak, etab.) eta hauez osatutako zerrendak onartzen dira.

5. ***Topic*-ak**
   Nodoen arteko mezu trukaketa argitaratu/harpidetu *(publish/subscribe)* ereduaren bidez burutzen denez, mezu horiek gordetzeko postontzi modukoak dira *topic*-ak. Hau da, nodo batek mezu bat bidaltzeko, *topic* batean argitaratzen du. Informazio hori jaso nahi duen nodoa berriz, *topic* horretara harpidetzen da. Horrela, *topic* batek hainbat argitaratzaile eta harpidedun izan ditzake beraien artean elkar ezagutu behar izan gabe. Komunikazio hau modu asinkronoan burutzen da.

6. **Zerbitzuak**
   Bestalde, batzuetan beharrezko da eskaera bati zuzeneko erantzuna ematea. Horretarako argitaratu/harpidetu eredua oso egokia ez denez, zerbitzuen bidez burutzen dira elkarrekintza horiek. Eskaera/erantzun motako komunikazioetan nahikoa dira bi mezu, bata eskaera burutzeko eta beste bat erantzuna jasotzeko, zerbitzari/bezero ereduaren modura. Nodo batek zerbitzu bat eskainiko du eta bezeroak eskaera burutu ondoren, erantzunaren zain geratuko da. Komunikazio hau modu sinkronoan burutzen da.

7. ***Bag* fitxategiak**
   ROS mezuen informazioa gorde eta berrerabiltzea ahalbidetzen dute. Esaterako, sentsore batetik jasotako datuak gorde ditzake aurrerago beste algoritmo batzuen neurketan erabiltzeko.



3.11 Irudia: ROS-en funtzionamendu diagrama

3.3.4 Instalazioa

Proiektu honetan erabiliko den *ROS* bertsioa, *ROS Indigo* izango da. Honen instalazioa alde batetik *Raspberry Pi* txartelean [8] eta bestetik norberaren ordenagailuan burutu beharko da, *Ubuntu* sistema eragilearen gainean [9]. Lehenengo kasua bigarrena baino konplexuagoa izan arren, badaude instalazio gidak prozesua errazten dutenak.

Raspberry Pi-ren instalazioari dagokionez, esteka honetan irakur daiteke instalazio gida osoa guk nahi dugun *ROS* bertsioarentzat: `http://wiki.ros.org/ROSberryPi`

Ordenagailuaren kasuan, *Ubuntu* plataformarako instalazio gida, esteka honetan dago: `http://wiki.ros.org/indigo/Installation/Ubuntu`

3.4 Hainbat gailu

3.4.1 Bateria

Raspberry Pi txartela elikatzeko, bateria txiki bat erabiliko da. Bateria hori autoarekin batera joan behar da, mugimendua izan dezan. Erabilitako eredua *Power Bank A5 Classic* da, 2600 mAh-koa (Ikus 3.12 irudia).

3.12 Irudia: Powerbank bateria

3.4.2 Wi-Fi txartela

Erabilitako *Raspberry Pi* ereduak ez du *Wi-Fi* txartelik, ondorioz, *USB* bidezko *Wi-Fi* txartela gehitu zaio. Erabilitako eredua *CISCO* etxeko *WUSB100v2* da (Ikus 3.13 irudia).

3.13 Irudia: Wi-Fi txartela

3.4.3 Urrutiko agintea

Autoa erraztasunez gidatzeko, teklatu itxurako urrutiko agintea erabiliko da. Urrutiko aginte hau *Avenzo* etxekoa da (Ikus 3.14 irudia).

3.14 Irudia: Urrutiko agintea

3.4.4 USB web kamera

Irudien transmisioa burutzeko *USB* konexioa duen web kamera bat erabili da. Kasu honetan *Qoopro* etxeko *11008* eredua (Ikus 3.15 irudia).

3.15 Irudia: Web kamera

4. KAPITULUA

Garapena eta egiaztapena

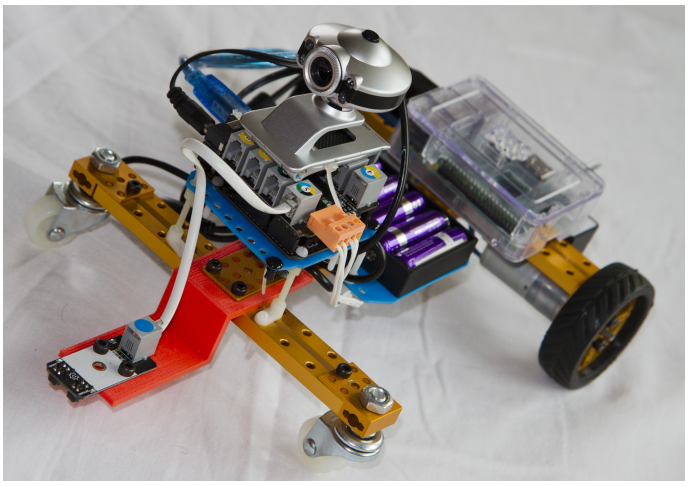
Aurkibidea

4.1 Eraikuntza

Makeblok etxeko robotaren eraikuntza, hasiera batean horrela izan bazen ere, ez da etxeak dakarren eraikuntza gida jarraituz burutu. Izan ere, gure robotak prototipo horrek baino gailu gehiago ditu: *Raspberry Pi* ordenagailu txikia, web kamera, etab. Gainera, sentsore infragorrien irakurketa ez zen ondo burutzen etxeak gomendatutako eraikuntzan, hori horrela izanik, 3D inprimagailuarekin komeni zen pieza garatu da.

Beraz, proiektuaren garapenarekin hasteko eman beharreko lehenengo pausoa, robotaren eraikuntza egokia da. Lortu nahi den itxura aztertu eta hori lortzeko behar diren piezak identifikatu eta lortu behar dira. Garrantzitsua da egitura sendo bat lortzea, eta piezak ez edozein modutan kokatzea. Izan ere, mugimenduan egongo den robota da eta egonkortasuna beharko du modu egokian mugitzeko.

Autoak lortutako azken itxura honakoa da:



4.1 Irudia: Autoaren itxura

4.2 Motorren kontrola

Autoak, korronte zuzeneko bi motor ditu, atzeko gurpil banari lotuta. Gurpil hauek kontrolatzeko, *Makeblock* etxeak dakarren liburutegia erabili da, izan ere, portuak huts hutsean kudeatu beharrean, zuzenean korronte zuzeneko motor bezala defini daitezke aldagaiak. Ondoren, aldagai hauei balio ezberdinak emango zaizkie lortu nahi den abiadura eta norabidearen arabera.

Beraz, esan bezala, *Makeblock* etxeak eskuragarri uzten ditu gailu ezberdinak modu errazean kontrolatzeko liburutegiak. Korronte zuzeneko motorrei dagokienez, jarraian agertzen diren bi kode zatietan hauek nola definitzen diren eta hauentzat sortutako funtzioak irakur daitezke.

```
///@brief Class for DC Motor Module
class MeDCMotor: public MePort
{
public:
   MeDCMotor();
   MeDCMotor(uint8_t port);
   void run(int speed);
   void stop();
};
```

```
/* MotorDriver */
MeDCMotor::MeDCMotor(): MePort(0)
{

}

...

void MeDCMotor::run(int speed)
{
   speed = speed > 255 ? 255 : speed;
   speed = speed < -255 ? -255 : speed;

   if(speed >= 0)
   {
      MePort::dWrite2(HIGH);
      MePort::aWrite1(speed);
   }
   else
   {
      MePort::dWrite2(LOW);
      MePort::aWrite1(-speed);
   }
}
void MeDCMotor::stop()
{
   MeDCMotor::run(0);
}
```

Ikus daitekeen bezala, garatzaileari lana errazteko, *run* eta *stop* funtzioak definitzen dira. Horrela, garatzaileak nahiko izango du aldagai bat *DCMotor* bezela definitu eta *run*

funtzioaren bidez nahi dion abiadura esleituz.

Liburutegia nola erabili ikasteko, hainbat adibide aurki daitezke *Makeblok* etxearen web orrialdean. Kasu honetan, *TestDCMotorDriver* izeneko fitxategian lehenago azaldutako funtzioak nola erabili erakusten da.

```
/*************************************************************************
* File Name : TestMotorDriver.ino
* Author : Steve
* Updated : Jasen
* Version : V1.0.1
* Date : 11/14/2013
* Description : Test for Makeblock Electronic modules of Me -Motor Driver. You can directly connect
                a motor to M1 or M2 Port, or connect to PORT_1 or PORT_2 through a DC motor driver.
* License : CC-BY-SA 3.0
* Copyright (C) 2013 Maker Works Technology Co., Ltd. All right reserved.
* http://www.makeblock.cc/
*************************************************************************/

#include <Makeblock.h>
#include <Arduino.h>
#include <SoftwareSerial.h>
#include <Wire.h>

MeDCMotor motor1(PORT_1);
MeDCMotor motor2(PORT_2);
MeDCMotor motor3(M1);
MeDCMotor motor4(M2);

uint8_t motorSpeed = 100;

void setup()
{

}

void loop()
{
   motor1.run(motorSpeed); // value: between -255 and 255.
   motor2.run(motorSpeed); // value: between -255 and 255.
   motor3.run(motorSpeed);
   motor4.run(motorSpeed);
   delay(2000);
   motor1.stop();
   motor2.stop();
   motor3.stop();
   motor4.stop();
```

```
41    delay(100);
42    motor1.run(-motorSpeed);
43    motor2.run(-motorSpeed);
44    motor3.run(-motorSpeed);
45    motor4.run(-motorSpeed);
46    delay(2000);
47    motor1.stop();
48    motor2.stop();
49    motor3.stop();
50    motor4.stop();
51    delay(2000);
52 }
```

Kode zati honek egiten duena erraza da, bi motor definitzen ditu eta hauei balio ezberdinak esleitzen joaten da. Horrela hauei lotutako gurpilak aurrera, atzera eta gelditu egingo dira hurrenez hurren.

4.3 Infragorrien kontrola

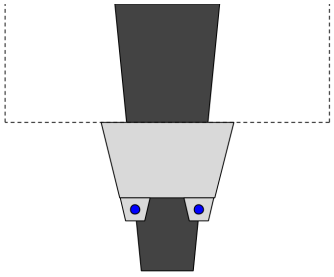
Motorrekin bezala, infragorrien kontrolerako *Makeblock* etxeak eskeintzen duen liburutegia erabili da. Liburutegi hau marra beltzaren detekziorako dago prestatuta eta dagokion kode zatia honakoa da.

```
1
2 //states of two linefinder sensors
3
4 #define S1_IN_S2_IN   0x00 //sensor1 and sensor2 are both inside of
      black line
5 #define S1_IN_S2_OUT   0x01 //sensor1 is inside of black line and
      sensor is outside of black line
6 #define S1_OUT_S2_IN   0x02 //sensor1 is outside of black line and
      sensor is inside of black line
7 #define S1_OUT_S2_OUT   0x03 //sensor1 is outside of black line and
      sensor is outside of black line
8
9          ...
10
11 ///@brief Class for Line Follower Module
12 class MeLineFollower: public MePort
13 {
14 public:
15    MeLineFollower();
16    ///@brief initialize
17    MeLineFollower(uint8_t port);
18    ///@brief state of all sensors
19    ///@return state of sensors
```

```
20    uint8_t readSensors();
21    ///@brief state of left sensors
22    bool readSensor1();
23    ///@brief state of right sensors
24    bool readSensor2();
25 };
```

```
1  /* LineFinder */
2  MeLineFollower::MeLineFollower(): MePort(0)
3  {
4
5  }
6  MeLineFollower::MeLineFollower(uint8_t port): MePort(port)
7  {
8
9  }
10 uint8_t MeLineFollower::readSensors()
11 {
12    uint8_t state = S1_IN_S2_IN;
13    bool s1State = MePort::dRead1();
14    bool s2State = MePort::dRead2();
15    state = ((1 & s1State) << 1) | s2State;
16    return state;
17 }
18 bool MeLineFollower::readSensor1()
19 {
20    return MePort::dRead1();
21 }
22 bool MeLineFollower::readSensor2()
23 {
24    return MePort::dRead2();
25 }
```

Hasieran, aldagai batzuk definitzen dira erabiltzailearen erraztasunerako. Sentsoreetatik jasotzen den informazioa huts hutsean kudeatu beharrean, hau da 0 eta 1-ekoekin, detektatutako datuekin konbinaketak eginez erraz erabiltzeko aldagaiak definitzen dira: bi sentsoreak marra beltzetik kanpo, biak barne, bat barne bestea kanpo...



4.2 Irudia: Marra jarraitzailearen goiko bista

Bestalde, motorren kasuan bezala, *Makeblock* etxeak atzigarri uzten du sentsore infragorriak testeatzeko programa bat. Funtzionamendua oso erraza eta sinplea da, sentsoreak irakurtzen ditu eta hauetatik lortzen dituen datuen arabera, sentsoreak marra beltzaren barne ala kanpo dauden esaten du. Informazio hori, serie lerroaren bitartez pantailaratzen du. Gainera, sentsore bakoitzaren goikaldean led argi bana dago. Hauek piztu egiten dira marra beltza detektatzen ez dutenean, hau da sentsorea marra beltzetik at dagoenean.

```
/**************************************************************************
* File Name : TestLineFollower.ino
* Author : Steve
* Updated : Jasen
* Version : V1.0.1
* Date : 11/14/2013
* Description : Example for Makeblock Electronic modules of Me -
                Line Finder. The module can only be connected to
                the PORT_3, PORT_4, PORT_5, and PORT_6 of Me -
                Base Shield.
* License : CC-BY-SA 3.0
* Copyright (C) 2013 Maker Works Technology Co., Ltd. All right
    reserved.
* http://www.makeblock.cc/
**************************************************************************/

#include <Makeblock.h>
#include <SoftwareSerial.h>
#include <Wire.h>

MeLineFollower lineFinder(PORT_3); //Line Finder module can only be
    connected to PORT_3, PORT_4, PORT_5, PORT_6 of base shield.

void setup()
{
   Serial.begin(9600);
}

void loop()
{
   int sensorState = lineFinder.readSensors();
   switch(sensorState)
   {
      case S1_IN_S2_IN:Serial.println("Sensor 1 and 2 are inside of
          black line");break;
      case S1_IN_S2_OUT:Serial.println("Sensor 2 is outside of black
           line");break;
      case S1_OUT_S2_IN:Serial.println("Sensor 1 is outside of black
           line");break;
      case S1_OUT_S2_OUT:Serial.println("Sensor 1 and 2 are outside
          of black line");break;
      default:break;
```

```
36    }
37    delay(200);
38 }
```

4.4 Marra jarraitzailea

Behin motorren kontrola eta infragorrien kontrola aztertu direla, bi hauen elkarketa egin behar da marra jarraitzailea sortzeko.

Hasiera batean sortutako programa egoeretan oinarritzen zen. Hau da, momentuko egoera eta aurreko egoerak gordetzeko aldagaiak zituen. Aldagai hauetan, bi sentsoreak marratik kanpo, barne ala bat kanpo eta bestea barne zeuden gordetzen ziren. Horrela, bi sentsoreak marratik kanpo agertzen zirenean, aurreko egoeretara begiratzen zen hartu beharreko portaera erabakitzeko.

```
1  //LIBURUTEGIAK
2  #include <Makeblock.h>
3  #include <SoftwareSerial.h>
4  #include <Wire.h>
5  #include <Arduino.h>
6
7  MeLineFollower lineFinder(PORT_6); //Sentsore infragorrien modulua
       shield-aren PORT_3, PORT_4, PORT_5 edo PORT_6-an egon daiteke
       konektatuta. Kasu honetan 6. portua erabiliko dugu.
8
9  MeDCMotor motor3(M1);
10 MeDCMotor motor4(M2);
11
12 int previousState0=S1_OUT_S2_OUT; //Previous State
13 int previousState1=S1_OUT_S2_OUT; //1xPre - previous State
14 int previousState2=S1_OUT_S2_OUT; //2xPre - previous State
15
16 void setup()
17 {
18    Serial.begin(9600);
19 }
20
21 void loop()
22 {
23    int sensorState = lineFinder.readSensors();
24    switch(sensorState)
25    {
26       case S1_IN_S2_IN:
27         Serial.println("Sensor 1 and 2 are inside of black line");
28         motor3.run(60);
29         motor4.run(60);
```

```
30          previousState2=previousState1; //2xPre - previous
31          previousState1=previousState0; //1xPre - previous
32          previousState0=S1_IN_S2_IN; //Previous
33          break;
34        case S1_IN_S2_OUT:
35          Serial.println("Sensor 2 is outside of black line");
36          motor3.run(100);
37          motor4.run(30);
38          previousState2=previousState1; //2xPre - previous
39          previousState1=previousState0; //1xPre - previous
40          previousState0=S1_IN_S2_OUT; //Previous
41          break;
42        case S1_OUT_S2_IN:
43          Serial.println("Sensor 1 is outside of black line");
44          motor3.run(30);
45          motor4.run(100);
46          previousState2=previousState1; //2xPre - previous
47          previousState1=previousState0; //1xPre - previous
48          previousState0=S1_OUT_S2_IN; //Previous
49          break;
50        case S1_OUT_S2_OUT:
51          Serial.println("Sensor 1 and 2 are outside of black line");
52          //AURREKO EGOERA BEGIRATU
53          switch(previousState0)
54          {
55            case S1_IN_S2_IN:
56
57               ...
58               (S1_OUT_S2_OUT kasuaren kode berdina)
59               ...
60
61            case S1_IN_S2_OUT:
62              Serial.println("Sensor 2 is outside of black line");
63              motor3.run(125);
64              motor4.run(30);
65              previousState2=previousState1; //2xPre - previous
66              previousState1=previousState0; //1xPre - previous
67              previousState0=S1_OUT_S2_OUT; //Previous
68              break;
69            case S1_OUT_S2_IN:
70              Serial.println("Sensor 1 is outside of black line");
71              motor3.run(30);
72              motor4.run(125);
73              previousState2=previousState1; //2xPre - previous
74              previousState1=previousState0; //1xPre - previous
75              break;
76            case S1_OUT_S2_OUT:
77              Serial.println("Sensor 1 and 2 are outside of black line")
                    ;
78              //1xPRE - PREVIOUS STATE ALDAGAIA BEGIRATU
79              switch(previousState1)
80              {
81                case S1_IN_S2_IN:
```

```
82
83                ...
84                (S1_OUT_S2_OUT kasuaren kode berdina)
85                ...
86
87              case S1_IN_S2_OUT:
88                Serial.println("Sensor 2 is outside of black line");
89                motor3.run(150);
90                motor4.run(30);
91                previousState2=previousState1; //2xPre - previous
92                previousState1=previousState0; //1xPre - previous
93                previousState0=S1_OUT_S2_OUT; //Previous
94                break;
95              case S1_OUT_S2_IN:
96                Serial.println("Sensor 1 is outside of black line");
97                motor3.run(30);
98                motor4.run(150);
99                previousState2=previousState1; //2xPre - previous
100               previousState1=previousState0; //1xPre - previous
101               previousState0=S1_OUT_S2_OUT; //Previous
102               break;
103             case S1_OUT_S2_OUT:
104               Serial.println("Sensor 1 and 2 are outside of black
                     line");
105               //2xPRE - PREVIOUS STATE ALDAGAIA BEGIRATU
106               switch(previousState2)
107               {
108                 case S1_IN_S2_IN:
109
110                   ...
111                   (S1_OUT_S2_OUT kasuaren kode berdina)
112                   ...
113
114                 case S1_IN_S2_OUT:
115                   Serial.println("Sensor 2 is outside of black line");
116                   motor3.run(200);
117                   motor4.run(30);
118                   previousState2=previousState1; //2xPre - previous
119                   previousState1=previousState0; //1xPre - previous
120                   previousState0=S1_OUT_S2_OUT; //Previous
121                   break;
122                 case S1_OUT_S2_IN:
123                   Serial.println("Sensor 1 is outside of black line");
124                   motor3.run(30);
125                   motor4.run(200);
126                   previousState2=previousState1; //2xPre - previous
127                   previousState1=previousState0; //1xPre - previous
128                   previousState0=S1_OUT_S2_OUT; //Previous
129                   break;
130                 case S1_OUT_S2_OUT:
131                   Serial.println("Sensor 1 and 2 are outside of black
                        line");
132                   //3xPRE - PREVIOUS STATE ALDAGAIA BEGIRATU
```

```
                motor3.run(0);
                motor4.run(0);
                previousState2=previousState1; //2xPre - previous
                previousState1=previousState0; //1xPre - previous
                previousState0=S1_OUT_S2_OUT; //Previous
                break;
              default: break;
            }
            break;
          default: break;
        }
        break;
      default: break;
    }
    break;
  default:break;
 }
 delay(50);
}
```

Ikusirik kode honek ez zuela behar bezala lan egiten, askotan marratik atera eta geldi geratzen zen eta, programa beste era batera burutzeko erabakia hartu da.

Funtzionamendua erraza da. Bi sentsore infragorriek marra beltza detektatuz gero, bi motorrak aurrerantz mugituko dira, hau da motorrei balio positibo bera emango zaie. Bestalde, sentsore batek marra beltza detektatzen badu eta besteak ez, biraketa burutuko da bi sentsoreek marra beltza detektatu arte. Horretarako, motor bati, balio positibo altu bat emango zaio eta besteari balio positibo baxuago bat. Azkenik, bi sentsoreak marra detektatzen ez badute, ausazko biraketa hasiko da bi sentsoreek marra detektatu bitartean.

Hau da garatutako kode zatia:

```
//LIBURUTEGIAK
#include <Makeblock.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <Arduino.h>

//SENTSORE INFRAGORRIAK
MeLineFollower lineFinder(PORT_6); //Sentsore infragorrien modulua
    shield-aren PORT_3, PORT_4, PORT_5 edo PORT_6-an egon daiteke
    konektatuta. Kasu honetan 6. portua erabiliko dugu.

//MOTORRAK
MeDCMotor motor1(M1);
MeDCMotor motor2(M2);

void setup()
{
```

```
16    Serial.begin(9600);
17 }
18
19 void loop()
20 {
21    int aux = 0;
22    int sensorState = lineFinder.readSensors(); //Sentsoreen egoera
          irakurri.
23    switch(sensorState)
24    {
25       case S1_IN_S2_IN:
26         motor1.run(60);
27         motor2.run(60);
28         delay(50);
29         break;
30       case S1_IN_S2_OUT:
31         while (sensorState != S1_IN_S2_IN){ //Bi sentsoreek marra
               beltza aurkitu arte jarraitu.
32           motor1.run(100);
33           motor2.run(30);
34           delay(50);
35           sensorState = lineFinder.readSensors(); //Sentsoreen egoera
                 irakurri.
36         }
37         break;
38       case S1_OUT_S2_IN:
39         while (sensorState != S1_IN_S2_IN){ //Bi sentsoreek marra
               beltza aurkitu arte jarraitu.
40           motor1.run(30);
41           motor2.run(100);
42           delay(50);
43           sensorState = lineFinder.readSensors(); //Sentsoreen egoera
                 irakurri.
44         }
45         break;
46       case S1_OUT_S2_OUT: //Ausaz mugitu
47         aux = random(0,2);
48         Serial.println(aux);
49         if (aux == 0){
50           motor1.run(60);
51           motor2.stop();
52         }else{
53           motor1.stop();
54           motor2.run(60);
55         }
56         delay(50);
57         break;
58       default:break;
59    }
60 }
```

Kode zati honek behar bezala lan egiten duela ziurtatu da, zirkuitu itxi batean biratuz.

4.5 Arduino eta Raspberry Pi-aren arteko komunikazioa

Arduino eta *Raspberry Pi* ordenagailu txikiaren arteko komunikazioa, serie lerroaren bidez burutzen da. Proiektu honi dagokionez, komunikazio hau norabide bakarrean egiten da, *Raspberry Pi*-tik *Arduino*-ra.

Komunikazio hau burutzeko, *Raspberry Pi*-an programazio lengoaia ezberdinak erabili daitezke. Proiektu honen kasuan, *Python*-ekin hasi zen lanean, baina gabezia batzuk zituela konturatuta *C* lengoaiara aldatzea erabaki da.

Python-ekin sortutako programan, beharrezkoa zen tekla bakoitza sakatu ondoren *"enter"* tekla sakatzea nahi zen karakterea irakurtzeko, *C* bidez sortutako programan berriz ez. Horretarako *WiringPi* liburutegia erabili da [10]. *C* lengoaian idatzitako *Raspberry Pi* txartelaren sarrera irteerako portuak kudeatzeko liburutegia zehazki. Jarraian agertzen den kode zatian ikus daiteke bere erabilera:

```
1  #include <stdio.h>
2  #include <termios.h>
3  #include <stdint.h>
4  #include <stdlib.h>
5  #include <string.h>
6  #include <errno.h>
7  #include <wiringPi.h>
8  #include <wiringSerial.h>
9
10 char device[] = "/dev/ttyACM0"; //Serie lerroa
11 int fd; //File descriptor
12 unsigned long baud = 9600; //Transmisio abiadura bera bi gailuetan
13
14 /* Programa nagusia */
15 int main(void){
16
17    printf("%s \n", "Raspberry startup!");
18    fflush(stdout);
19
20    /* Serie portua ezarri komunikaziorako */
21    if ((fd = serialOpen (device, baud)) < 0){
22       fprintf(stderr, "Unable to open serial device: %s\n", strerror
              (errno));
23       exit(1); //error
24    }
25
26    /* sarrera irteerako portuak konfiguratu wiringPi moduan */
27    if (wiringPiSetup() == -1){
28       fprintf (stdout, "Unable to start wiringPi: %s \n", strerror (
             errno));
29       exit(1); //error
```

```
30    }
31
32    char c = 'A';
33
34    while (1){
35       printf(" %c \n", c);
36       serialPutchar (fd, c);
37    }
38    return 0;
39 }
```

Adibide honetan, karaktere bakarra pasa zaio *Arduino* txartelari serie lerroaren bidez 9600 baudiotan, "A" karakterea zehazki.

Arduino txartelari dagokionez, serie lerroa kudeatzeko liburutegia gehitzea nahikoa da. Hona hemen adibidea:

```
1 #include <Arduino.h>
2 #include <SoftwareSerial.h>
3 #include <Wire.h>
4
5 void setup()
6 {
7 Serial.begin(9600);
8 }
9
10 void loop()
11 {
12  if (Serial.available()){
13    char c = Serial.read();
14 }
```

Horrela, *Arduino* txartelak, *Raspberry Pi*-tik bidalitako karakterea jasoko du "c" aldagaian, kasu honetan "A" karakterea.

4.6 Teklatuaren irakurketa

Autoa gidatu ahal izateko, teklatuaren irakurketa burutu beharko da. Gidariak tekla bat sakatuta, inolako beste teklarik sakatu gabe, *Raspberry Pi* txartelak karakterea jaso beharko du eta serie lerrotik bidali.

Horretarako, jarraian agertzen diren funtzioak erabili beharko dira:

```
static struct termios old, newe;

/*Terminal berriaren sarrera/irteerako konfigurazioa berrezarri */
void initTermios (int echo)
{
   tcgetattr(0, &old); /* terminalaren sarrera/irteerako
      konfigurazioa gorde*/
   newe = old; /* konfigurazio berrian konfigurazio zaharra ezarri
      */
   newe.c_lflag &= ~ICANON; /* sarrera/irteerako buferra desgaitu */
   newe.c_lflag &= echo ? ECHO : ~ECHO; /* echo modua ezarri */
   tcsetattr(0, TCSANOW, &newe); /* konfigurazio berria erabili */
}

/* Sarrera/irteerako balio zaharrak berrezarri */
void resetTermios(void)
{
   tcsetattr(0, TCSANOW, &old);
}

/* Karaktera irakurri eta pantailaratu */
char getch_(int echo)
{
   char ch;
   initTermios(echo);
   ch=getchar();
   resetTermios();
   return ch;
}

/* Karaktere bat irakurri pantailaratu gabe */
char getch(void)
{
   return getch_(0);
}

/* Karaktere bat irakurri pantailaratuz*/
char getche(void)
{
   return getch_(1);
}
```

Funtzio hauei esker, teklatuan sakatzen diren karaktereak zuzenean prozesatuko dira beste teklaturik sakatu behar izan gabe.

4.7 Auto gidatua

Behin *Arduino* eta *Raspberry Pi* txartelen arteko komunikazioa nola burutu jakinik eta teklatua zuzenean irakurtzeko funtzioak definituta daudela autoa gidatzeko programa

sortu behar da.

Funtzionamendua erraza izango da. Motorrak geldi egongo dira teklarik sakatzen ez den bitartean, hau da *Arduino* txartelak serie lerrotik informaziorik jasotzen ez duen bitartean. Bestalde, teklatutik "u" letra sakatuta mantentzean, autoa aurrera mugituko da, "m" letra sakatzean atzera, "k" letra sakatzean eskubira biratuko da eta "h" letra sakatzean ezkerretara. Exekuzioa gelditzeko "o" letra sakatu beharko da.

Raspberry Pi-an garatutako kode zatia honakoa da:

```
1  //Liburutegiak
2  #include <stdio.h>
3  #include <termios.h>
4  #include <stdint.h>
5  #include <stdlib.h>
6  #include <string.h>
7  #include <errno.h>
8
9  //wiring Pi
10 #include <wiringPi.h>
11 #include <wiringSerial.h>
12
13 char device[] = "/dev/ttyACM0"; //Serie lerroa
14 int fd; //File descriptor
15 unsigned long baud = 9600; //Transmisio abiadura bera bi gailuetan
16
17 static struct termios old, newe;
18
19 //Karaktere bat lortu teklatutik
20
21 /*Terminal berriaren sarrera/irteerako konfigurazioa berrezarri */
22 void initTermios (int echo)
23 {
24    tcgetattr(0, &old); /* terminalaren sarrera/irteerako
       konfigurazioa gorde*/
25    newe = old; /* konfigurazio berrian konfigurazio zaharra ezarri
       */
26    newe.c_lflag &= ~ICANON; /* sarrera/irteerako buferra desgaitu */
27    newe.c_lflag &= echo ? ECHO : ~ECHO; /* echo modua ezarri */
28    tcsetattr(0, TCSANOW, &newe); /* konfigurazio berria erabili */
29 }
30
31 /* Sarrera/irteerako balio zaharrak berrezarri */
32 void resetTermios(void)
33 {
34    tcsetattr(0, TCSANOW, &old);
35 }
36
37 /* Karaktera irakurri eta pantailaratu */
38 char getch_(int echo)
```

```
39 {
40    char ch;
41    initTermios(echo);
42    ch=getchar();
43    resetTermios();
44    return ch;
45 }
46
47 /* Karaktere bat irakurri pantailaratu gabe */
48 char getch(void)
49 {
50    return getch_(0);
51 }
52
53 /* Karaktere bat irakurri pantailaratuz*/
54 char getche(void)
55 {
56 return getch_(1);
57 }
58
59 /* Programa nagusia */
60 int main(void){
61
62    printf("%s \n", "Raspberry startup!");
63    fflush(stdout);
64
65    /* Serie portua ezarri komunikaziorako */
66    if ((fd = serialOpen (device, baud)) < 0){
67       fprintf(stderr, "Unable to open serial device: %s\n", strerror
           (errno));
68       exit(1); //error
69    }
70
71    /* sarrera irteerako portuak konfiguratu wiringPi moduan */
72    if (wiringPiSetup() == -1){
73       fprintf (stdout, "Unable to start wiringPi: %s \n", strerror (
          errno));
74       exit(1); //error
75    }
76
77    char c;
78
79    printf("---- AUTO GIDATUA -----------------\n");
80    printf("---- sakatu U aurrera mugitzeko ---\n");
81    printf("---- sakatu M atzera mugitzeko ----\n");
82    printf("---- sakatu K eskuinera biratzeko -\n");
83    printf("---- sakatu H ezkerrera biratzeko -\n");
84    printf("---- sakatu O programa bukatzeko --\n");
85
86    while (1){
87       c = getch();
88       printf(" %c \n", c);
89       serialPutchar (fd, c);
```

```
90        if (c=='o'){
91           break;
92        }
93     }
94     return 0;
95 }
```

Konpilatzeko eta exekutatzeko honakoa idatziko da:

```
# sudo gcc -o raspduino.o raspduino.cpp -lwiringPi -DRaspberryPi
# sudo ./raspduino.o
```

Aldi berean, *Arduino* txartelean kargatu beharreko kodea honakoa da:

```
1  #include <Makeblock.h>
2  #include <Arduino.h>
3  #include <SoftwareSerial.h>
4  #include <Wire.h>
5
6  MeDCMotor motor3(M1);
7  MeDCMotor motor4(M2);
8
9  uint8_t motorSpeed_1 = 0;
10 uint8_t motorSpeed_2 = 0;
11
12 void setup()
13 {
14 Serial.begin(9600);
15 }
16
17 void loop()
18 {
19   if (Serial.available()){
20     char c = Serial.read();
21     if (c == 'h') {
22       //Move right motor
23       motor3.run(90);
24       motor4.run(30);
25     } else if (c == 'k') {
26       //Move left motor
27       motor3.run(30);
28       motor4.run(90);
29     } else if (c == 'u'){
30       //Move both motors
31       motor3.run(60);
32       motor4.run(60);
33     } else if (c == 'm'){
34       //Move both motors
35       motor3.run(-60);
36       motor4.run(-60);
37     }
38   }
```

```
39   else{
40      motor3.run(0);
41      motor4.run(0);
42   }
43   delay(40);
44 }
```

Bi txartelak martxan daudela autoa gidatzeko prest egongo da.

4.8 Kameraren kontrola

Auto gidatuari beste ezaugarri bat gehituko zaio, web kamera bat eramango du goikaldean gidatu bitartean irudiak transmititu ditzan. Horretarako *ROS* softwarea erabiliko da, alde batetik *Raspberry Pi* txartelean eta bestetik ordenagailuan.

USB bidezko web kameraren kontrolerako, *usb_cam* paketea erabiliko da [11]. Pakete honek nodo bakarra du *"usb_cam_node"* deiturikoa eta *topic* bakarra argitaratzen du, irudia. Bestalde hainbat parametro ditu, irudiaren dimentsioak, pixelaren formatua, kontrastea eta argitasuna besteak beste. Pakete hau *Raspberry Pi* txartelean instalatu beharko da.

Esteka honetan aurkitu daiteke paketearen inguruko informazioa eta deskargazteko esteka: `http://wiki.ros.org/usb_cam`

Behin paketea instalatu dela, hau da exekutatu beharko den *launch* fitxategia:

```
1 <launch>
2   <node name="usb_cam" pkg="usb_cam" type="usb_cam_node" output="
       screen" >
3     <param name="video_device" value="/dev/video0" />
4     <param name="image_width" value="320" />
5     <param name="image_height" value="240" />
6     <param name="pixel_format" value="yuyv" />
7     <param name="camera_frame_id" value="usb_cam" />
8     <param name="io_method" value="mmap"/>
9   </node>
10 </launch>
```

Fitxategi honetan *usb_cam_node* nodoa jartzen da martxan parametro zehatz batzuekin. Lehenik bideo gailua zehazten da, ondoren irudiaren dimentsioak, pixel formatua, irudiaren marko idenfitikatzailea eta sarrera/irteera metodoa.

Bestalde, *USB* web kameratik jasotako irudiak ikusteko nodo bat behar da. Horretara-

ko *image_pipeline* paketeko *image_view* erabiliko da [12]. Hau *Raspberry Pi*-an instalatu daitekeen arren, errendimendua hobea izan dadin, ordenagailuan burutuko da instalazioa.

Pakete hau jarraian agertzen den estekan deskargatu daiteke: `http://wiki.ros.org/image_view`

Erabiliko dena *"image_view"* nodoa izango da. Nodo hau *"usb_cam_node"* nodoak argitaratzen duen irudiaren *topic*-era dago harpidetuta.

Behin instalatuta dagoenean exekutatuko den *launch* nodoa hurrengoa da:

```
<launch>
   <node name="image_view" pkg="image_view" type="image_view"
      respawn="false" output="screen">
      <remap from="image" to="/usb_cam/image_raw"/>
      <param name="autosize" value="true"/>
   </node>
</launch>
```

Lehen aipatu bezala, *USB* kamerak transmititzen duen irudia bistaratuko du.

Behin instalazioak burututa eta *launch* fitxategiak idatzita, bi gailuak konfiguratu beharko dira elkar komunikatu daitezen *ROS* nukleo bakarrarekin [13]. Horretarako honakoa egin beharko dugu:

1. Bien artean sare komunikazioa dagoela ziurtatu beharko da, *"ping"* komandoa erabili daiteke horretarako.

2. Ordenagailuan *ROS_MASTER_URI* aldagaiaren balioa aldatu beharko da, *ROS*-eko maisua *Raspberry Pi*-an egongo dela adieraziz. Lehenengo, *Raspberry Pi*-an agertzen den helbidea aztertu beharko dugu hurrengo komandoa exekutatuz:

   ```
   # echo $ROS_MASTER_URI
   ```

   Ondoren, helbide hau ordenagailuko *ROS_MASTER_URI* aldagaian ezarri beharko da *"localhost"*-en lekuan *Raspberry Pi* ordenagailuaren helbidea ezarriz:

   ```
   # export ROS_MASTER_URI=http://192.168.1.141:11311
   ```

3. Izenen ebazpena ondo burutzeko, ordenagailuko */etc/hosts* fitxategian *Raspberry Pi*-ari dagokion helbidea jarri beharko da.

4. *Raspberry Pi*-an *ROS*-en nukleoa martxan jarriko da, *roscore* exekutatuz.

5. *Raspberry Pi*-an *usb_cam* nodoari dagokion *launch* fitxategia exekutatu beharko da *roslaunch* bidez.

6. Ordenagailuan irudia ikusteko sortutako *launch* fitxategia exektutatu beharko da *roslaunch* bidez.

Pauso hauek jarraituta, *Raspberry Pi*-an konektatuta dagoen *USB* web kamerako irudiak ikusten dira ordenagailuan. Irudi hauen transmisioa denbora errealetik gertu burutzen da funtzio hau soilik exekutatzean. Beste funtzioekin batera exektutatzen bada berriz, auto gidatuarekin esaterako, irudien transmisioa nahiko mantsotzen da.

5. KAPITULUA

Ondorioak eta etorkizuneko lana

Aurkibidea

Azken kapitulu honetan proiektuaren emaitzak aztertuta lortu diren ondorioak azalduko dira. Bestalde, proiektu honek izan ditzakeen hobekuntzak, edo etorkizunean egiteko interesgarriak izan daitezkeen lanak aipatuko dira.

5.1 Ondorioak

Orokorrean, proiektua modu arrakastatsuan amaitu dela ondoriozta daiteke. Hasieran planteatutako helburu guztiak bete baitira eta gainera hobekuntza txiki batzuk egin dira errendimendua hobetze aldera.

Proiektu honen bitartez, gaur egun merkatuan pil pilean dabiltzan bi plataformaren ahalmena neurtu nahi izan da eta esan daiteke azterketa hori burutu dela.

Arduino txartela, gailu oso egokia da behe mailako periferikoen kontrol errazerako. Izan ere, hauen kudeaketa modu errazean egiteko prestatuta dago eta horrek garatzaileari oso laguntza handia suposatzen dio.

Bestalde, *Raspberry Pi* txartelak oso ahalmen handia du duen tamaina eta kosturako. Hasieran ezarritako funtzio guztiak betetzeko gai izan den arren, egia da bere geldotasuna nabaria dela karga handia duenean. Bestalde, softwarearen instalazio garaian, zailtasunak aurki daitezke sarritan. Dena dela, geroz eta gehiago erabiltzen den plataforma bat izanik hainbat laguntza gida aurki daitezke sarean.

Ondorio nagusi bezala esan daiteke proiektua modu arrakastatsuan bete dela hasieratik planteatutako tresnekin.

5.2 Etorkizunerako lana

Dokumentuaren hasieran esan bezala, proiektu hau oinarrizko proiektua da. Huts hutsetik hasi da plataforma berriekin lanean eta beraien errendimendua neurtu nahi izan da. Beraz, oraindik lan asko dago egiteko etorkizunera begira.

Autoak betetzen dituen funtzioak, oso sinpleak dira. Izan ere, proiektua burutzeko denbora mugatuta dago eta denbora asko pasa da teknologia berrien informazioa bilatzen eta hauei buruz ikasten.

Jarraian, denbora faltagatik burutu ez diren hainbat hobekuntza aipatzen dira:

1. Proiektuaren helburu nagusia ez izan arren, *Arduino* txartelak kontrolatzen duen marra jarraitzailea badago hobetzerik. Izan ere, kurbadun zirkuituetan ondo egiten du lan baina ez da berdina gertatzen zirkuitu karratudunetan. Horretarako, beharrezko izango litzateke angelu zuzen batera iristen denean, ausaz alde batera edo bestera mugitzen hastea denbora batean, eta ezer aurkitzen ez badu, beste aldera biratzen hastea marra aurkitu arte.

2. Sortutako bi funtzionamenduak bi programa ezberdin bezela daude gordeta. Hau da, marra jarraitzailea erabili nahi bada, hori da *Arduino* txartelean kargatu beharrekoa; auto gidatua erabili nahi bada berriz, dagokion programa kargatu beharko da. Hobekuntza bat izango litzateke bi programak bateratu eta aukeraketa *Raspberry Pi*-tik egin ahal izatea.

3. *Raspberry Pi B+* txartelaren eragina handia den arren, motz gelditu daiteke zenbait gauzetarako. Hobekuntza bat izango litzateke proiektua *Raspberry Pi 2*-ra migratzea eta gainera barne biltegirako gaitasun handiagoko txartel bat jartzea, 16 GB-eko txartela esaterako.

4. Autoa gidatzeko, teklatua erabiltzen den moduan, ondo egongo litzateke hizkiak jaso ordez geziekin kontrolatu ahal izatea. Horrela, gidatzea intuitiboagoa izango litzateke.

5. Robota autonomoki mugitzen da marra jarraitzailea ereduan, baina marra gabe autonomoki mugitzeko aukera izatea ondo egongo litzateke. Aukera bat web kamera bitartez, kolore bat detektatuz, kolore hori jarraitzea da. Horretarako *ROS* erabili beharko litzateke: irudia jasoko duen nodo bat erabili beharko litzateke lehenengo (oraingoan erabili den bezala *usb_cam_node* esaterako), irudi hori prozesatu eta kolore detekzioa egin beharko litzateke (horretarako *cmvision* moduko nodo bat erabili daiteke), nodo horrek kolorearen posizioa esango liguke eta posizio horren arabera, *Arduino* txartelari gurpilak modu batean ala bestean mugitzeko agindua bidaliko litzaieke.

Hobekuntza txiki hauez gain, hamaika dira egin daitezkeenak. Robota oso oinarrizkoa da eta beti posible izango da ezaugarri gehiago gehitzea. Esaterako posible litzateke kontrola *Bluetooth* bidezkoa izatea, ala *GPS* bat barneratzea momentuko kokapena esateko, ala sentsore gehiago gehitzea paretekin ez dadin kolpatu, etab.

Eranskinak

A. ERANSKINA

Erabilpen gida

A.1 Sarrera

Erabilpen gida honen bitartez, sortutako robota erabiltzeko eman beharreko pausoak azaltzen dira.

Dokumentuan azaldu bezala, hiru dira robotaren funtzionalitate nagusiak: marra jarraitzailea, auto gidatua eta kameraren irudi transmisioa. Hortaz, erabilpen gida hiru zatitan banatzen da hauetako bakoitzaren erabilera azalduz. Zehaztu beharra dago, bakoitza bere aldetik azaltzen den arren, hirugarren funtzionalitatea konbinagarria dela bai lehenengo bai azkenarekin.

Erabilpen gida honetan ez da instalazioen inguruko ezer azaltzen, horretarako gainerako proiektuaren dokumentaziora jotzea gomendatzen da.

A.2 Marra jarraitzailea

Autoa marra beltz bat jarraitzeko gai da robotaren aurrekaldean dituen sentsore infragorriei esker. Hau martxan jartzeko honako pausoak jarraitu beharko dira:

1. Lehenik eta behin, sortutako kodea konpilatu eta txartelera igo beharko da. Horretarako, *Arduino* txartela ordenagailuarekin konektatu beharko da *USB* kable baten bidez. Ondoren, goikaldean azaltzen den gezia sakatu beharko da programa konpilatu eta txartelean kargatzeko.

2. Behin programa txartelean dagoela, *Arduino* txartelaren gainean dagoen shieldari korrontea eman behar zaio. Horretarako bateriak konektatu beharko zaizkio.

3. Azkenik, autoa zirkuitu gainean kokatu beharko da. Sentsore infragorriak marra beltzaren gainean kokatuta daudelarik, shieldaren etengailua *ON* posizioan ezarri beharko da.

A.3 Auto gidatua

Autoa norberak gidatu ahal izango du teklatuaren bidez, aurrera, atzera, ezker edo eskuinera mugituz. Horretarako jarraitu beharreko pausoak hauek dira:

1. Lehenik eta behin, sortutako kodea konpilatu eta txartelera igo beharko da. Horretarako, *Arduino* txartela ordenagailuarekin konektatu beharko da *USB* kable baten bidez. Ondoren, goikaldean azaltzen den gezia sakatu beharko da programa konpilatu eta txartelean kargatzeko.

2. Behin programa txartelean dagoela, *Arduino* txartelaren gainean dagoen shieldari korrontea eman behar zaio. Horretarako bateriak konektatu beharko zaizkio eta shieldaren etengailua *ON* posizioan ezarri beharko da.

3. Hurrengo pausoa, *Raspberry Pi* eta *Arduino* txartelen artean konexioa ezartzea da. Horretarako *USB* kablea erabiliko da.

4. *Raspberry Pi* ordenagailu txikia pizteko, nahikoa izango da bateria konektatzearekin.

5. Ordenagailuaren eta *Raspberry Pi* txartelaren konexioa egiaztatu behar da, horretarako nahikoa da *"ping"* komandoa erabiltzea *Raspberry Pi*-aren IP helbidearekin. Konexioa eginda dagoenean, *Raspberry Pi*-a ordenagailutik kontrolatu ahal izateko *ssh* konexioa ezarriko da.
   ```
   # ssh -X pi@192.168.1.140
   ```

6. Sortutako programa konpilatu eta exekutatu behar da, horretarako:
   ```
   # sudo gcc -o raspduino.o raspduino.cpp -lwiringPi -DRaspberryPi
   # sudo ./raspduino.o
   ```

7. Autoa gidatzeko, jarraian azaltzen diren teklak sakatu beharko dira:
   u tekla: Aurrera mugitzeko.
   m tekla: Atzera mugitzeko.
   k tekla: Eskubira biratzeko.
   h tekla: Ezkerrera biratzeko.
   o tekla: Programaren exekuzioa geldiarazteko.

A.4 Kamera

Raspberry Pi txartelera konektatutako *USB* kamera batek jasotako irudiak transmitituko ditu ordenagailuak zuzenean.

1. Lehenengo *USB* kamera *Raspberry Pi* txartelera konektatu behar da.

2. *Raspberry Pi* ordenagailu txikia pizteko, nahikoa izango da bateria konektatzearekin.

3. Ordenagailuaren eta *Raspberry Pi* txartelaren konexioa egiaztatu behar da, horretarako nahikoa da *"ping"* komandoa erabiltzea *Raspberry Pi*-aren IP helbidearekin. Konexioa eginda dagoenean, *Raspberry Pi*-a ordenagailutik kontrolatu ahal izateko *ssh* konexioa ezarriko da.

   ```
   # ssh -X pi@192.168.1.140
   ```

4. *Raspberry Pi*-an *ROS*-en muina martxan jarri beharko da, baita *USB* kemeratik irudiak jasotzen dituen nodoa. Horretarako sortutako *launch* fitxategia jaurti beharko da.

   ```
   # roscore # roslaunch usb_cam.launch
   ```

5. Ordenagailuan *ROS*-en muina *Raspberry Pi*-an exekutatuko dela adierazi behar da, horretarako *Raspberry Pi*-aren IP helbidea erabili behar da:

   ```
   # export ROS_MASTER_URI=http://192.168.1.140:11311
   ```

6. Ordenagailuan *usb_cam* nodotik jasotako irudia bistaratuko duen nodoa jarriko da martxan. Horretarako sortutako *launch* fitxategia jaurti behar da:

   ```
   # roslaunch image_view.launch
   ```

B. ERANSKINA

Bilera aktak

B.1 Konstituzio bilera

Data: 2015/03/02, 12:00 Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Elena Lazkano
- Txelo Ruiz

Helburuak

- Proiektuaren helburuak eta nondik norakoak finkatu.

Bileran hitz egindakoa eta hartutako erabakiak

- Auto itxurako robot mugikorra burutuko da.
- *Starter Robot Kit*-en oinarrituko da.

Egin beharrekoak

Goiatzek:

- *Meduino* txartelaren inguruko informazioa bilatu.
- Motor eta infragorrien funtzionamendua ulertu.
- Proiektuaren plangintza burutu.

Txelok:

- Proiektuaren garatzailea 1.3 laborategira sartzeko baimena eskatu.

B.2 Lehenengo bilera

Data: 2015/03/27, 14:00

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Txelo Ruiz

Helburuak

- Proiekturako beharrezko materiala jaso.
- Eman beharreko hurrengo pausoak finkatu.

Aurretik egin beharrekoak

- *Meduino* txartelaren inguruko informazioa lortu.

Bileran hitz egindakoa eta hartutako erabakiak

- *Starter Robot Kit*-az gain *Raspberry Pi* eta *ROS*-ekin egingo da lan.
- Autoak marra beltz bat jarraitzeko gai izan beharko du modu autonomoan.
- Autoa norberak gidatu ahal izango du *Raspberry Pi*-tik informazioa bidaltzen delarik.
- Autoak kamera bat izango du irudiak transmititzeko.

Egin beharrekoak

- Plataforma hauen inguruko informazioa bilatu.
- Beharko den materiala identifikatu.

B.3 Bigarren bilera

Data: 2015/04/17, 14:00

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal

- Elena Lazkano
- Txelo Ruiz

Helburuak

- Egindakoa erakutsi.
- Zalantzak argitu.
- Eman beharreko hurrengo urratsak finkatu.

Aurretik egin beharrekoak

- Plataformen inguruko informazioa bilatu.
- Beharko den materiala identifikatu.

Bileran hitz egindakoa eta hartutako erabakiak

- Kamera moduan *USB* web kamera arrunta erabiliko da.
- Robotaren funtzionamendu eskema orokorra finkatu da.
- Sentsore infragorrien konexioak aztertu dira.

Egin beharrekoak

- Produktuaren garapenarekin jarraitu.

B.4 Hirugarren bilera

Data: 2015/05/07, 16:30

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal

- Txelo Ruiz

Helburuak

- Zalantzak argitu.

Aurretik egin beharrekoak

- Proiektuaren garapenarekin jarraitu.

Bileran hitz egindakoa eta hartutako erabakiak

- Robotaren atzeko sentsore infragorrien informazioa ez da lortu, ondorioz hauek ez erabiltzea erabaki da.

B.5 Laugarren bilera

Data: 2015/06/01, 13:00

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Elena Lazkano
- Txelo Ruiz

Helburuak

- Egindakoa erakutsi.
- Zalantzak argitu.
- Hurrengo pausoak finkatu.

Aurretik egin beharrekoak

- Proiektuaren garapenarekin jarraitu.

Bileran hitz egindakoa eta hartutako erabakiak

- Sentsore infragorriak ez dute ondo irakurtzen kokatutako tokian, ondorioz, pieza bat sortuko da 3D inprimagailuarekin.
- *Arduino* eta *Raspberry Pi*-ren arteko konexioa lortu da, baina *Pythonen* teklatua irakurtzeko *"enter"* sakatu behar da. Ondorioz, *C* lengoaiarekin saiatzea erabaki da.
- *USB* bidezko *Wi-Fi* txartel bat eskatu da *Raspberry Pi*-an konektatzeko.

Egin beharrekoak

- Produktuaren garapenarekin jarraitu.

B.6 Bosgarren bilera

Data: 2015/06/23, 13:30

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Elena Lazkano
- Txelo Ruiz

Helburuak

- Autoan piezak kokatu

Aurretik egin beharrekoak

- Proiektuaren garapenarekin jarraitu.

Bileran hitz egindakoa eta hartutako erabakiak

- Autoaren piezak kokatu dira, sortutako eta lortutako zenbait piezekin.
- Torloju sendoago batzuk erosi eta jartzea erabaki da.
- Memoria LaTeX bidez idaztea erabaki da.

Egin beharrekoak

- Produktuaren garapenarekin jarraitu.
- Torlojuak erosi eta jarri.

B.7 Seigarren bilera

Data: 2015/07/08, 10:30

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Elena Lazkano

Helburuak

- Zalantzak argitu

Aurretik egin beharrekoak

- Proiektuaren garapenarekin jarraitu.

Bileran hitz egindakoa eta hartutako erabakiak

- *usb_cam* paketea instalatzea lortu da.
- *Raspberry Pi*-aren errendimendua hobea izan dadin, *ROS* ordenagailuan eta *Raspberry Pi*-an exekutatzea erabaki da, lan karga banatuz.

Egin beharrekoak

- Produktuaren garapenarekin jarraitu.
- Memoria idatzi.

B.8 Itxiera bilera

Data: 2015/08/27, 12:00

Lekua: Informatika Fakultatea

Bilerara hurbilduak:

- Goiatz Irazabal
- Txelo Ruiz

Helburuak

- Bidalitako txostenaren berrikuspena.

Aurretik egin beharrekoak

- Txostena bidali.

Bileran hitz egindakoa eta hartutako erabakiak

- Txostena berrikusi eta aldatu beharko atalak zehaztu dira.

Egin beharrekoak

- Txostena zuzendu, amaitu eta *ADDI* plataformara igo.
- Aurkezpena prestatu.

Bibliografia

[1] Arduino. 2015eko uztailan atzitua, Arduino etxearen webgunetik: https://www.arduino.cc/

[2] Raspberry Pi. 2015eko uztailan atzitua, Raspberry Pi etxearen webgunetik: https://www.raspberrypi.org/

[3] Robotika eta Sistema Autonomoen Ikerketa Taldea (RSAIT). 2015eko maiatzean atzitua, RSAIT taldearen webgunetik: http://www.sc.ehu.es/ccwrobot

[4] Makeblock. 2015eko uztailan atzitua, Makeblock etxearen webgunetik: http://makeblock.es/

[5] Arduino. 2015eko uztailan atzitua, Wikipedia webgunetik: https://es.wikipedia.org/wiki/Arduino

[6] Wikipedia Raspberry Pi. 2015eko uztailan atzitua, Wikipedia webgunetik: https://en.wikipedia.org/wiki/Raspberry_Pi

[7] ROS Start Guide. 2015eko uztailan atzitua, ROS Wiki webgunetik: http://wiki.ros.org/ROS/StartGuide

[8] Installing ROS in Raspberry Pi. 2015eko uztailan atzitua, ROS Wiki webgunetik: http://wiki.ros.org/ROSberryPi

[9] Installing ROS Indigo in Ubuntu. 2015eko uztailan atzitua, ROS Wiki webgunetik: http://wiki.ros.org/indigo/Installation/Ubuntu

[10] Henderson, G. WiringPi Library. [Blog argitalpena] 2015eko uztailan atzitua: https://projects.drogon.net/raspberry-pi/wiringpi/serial-library/

[11] USB_CAM ROS package. 2015eko uztailan atzitua, ROS Wiki webgunetik:
http://wiki.ros.org/usb_cam

[12] IMAGE_VIEW ROS package. 2015eko uztailan atzitua, ROS Wiki webgunetik:
http://wiki.ros.org/image_view

[13] Running ROS across multiple machines. 2015eko uztailan atzitua, ROS Wiki webgunetik:
http://wiki.ros.org/ROS/Tutorials/MultipleMachines