eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Informatika Ingeniaritzako Gradua

Software Ingeniaritza

Gradu Amaierako Proiektua

D-Lappse: time-lapseak egiteko dolly bat automatizatzeko Android aplikazioa

---

Egilea

Eneko Irazusta Aizpurua

informatika fakultatea
facultad de informática

2015

Laburpena

Jakina da informatika oso gai zabala dela. Horregatik, oro har, sistema informatikoen garatzaileak sistemaren osagai bakan batzuetaz soilik arduratzen dira. Proiektu honek hardwarearen eta softwarearen munduak uztartzea du helburu; hardware- eta software-osagaiak dituen sistema bat sortu, behar diren osagai guztiak garatuz. Horretaz gain, garatutakoa erabilgarria izatea bilatu da, hau da, funtzio praktiko eta erreal bat edukitzea.

Horretarako, Android eta Arduino plataformak aztertu dira: Android erabiltzaileari interfaze grafikoa eskaini eta elkarrekintza burutzeko erabili da; Arduino, berriz, hardwarearen kontrolatzailea izateko. Horrekin, eragingailuak kontrolatuz, time-lapseak egiteko sistema automatizatua garatu da; D-Lappse Android aplikazioarekin sistema kontrolatu daiteke eta dollyari aginduak bidali, argazki-sekuentziak era automatizatu batean egiteko.

Hitz gakoak: Android, Arduino, time-lapse, dolly.

Gaien aurkibidea

1. KAPITULUA

Proiektuaren eraketa

1.1 Sarrera

Dokumentu hau Euskal Herriko Unibertsitatean, Informatika Fakultatean egindako Gradu Amaierako Proiektuaren memoria da. Dokumentu honetan proiektuaren nondik norako garrantzitsuenak azaltzen dira.

1.2 Motibazioa

Gradu-amaierako proiektua, gradua amaitzeko lan edota betebehar soil bat bezala ikusi beharrean, gradu osoan ikasitako gaitasunak proban jartzeko aukera bezala ikusi nahi izan da. Alde batetik, software-ingeniaritza espezialitateko ikasle izanik, softwarearen garapenaren arloan interesa bistakoa da, baina gutxiago ikusi ahal izan den hardwarearen mundua ere interesgarria den atala da. Hori kontuan hartuta, bi gaiak uztartzen dituen proiektu bat egiteko nahia sortu da.

Hori kontuan izanik, eta interes pertsonalak ere tarteko, ikus-entzunekoen master bateko ikasle ohia izaki, erabilgarria suertatu daitekeen gailu bat garatzea erabaki da.

1.3 Proposatutako soluzioa

Time-lapseak egiteko sistema automatizatua deituko diogu garatuko dugun gailuari. Gainetik azaltzeko, sistema hau gai izango da era automatizatu batean *time-lapse*[1] argazki-sekuentziak ateratzeko eta baita kameraren desplazamendu lineal bat sortzeko ere *dollyari*[2] esker. Ez da nahikoa izango aplikazio bakar bat sistema guztia kontrolatu ahal izateko. Edozein sistema automatizatuk izan ohi duen moduan, alde batetik aginteko atala izango du eta beste aldetik, atal eragilea:

Aginte-atala: erabiltzaileak sistemarekin elkarrekintza izateko aplikazioa izango da agin-terako erabiliko dena. Aplikazio hau Android gailu batean exekutatuko da eta atal eragilearen urruneko aginte bezala funtzionatuko du. Atal hau ondo lantzen egingo da esfortzurik handiena.

Atal eragilea: dollya bera fisikoki maneiatzeko kontrolatzaile bat izango dugu, *A*rduino[3] plaka bat hain zuzen ere. Arduinoak dollyaren motor, sentsore eta gainontzeko eragingailuak kontrolatuko ditu, eta aginte-ataletik iristen diren aginduak behar den moduan aurrera eramatea izango da bere helburua.

Software eta hardware informatikoaz aparte, dollya fisikoki ere eraiki behar da; bertan, Arduino kontrolatzaileaz gain, beharrezko sentsoreak jarri behar dira, baita *DSLR*[4] kamera bat ere.

1.4 Dokumentuaren egitura

Dokumentuak modu honetan egituratua dago:

- Proiektuaren eraketa
- Proiektuaren kudeaketa-plana
- Aurrekarien azterketa

[1] *Time-lapse* argazkilaritza teknika bat da; oso motel gertatzen diren gertaerak atzeman eta abiadura handiagoan erreproduzitzea du helburu.

[2] *Dolly*-a tresna zinematografiko bat da, errail batzuen gainean doa eta mugimendu naturalak sortzeko balio du; kamera dollyaren gainean finkatzen da.

[3] *Arduino* hardware libreko plataforma bat da, mikrokontroladoredun plaka bat eta garatze-ingurune bat dituena.

[4] *DSLR* (Digital Single Lens Reflex) objektibo bakarreko reflex argazki-kamera mota bat da.

- Sistemaren funtzionalitateen analisia
- Softwarearen diseinua
- Hardwarearen diseinua eta garapena
- Softwarearen inplementazioa
- Sistemaren ebaluazioa eta hobekuntzak
- Jarraipena eta kontrola
- Ondorioak
- Eranskinak

2. KAPITULUA

Proiektuaren kudeaketa-plana

Kapitulu hau proiektuaren irismenaren plangintza eta jarraipen eta kontrola zein eratan egingo den definitzean zentratzen da. Hauek dira aztertuko diren gaiak:

- Helburuak
- Irismena
- Lan-metodologia
- Baliabideak
- Proiektua gauzatzeko plana
- Kalitate-plana
- Arriskuak
- Parte interesatuak

2.1 Helburuak

Helburu batzuk Gradu Amaierako Proiektuaren izaeragatik definituak daude, memoria idaztea esaterako. Gainontzekoak, proiektuaren irismena finkatzean definitu dira, baina baliteke helburu horiek ere denborarekin aldatzen joatea. Ondoren, proiektuaren helburuak definitu dira:

- Helburu nagusiak
  - Time-lapseak egiteko sistema automatizatua garatzea
    - Dollya fisikoki eraikitzea.
    - Arduino aplikazio bat dollya kontrolatzeko (dollyaren driverra).
    - Android aplikazio bat sistema kontrolatzeko.
  - Proiektuaren garapenaren nondik norakoak biltzen dituen memoria idaztea.
  - Proiektuaren defentsarako aurkezpena egitea.
- Helburu gehigarriak
  - Zeharkako trebetasunak eskuratzea
    - Android garapenean ezagutza lortzea.
    - Arduino plaka ezagutzea.
    - Elektronikaren inguruan ezagutza zabaltzea.
    - Ikasketa autonomoa eta arazoei aurre egiteko gaitasunak frogan jartzea.

2.2 Irismena

Hasieratik argi utzi beharra dago proiektu honekin ez dela bilatzen teknologia berritzaile bat garatu edota argazkilaritza-munduan iraultza bat sortzea. Funtsean, ditugun baliabideak erabiliz, funtzionala den produktu bat sortu nahi da. Proiektuaren irismena, beraz, helburu horretan zentraturik dago.

2.2.1 Betekizunak

EHUko Informatika Fakultateak hainbat jarraibide ematen ditu Gradu Amaierako Proiektuari begira. Horretarako, fakultatearen webgunean [EHU, 2015] arautegi bat dago zintzilikatua. Honako hauek dira arautegian definiturik dauden betebehar batzuk:

- *Gradu Amaierako Proiektua defendatu eta ebaluatu ahal izateko, egiaztatu egin beharko da ikasleak ikasketa planeko gainerako gai guztiak gaindituta dituela eta hortaz, gradu amaierako lanari dagozkion kredituak izan ezik, graduko titulua lortzeko beharrezkoak diren gainerako kreditu guztiak dituela.*

- *Informatikaren Ingeniaritzako Graduan, lan hau Gradu Amaierako Proiektua izeneko derrigorrezko irakasgaiari dagokio, 12 kredituko irakasgaia.* 12 ETCS kreditu, gutxienez 300 ordu.
- *Matrikulatutako ikasleek fakultateko idazkaritzan aurkeztu beharko dute Gradu Amaierako Proiektuari dagokion memoria eta inprimakina modu digitalean, eta beharrezkoa balitz paperean ere.*
- *Memoriaren formatuak fakultateko estandarrari jarraituko dio.*

Betebehar horiek eta GAP araudia dokumentuan definitzen diren guztiak proiektuaren betekizunak izango dira.

2.2.2 Mugak

Ataza batzuek eskatzen duten denbora oso handia izaten da, eta ditugun baliabideak mugatuak direla jakinik, zentzuz erabili beharra dago denbora. Denbora asko eskatzen duen ataletako bat kudeaketa da; kudeaketa-lanak ez luke inoiz traba bat izan behar proiektuaren garatze-prozesuan, eta honenbestez, behar-beharrezkoak diren gaiak soilik azalduko dira txosten honetan; hau da, proiektuaren irismenean barneratzen den guztia.

Beste ildo batetik, erabiliko diren teknologien inguruko ikasketa eta ikerkuntza ez da behar-beharrezko jakintza eskuratzetik harago joango, 300 orduen mugara iritsi ezean behintzat. Antzeko zerbait egingo da produktuaren hardwarearekin, bere funtzionalitatea lehenbailehen lortzeari egingo zaio jaramon, eta xehetasunak ondoren aztertuko dira.

Nahiz eta 300 orduko proiektua izan, litekeena da denbora gehiago behar izatea proiektuaren kalitate-maila egokia lortzeko; denbora maximoaren muga bat jartzearren, 400 orduko dedikazio maximoa jarriko dugu, nahiz eta nahiko estimazio goiztiarra den.

2.2.3 Produktuaren irismena

Alde batetik, hardware ingurunea dugu; Arduino kontrolatzailea eta dollya. Dollya eskuz egindako gailua izango da, oraindik definitzeke dauden materialekin. Dollyak Arduino kontrolatzailea eramango du bera kontrolatu dezan baita DSLR bat ere. Dollya eta Arduino kontrolatzailea kontrolatzeko, Android aplikazioa izango dugu eta aplikazio honen bitartez erabiltzaileak sistema kontrolatu ahal izango du.

Sistemak, argazkigintzan formakuntzarik handirik ez duen pertsona bati, modu erraz batean, time-lapse argazki-sekuentziak ateratzea ahalbidetuko dio. Atal hau izango da gure sistemaren puntu indartsuena produktuari begira, sistema konplexu bat modu erraz eta intuitiboan kontrolatzeko aukera.

2.2.4 Irismen-mailak

Proiektuan erabiliko diren teknologiak ez zaizkigu ezagunak, eta, dollya eraikitzeak suposatuko duen denbora-kostua oso zaila denez estimatzen, proiektuaren irismena hiru mailatan definitu da. Irismen-maila hauek produktuari eragingo diote gehienbat, baina azkenean proiektuaren kalitatean eragin zuzena izango dute. Hauek dira definitutako irismen-mailak:

1. Lehen prototipo funtzionala: time-lapseak egiteko sistema automatizatuaren lehen prototipo funtzionala garatuko dugu. Xehetasun guztiak zaindu beharrean, funtzionamendu onargarria lortzen saiatuko gara.
2. Erroreak eta hobekuntzak: behin sistema erabilgarria dugunean, beste xehetasunak zainduko ditugu, besteak beste, erroreak zuzendu, elkarrekintza erraztu eta kodea optimizatu.
3. Funtzio osagarriak: sistemak behar dituen oinarrizko funtzioetatik aparte funtzio gehiago gehitzea izango da azken iterazio honen helburua.

Lehenengo iterazioaren amaieran, produktuaren oinarrizko irismenak betea behar luke, eta bigarren iterazioaren amaieran, kalitate-planak definitzen dituen adierazle gehienak ere bete beharko lirateke. Proiektuaren hasiera den honetan, goizegi da ziur esateko hirugarren irismen-maila betetzeko denbora izango dugun, eta horretxegatik utzi ditugu kritikoak ez diren eginbeharrak azkenerako.

Kontuan hartu behar den zerbait da irismen-maila bakoitzak sistemaren gailu eta osagai guztiak zeharkatuko dituela, hau da, Android aplikazioa, Arduino aplikazioa, hardwarea, etab.

2.2.5 Irismenaren kudeaketa

Gerta liteke hasiera batean finkatutako irismena aldatu beharra izatea, atzerapenak medio edota parte interesatuen eskaerei erantzuna emateko. Edonola ere, proiektuaren irisme-

na aldatzeak proiektuaren mugak edota produktuaren irismena aldatzea ekar lezake, eta inola ere ezin dira finkatutako betebeharrak bertan behera utzi. Beste edozeren aurretik proiektuaren arrakasta bermatu behar da eta aldaketak norabide horretan egingo dira.

Proiektuaren irismenean eragiten duten aldaketak ez dira memento batetik bestera erabakiko, plangintza berraztertu beharko da eta behar diren aldaketak egin irismen berrira egokitzeko.

2.2.6 Lanaren Deskonposaketa Egitura

LDEak proiektuaren irismena definitzen du eta lana ataza eta azpiatazetan banatzen du, era hierarkiko batean. Azpiataza bakoitza programatu eta gainbegiratu daiteke, baita bere kosteak estimatu ere.

Proiektua bost ataza nagusitan banatu da: kudeaketa, ikerkuntza, arkitektura, garapena eta itxiera.

Kudeaketa

Proiektuaren kudeaketak plangintza eta azpiatazen jarraipen eta kontrola biltzen ditu. Plangintza ez da egutegi bat definitzea soilik; horretaz gain, arriskuen kudeaketa-plana edota kalitate-plana ere biltzen ditu, besteak beste.

Ikerkuntza

Erabiliko diren teknologiak eta garapen-inguruneak hobeto ulertzeko, ezinbestean egin beharreko lana da ikerkuntza. Baita denbora aurrezten lagunduko duten baliabideak bilatzeko ere, horien artean, kode-zatiak, liburutegiak, etab. Proiektu honetan berebiziko garrantzia duen atala da hau, erabiliko diren teknologiak ez baitira ezagunak.

Arkitektura

Edozein proiektuk bere analisi eta diseinua beharrezkoak ditu. Alde batetik, analisian, proiektuaren helburuak betetzeko osagaiak identifikatzen dira, eta diseinuan, osagai bakoitzaren barne-egitura definitu behar da.

Garapena

Hau da denbora gehien eskainiko zaion atala. Alde batetik, ingurunearen prestaketa daukagu, bai software aldetik eta baita hardware aldetik ere. Ondoren, programazioak hartuko du garrantzia. Programazio aldetik, Arduino aplikazioaren garapena dugu eta baita Android aplikazioarena ere. Aplikazioak garatzearekin batera probak ere egingo dira.

Itxiera

Proiektuaren bizi-zikloa amaitze aldera, memoria idazteak edota defentsa prestatzeak garrantzia hartuko dute. Nahiz eta garapenarekin batera ataza hauek ere paraleloan egiten joan behar den, itxiera-prozesuan izango da dena txukun eta modu antolatu batean bilduta uzteko unea.

2.1 irudian, proiektuaren LDEa ikus daiteke, eta, segidan, ataza nagusi eta erdi mailakoen laburpena 2.1 taulan.

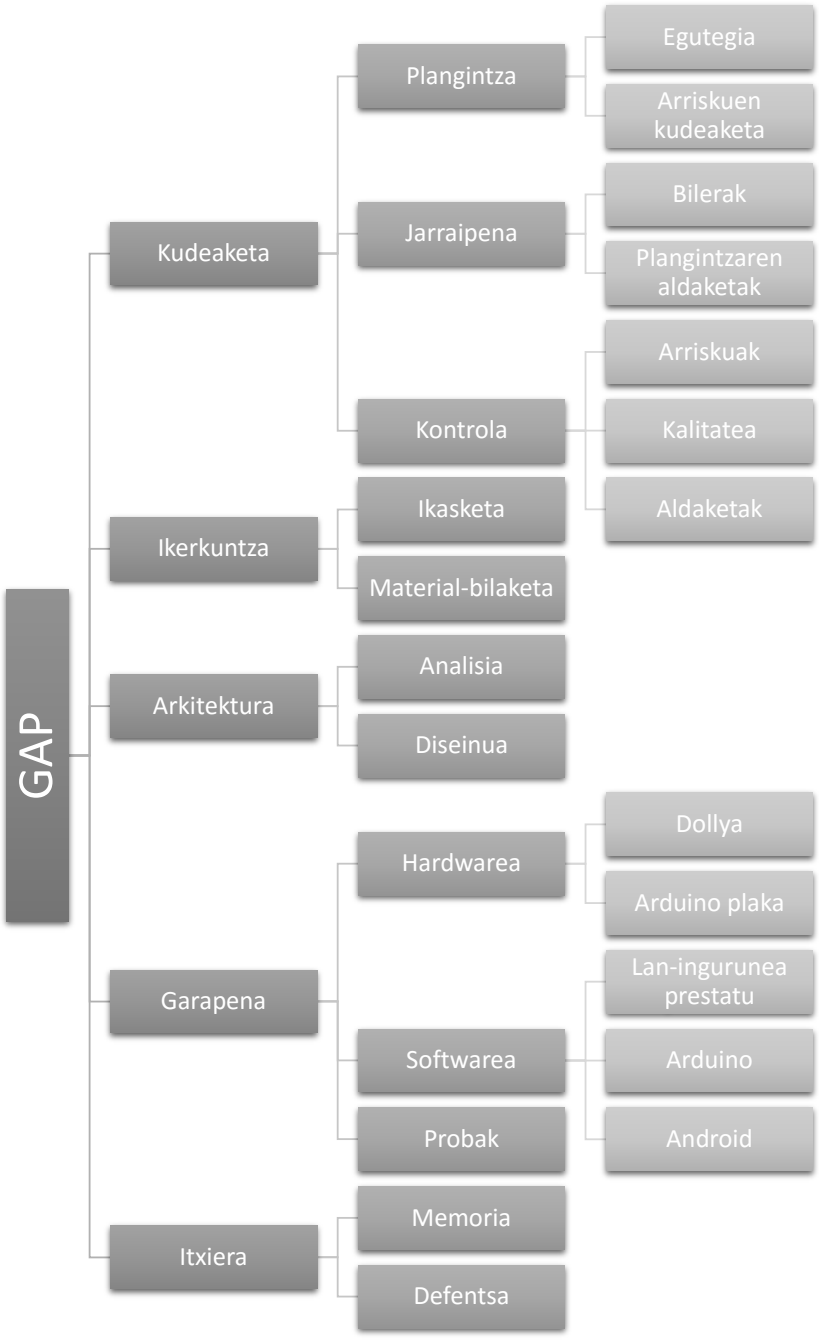
2.3 Lan-metodologia

Proiektuaren garapenean metodologia bat finkatu da hasieratik, denbora-galerak ekidin eta erabilitako denbora ahal bezain emankorrena izateko. Horretaz gain, lan-metodologia honek akatsak ekidin edota identifikatzeko balio behar du, baita plangintzarekiko izandako aldea identifikatzeko ere.

Proiektua bere osotasunean ikusita, zailena lehenengo irismen-maila betetzea izango da. Lehenengo irismen-maila betetzeko honako fase hauek identifikatu dira:

1. Hardware-ingurunea garatu: dollya eraiki.
2. Eragingailuak kontrolatu: motorrak eta sentsoreak Arduinoaren bidez kontrolatu.
3. Komunikazioa: Arduinoaren eta Android aplikazioaren arteko komunikazioa lortu.
4. Funtzionalitateak inplementatu.

Funtzioak garatzen hasten garen unetik aurrera garapen gehiena iterazioka egingo dugu. Funtzio berri edota aldaketa bakoitzak aplikazio eta gailu desberdinekin lan egitea eskatu-



2.1 Irudia: Proiektuaren LDEa

.

Kodea	Ataza	Azalpena
Kudeaketa		
1.1	Plangintza	Proiektuaren plangintza integrala, orduen estimazioak, egutegia, arrisku- eta kalitate-planak, etab.
1.2	Jarraipena	Proiektuaren egoeraren berri emateko noizbehinka zuzendariekin bilerak egin behar dira eta beharrezkoa izango balitz plangintza aldatu.
1.3	Kontrola	Identifikatutako arriskuak kontrolatu eta kalitatea bermatu behar da; egin beharreko aldaketak ere azpiataza honetan definituko dira.
Ikerkuntza		
2.1	Ikasketa	Proiektuan erabiliko diren teknologiak ulertu eta horien inguruan beharrezko informazioa bildu.
2.2	Material-bilaketa	Erabilgarriak izan litezkeen baliabideak bilatu eta gorde; adibidez, kode-zatiak edo liburutegiak.
Arkitektura		
3.1	Analisia	Aztertu sistemak zein osagai beharko dituen.
3.2	Diseinua	Aztertu nola komunikatuko diren osagaiak beren artean.
Garapena		
4.1	Hardwarea	Hardwarearekin lan egiteko ingurunea prestatu, hau da dollya eta Arduinoaren osagaiak muntatu.
4.2	Softwarea	Lan ingurunea prestatu ondoren, softwarearen programazioa burutu behar da: Arduino aplikazioa eta Android aplikazioa.
4.3	Probak	Softwarea garatzearekin batera egindako lana frogatzeko probak egingo dira.
Itxiera		
5.1	Memoria	Proiektuan zehar egindako lana eta izandako gorabeherak biltzen dituen dokumentua idatzi.
5.2	Defentsa	Proiektuaren defentsarako beharrezkoak diren materialak prestatu.

2.1 Taula: LDEaren ataza nagusi eta erdi-mailako atazen azalpena.

ko du; beraz, funtzionalitate bat hasieratik amaieraraino egiten saiatuko gara hurrengoarekin hasi aurretik. Hurrengo irismen-mailak burutzeko ere iterazioak egiten jarraituko dugu, modu horretan sistema pixkanaka osatzen joateko.

Eguneroko lan-metodologia ere definitu da atal honetan; nahiz eta ez guztiz jarraitu metodologia hori beti, ahal den neurrian jarraitu beharko da arriskuak gutxitzeko. Lan-egun bakoitzak honako atal hauek izan beharko lituzke eta ordena honetan jarraitu:

1. Aurreko lan-egunaren balorazio azkarra: aurreko egunean finkatutako helburuak be-

te ziren ikusi, eta proiektuarengan nola eragiten duen hausnartu. Hurrengo pausoa egiterako garaian hutsak baztertzeko balioko du.

2. Eguneko helburuak finkatu: proiektuaren plangintzaren baitan finkatu beharko dira eguneko helburu horiek, baina beti ere malgutasun batekin; motibazioa eta morala altu mantentzeko balio behar dute eguneko helburuek.

3. Gauzatzea: finkatutako helburuak betetzeko egin behar dena egin.

4. Dokumentazioa: egunean izandako eta dokumentatuak izan behar duten oharrak idatzi, proiektuaren memoriari begira batez ere. Hurrengo egunerako oharrak ere idatzita utziko dira badaezpada.

2.4 Emangarri nagusiak

Proiektuak lau emangarri nagusi ditu: Time-lapseak egiteko sistema automatizatua, memoria, kodea eta proiektuaren defentsarako aurkezpena.

- Time-lapseak egiteko sistema automatizatua: Proiektuaren helburua den produktua da. Egindako lana egin dela frogatzeko ezinbesteko emangarria.
- Memoria: Dokumentu hau, Informatika Fakultateko araudiak agintzen duenari jarraiki, ADDI plataformara igo beharko da irailaren 2a baina lehenago.
- Kodea: Garatutako aplikazioen kodea, memoriarekin batera igo beharko da ADDI plataformara.
- Aurkezpena: Proiektuaren defentsan erabiliko den aurkezpena. 2015eko irailaren 16rako prest egon behar du.

2.5 Baliabideak

2.5.1 Software-baliabideak

Proiektua gauzatzeko programazio eta idazketa ugari egin beharko da; ataza bakoitzak software espezifikoa behar izan ohi du eta askotan ez da aukera bakarra izaten. Software

libre eta irekiari lehentasuna eman nahi izan zaio proiektuaren garapenean, beti ere, bere desabantailek ez badituzte bere abantailak guztiz zapaltzen.

Programatzeko erabiliko diren inguruneak bi dira nagusiki: Arduino IDE eta Android Studio. Zorionez, bi inguruneak sistema eragile ohikoenentzat bateragarriak dira, horien artean Linux eta Windows. Programazio-lan gehiena Ubuntu 14.04 sistema eragilea duen mahaigaineko konputagailu batean egingo da; hala ere, probak egiterako garaian, Windows 8 sistema eragilea daraman eramangarri batean egingo dira aplikazioen zuzenketak.

Hala ere programazio ingurunea ez da erabiliko diren aplikazio guztien zati bat besterik. Plangintza egiteko eta jarraipen eta kontrol eguneratua eramateko, Microsoften Office paketea erabiliko da, Excel hain zuzen ere. Memoria idazteko ere software espezializatua behar da, kasu honetan LaTeX motorra erabiltzea erabaki da (Textmaker aplikazioarekin batera), izaera honetako dokumentuek mantendu behar duten itxura eta egitura zorrotza modu egokian mantentzeko.

Horietaz gain, eta proiektuaren emangarriekin zuzenean lotuta ez badaude ere, beste hainbat software-baliabide erabiliko dira; esaterako, Mozilla Firefox informazioa bilatzeko edota OneDrive segurtasun-kopiak egiteko.

2.5.2 Hardware-baliabideak

Hardware-baliabide aldetik, esan beharrik ez dago aurreko ataleko aplikazioak erabiltzeko konputagailua behar-beharrezkoa dela. Xehetasunetan sartu gabe, esan, erabiliko diren konputagailuetako bat mahaigaineko PC bat dela eta bestea PC eramangarri bat.

Horretaz gain, Android gailuak ere behar dira kontrol-aplikazioa instalatu eta probak egiteko. Erabiliko den gailua edozein izan liteke, betiere Bluetooth antena baldin badu eta aplikazioa instalatzeko sistema eragilea eguneratua baldin badago. Garatzeko unean erabiliko den gailua Elephone P6000 mugikorra da.

Gainontzeko hardware-baliabideak Arduinoarekin erlazionatuak daude. Oraindik goizegi da esateko zein diren Arduinoak beharko dituen osagaiak sistema guztia martxan jartzeko, baina, gutxienez Arduino oinarrizko plaka eta Bluetooth komunikazioa sortzeko antena beharko dira. Beharrezkoak izango diren osagai guztiak 4 kapituluan definitzen dira.

2.5.3 Beste baliabideak

Proiektu honek ez ditu gailu informatikoak soilik kontenplatzen, gailu fisiko ez-informatikoak ere eraiki behar dira. Gailua nola eraiki oraindik definitzeke badago ere, dagoeneko badakizkigu zenbait osagai behar izango ditugunak: horien artean, motorra, errailak eta etengailuak, adibidez. Guztia muntatzeko ere erreminta dezente beharko dira.

2.6 Proiektua gauzatzeko plana

2.6.1 Egutegia

Proiektu hau nahiko handia da, orain arte aurre egin diogun proiektu handiena behintzat. Gutxienezko 300 orduko dedikazioa du proiektuak, eta ez gaude ohituta hain epe luzeko estimazioak egitera. Beraz, litekeena da egindako plangintza ez oso zehatza izatea eta benetan eskainitako dedikazioak estimatuetatik asko desbideratzea. Hala ere, ahal den estimazio zehatzena egiten saiatu gara, betiere proiektuaren jarraipen eta kontrol eraginkorragoa egiteko.

Dedikazioaren ordu kopurua banatzerakoan, arreta berezia jarri zaio proiektuaren betebeharren artean definitua dagoen puntu bati. Proiektua Software Ingeniaritza espezialitatearen barnean egin behar da. Proiektuak duen izaera bereziagatik ezinbestekoa da hardware alorrean ere denbora inbertitzea, baita informatikarekin zerikusirik ez duten gauza batzuetan ere, dollya eraikitzen esaterako. Honenbestez, softwarearekin zerikusia duten atalek ia 300 orduak ezarritako dedikazio minimoa betetzen dute eta hardware eta beste osagaien garapenarekin erlazionatutako atazak dedikazio minimo bat besterik ez dute izango.

Dedikazioaren jarraipen eta kontrola errazteko, Excel taula bat eraiki da, modu horretan erraz egin daitezkeelako eguneraketak, eta baita, desbideraketek proiektuaren osotasunari nola eragiten dioten ikusi ere. Estimazio hauek 2.2 taulan ikusi daitezke.

2.6.2 Gantt diagrama

Egutegia egitean eraikitako taularekin batera Gantt diagrama bat egin dugu; era horretan, denboran zehar kokaturik ikus ditzakegu egin beharreko ataza nagusiak, eta baita atazen

Kodea	Atazaren deskribapena	Estimazioa	Hasiera	Amaiera	Iraupena
1	Kudeaketa	33:00	02-02	16-09	226 egun
1.1	Plangintza	14:00	02-02	12-02	10 egun
	Egutegia definitu	12:00	02-02	12-02	10 egun
	Arriskuen kudeaketa plana	2:00	05-02	12-02	7 egun
1.2	Jarraipena	6:00	12-02	10-09	210 egun
	Bilerak	4:00	12-02	10-09	193 egun
	Plangintzaren aldaketak	2:00	12-02	10-09	193 egun
1.3	Kontrola	13:00	02-02	16-09	226 egun
	Arriskuak eguneratu	3:00	02-02	16-09	226 egun
	Kalitatea kontrolatu	5:00	02-02	16-09	226 egun
	Aldaketak kudeatu	5:00	02-02	16-09	226 egun
2	Ikerkuntza	46:00	02-02	29-04	86 egun
2.1	Ikasketa	36:00	02-02	29-04	86 egun
2.2	Material bilaketa	10:00	02-04	29-04	27 egun
3	Arkitektura	35:00	30-04	20-05	20 egun
3.1	Analisia	15:00	30-04	08-05	8 egun
3.2	Diseinua	20:00	11-05	20-05	9 egun
4	Garapena	139:00	02-03	10-08	161 egun
4.1	Hardwarea	30:00	01-03	29-06	120 egun
	Dolly-a muntatu	20:00	01-03	24-06	115 egun
	Arduino eta gainerako osagarriak muntatu	10:00	24-06	29-06	5 egun
4.2	Softwarea	103:00	29-06	31-07	32 egun
	Lan ingurunea prestatu	3:00	29-06	30-06	1 egun
	Arduino aplikazioa programatu	30:00	30-06	31-07	31 egun
	Android aplikazioa programatu	70:00	06-07	31-07	25 egun
4.3	Probak	6:00	22-07	31-07	9 egun
5	Itxiera	90:00	01-08	16-09	46 egun
5.1	Memoria	80:00	01-08	01-09	31 egun
5.2	Defentsa	10:00	02-09	16-09	14 egun
		343:00	**02-02**	**16-09**	**226 egun**

2.2 Taula: Dedikazioaren estimazioa

.

arteko dependentziak ere. Honekin plangintzan gerta litezkeen huts egiteak saihestuko ditugu. 2.2 irudian hasierako estimazioen Gantt diagrama bat ikus daiteke.

2.6.3 Lan-karga

Hasierako estimazioen taulan eta Gantt diagraman ikus daitekeen moduan, proiektuaren iraupena nahiko luzea da: 226 egun. Lan-karga ez da beti berdina izango, ezta antzekoa ere. Proiektua ikasturteko beste ikasgaiekin batera egingo denez, hasierako hilabeteetan nahiko dedikazio-ordu gutxi izango ditugu, eta, ikasgaiak amaitzearekin batera, dedikazioa asko kontzentratuko da azken bi hilabeteetan. Lan-karga txikiena maiatza eta ekaina bitartean egongo da, orduan izango baitira azterketak, eta, horrenbestez, proiektuari denbora gutxi eskainiko zaio.

Ataza batzuei erreparatzen badiegu, eta batzuk aipatzearren, alde batetik hardwarearen garapena daukagu; oso denbora luzean egingo den ataza da hori. Iraupen luze horren arrazoia ataza hau egitean desbideraketa handiak egon daitezkeela aurreikusi dela da, eta, horregatik, malgutasun hori eman diogu, dedikazioa gehitu behar bazaio ere. Atentzioa eman lezakeen beste puntu bat softwarearen garapena da, hilabete bateko epea soilik baitugu eta dedikazio asko; horren arrazoia sinplea da, programazio guztia batera eta segidan egin nahi izan dugu eta irailean dedikazio-ordu asko egin ditzakegu.

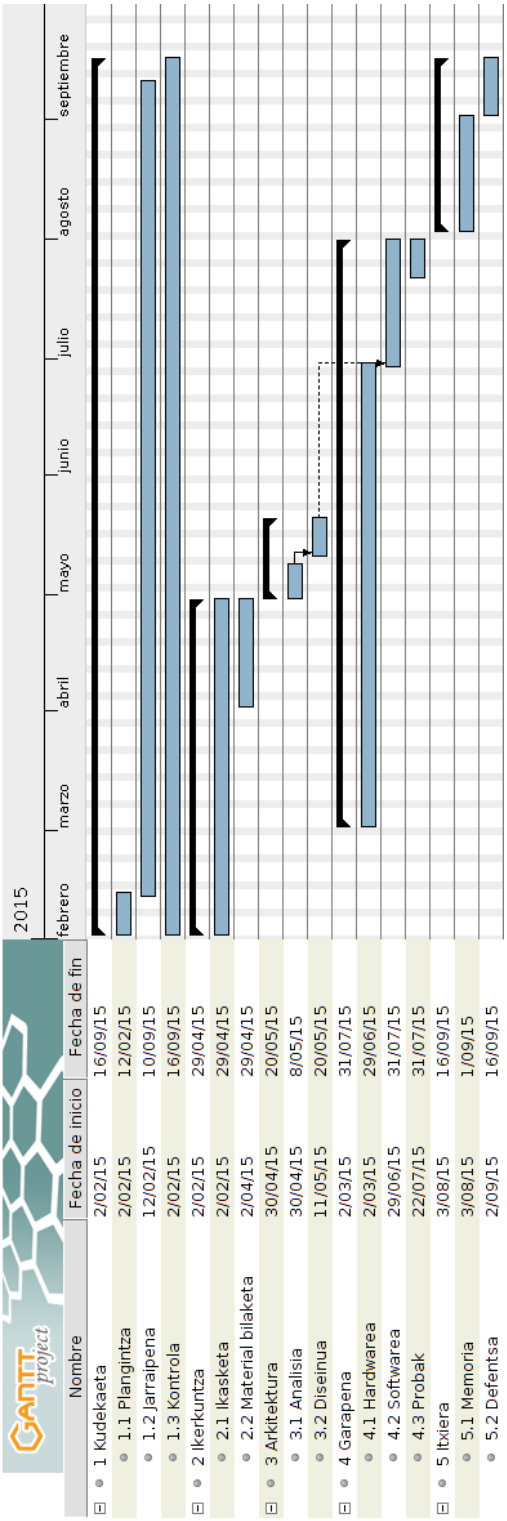
2.7 Kalitate plana

Proiektuak eginbehar asko ditu eta atal horien guztien kalitatea bermatu behar da. Horretarako, atal honetan hiru gai jorratuko ditugu:

- Kalitatearen adierazleak
- Kalitatea ziurtatzea
- Kalitatearen kontrola

2.7.1 Kalitatearen adierazleak

Kalitatearen adierazleak asko izan balitezke ere, garrantzitsuenak kontuan hartzen saiatuko gara; bi multzo nagusi ditugu adierazleen artean: kuantitatiboak eta kualitatiboak.



2.2 Irudia: Hasierako estimazioen Gantt digrama

.

Adierazle kuantitatiboak

- Plangintzaren jarraipena: plangintzan definitu diren datak eta epe-mugak errespetatzea.
- Kodearen lerro-kopurua: kalitate onaren seinale ez bada ere, egunetik egunera kode kopurua hazten joatea, lana aurreratzen doanaren seinale da.

Adierazle kualitatiboak

- Sistemarekiko elkarrekintza: sistema erabiltzerakoan erabiltzaileentzako erabilerraza izan behar du, geroz eta errazagoa izan erabiltzaile eta sistemaren arteko elkarrekintza, produktuaren kalitatea hobea izango baita.
- Sistemaren fidagarritasuna: sistema automatizatua denez, ustekabekoen aurrean erreakzionatzeko duen gaitasuna.
- Kodearen argitasuna: kodea era txukun eta ordenatuan idatzi behar da, beharrezko komentario eta argibideekin.

2.7.2 Kalitatea ziurtatzea eta kontrola

Aurreko atalean definitutako adierazleak betetzen direla edota horiek betetzeko premia dagoela detektatzeko, proiektuaren bizi-zikloan zehar etengabe egin behar dira egiaztapenak. Proiektuak urrats garrantzitsu bat ematen duen aldiro edota denborak finkatutako maiztasun batean, gure kasuan astean behin edota gutxienez bi astean behin egin beharko da adierazle horien konprobaketa.

2.7.3 Kalitatearen kontrola

Proiektuaren arrakastari begira, kalitate-maila minimo batzuk betetzea ezinbestekoa da. Kalitate-maila minimo hori betetzen dela egiaztatzeko egiaztapen-zerrenda bat daukagu, C eranskinean. Kalitate maila minimo horietatik aparte, arrakasta bermatua dagoen heinean, arrakastaren maila neurtzeko ere erabilgarria izango da egiaztapen zerrenda hori.

Kalitatearen kontrol hau proiektuaren amaiera aldera egingo da, baina ez beranduegi.

2.8 Arriskuak

Proiektuaren dimentsio eta izaerarengatik hainbat arrisku topatu ditzakegu. Atal honetan, arrisku nagusiak identifikatu, eta horiek ekiditeko eta konpontzeko zein neurri hartuko diren zehaztuko dira.

- **Datuen galera:** egingo den lanaren zati handiena konputagailuekin eta era digitalean egingo denez, kontuan hartu beharreko zerbait da sistema informatikoek huts egiteko duten joera. Datuak galdu, hondatu edota ezabatu egin daitezke. Horren aurrean, bi dira hartutako neurriak. Lehenengoa lanerako erabiltzen den sistema ahal den neurrian lanerako eta soilik lanerako erabiltzea, modu honetan datuen galera izateko aukerak murriztuko baititugu. Bestea, eta garrantzitsuagoa, segurtasun-kopiak egitea. Segurtasun-kopiak ahal den neurrian egunero egingo dira, eta bi modutan egingo dira gainera: alde batetik, memoria eramangarri batean, eta, bestetik *OneDrive*[1] zerbitzuan.

- **Hardware-ingurunea ezin prestatzea:** proiektuaren arrisku argienetako bat da, dollya ezin muntatu izateak ondorio latzak eragingo lituzke. Honi aurre egiteko ere ez dago kontingentzia plan-argirik; ezingo balitz dollya muntatu proiektuaren irismena birplanteatu beharko litzateke eta beste modu batera egin gauzak. Zorionez, hau gertatzeko aukerak txikiak dira, nahiko ziur baikaude arrisku hau ez dela arazo bat bilakatuko.

- **Aplikazioaren garapenean arazoak:** aplikazioak garatzerako garaian ohiko arazo bat izaten da aplikazioan aldaketaren bat egin eta aplikazioak funtzionatzeari uztea. Hori gertatzean denbora asko galdu daiteke, eta, erabiliko diren teknologiak ezezagunak direnez, oraindik ere gehiago areagotzen da arriskua. Hori ekiditeko *Git*[2] erabiltzea erabaki da, aplikazioaren bertsioen kontrola eramateko eta arazo larriren batekin topatzen bagara, atzera egiteko aukera izateko.

- **Plangintza ez jarraitzea:** plangintza jarraitzea proiektuaren arrakastarako gida da, eta plangintza hori ez jarraitzeak proiektuaren porrota ekar lezake. Horretarako, plangintzan bertan definitu diren prozesuak jarraituko dira, desbideraketak gutxitu eta detektatzeko.

[1] *OneDrive* Microsoften fitxategi-biltegiratze sistema bat da, lainoko biltegiratzean oinarritua.
[2] *Git* kodearen bertsioak kudeatzeko sistema da.

2.9 Parte interesatuak

Proiektuaren interesatu nagusia proiektuaren egilea bera da. Proiektuaren erantzukizuna harena da, eta harena da proiektuaren arrakastarako bidean eman beharreko pauso guztiak ematearen ardura.

Beste maila batean, baina proiektuaren interesatu nagusiak izango dira proiektuaren zuzendariak ere, kasu honetan, Elena Lazkano eta Xabier Artola. Hauen ardura izango da proiektuaren egilea gidatzea eta baita egin beharreko zuzenketak adieraztea ere. Zuzendarien ardura izango da, memoria entregatzen denean, behar den txostena egitea ere.

Hurrengo interesatua proiektua aztertu eta ebaluatuko duen epaimahai taldea da. Gradu amaierako proiektuaren arautegiaren arabera epaimahaia ikasi den espezialitateko irakasleez osatua egongo da, Software Ingeniaritzako irakasleez kasu honetan.

Azkenik, oraindik ezagunak ez diren interesatuak daude. Interesatu horiek proiektuan interesa dutelako edota proiektuaren helburuak betetzeko beren laguntza behar dugulako azalduko dira. Horien artean, familia eta lagunartekoak egongo dira ziur.

2.9.1 Komunikazio-plana

Zuzendariekin komunikazioa aurrez aurre egitea lehenetsiko da, telefonoz edota emailez egitea baino, betiere kasuaren arabera noski. Modu horretan, komunikazio argiagoa lortzen da eta gaizki-ulertuak ere saihestuko dira.

Epaimahaiarekin ez da komunikaziorik izango defentsaren eguna arte, 2015eko irailaren 16a arte, alegia.

Oraindik ezagutzen ez diren interesatuak azaltzean, interesatu bakoitzarekin adostuko dira komunikazio moduak.

3. KAPITULUA

Aurrekarien azterketa

Kapitulu honetan proiektuan parte hartuko duten osagaien informazioa bildu da; horien artean, antzeko proiektuen informazioa, erabiliko diren teknologien inguruko ikerlanak etab. Hemen bildutako informazioarekin, hurrengo kapituluetan definituko den analisi eta diseinua modu eraginkorragoan definitzeko ahalmena izango dugu.

3.1 Antzeko sistemak

Sortu nahi den sistema ez da zerbait guztiz berritzailea, eta, ondorioz, ez gara lehenengoak modu honetako sistema bat garatzen. Horretxegatik, jendeak egindako edota komertzializatzen diren sistemak aztertuko ditugu. Helburu nagusia sistema horien alderdi indartsu eta ahulak identifikatzea izango da, eta aspektu horiek gure sistema diseinatzerakoan erreferentzia moduan erabiltzea.

3.1.1 Etxean egindako sistemak

Sistema hauen abantaila nagusia argia da: merkeak dira. Jendeak ideia apartak izaten ditu bere proiektuak garatzerakoan, eta hori begi bistakoa da Interneten aurkitu daitezkeen antzeko sistemetan. Proiektu gehienek bi alor lantzen dituzte: alde batetik, kameraren eta denboraren kontrola, eta beste aldetik, dolly baten laguntzaz mugimendua sortzea.

Time-lapsea programatzerako garaian, bi dira aukera nagusiak. Errazena, dollyaren mu-

gimendua kontrolatzen duen sistematik aparte beste gailu bat izatea, *interbalometro*[1] deritzona. Gailu hori kamerari konektatzen zaio zuzenean, eta hautatutako aukerei jarraituz time-lapsea sortzeko argazkiak aterako ditu. Kamera-marka bakoitzak (modelo jakin batzuek ere bai) bere konektore berezia izan ohi du, eta, ondorioz, interbalometroa erabiliz kamera jakin bat kontrolatzeko aukera dago soilik, hau da, beste kamera bat erabili nahiko balitz beste interbalometro bat beharko litzateke. Time-lapsea egiteko beste modua kontroladore baten laguntzaz egitea da; dollya kontrolatzen duen kontroladore berak egingo luke, eta aurkitutako kasu guztietan Arduino kontroladore bat erabili da. Modu honetan, interbalometroa erabiltzean egiten den moduan, Arduinoak kameraren konektore berezia erabiltzen du eta hark interbalometroak sortuko lituzkeen seinaleak sortzen ditu argazkiak ateratzeko. Kontroladore bat erabiltzean bai mugimendua eta bai argazkiak ateratzeko prozesua gailu bakarrak kontrolatzen ditu, eta horrela, bien arteko sinkronizazio-arazoak saihesten dira. Horretan datza abantaila nagusia.

Dollya garatzerako garaian desberdintasunak nabarmenagoak dira. Norberak bere modura egin ohi du eta oso kalitate ezberdinetako emaitzak lortzen dituzte. Desberdintasunak eta antzekotasunak azaltzen, zerrenda amaigabe bat osatuko genuke, baina badira emaitza onenak lortzen dituztenen artean identifikatu ditugun zenbait ezaugarri:

- Errailek gurpilek baino emaitza hobeak lortzen dituzte. Gurpilekin lurraren gorabera guztiak nabaritzen dira, eta hori ekiditeko askoz eraginkorragoa da errailak erabiltzea.

- Motorraren indarraren transmisiorako horzdun zintak zehatzagoak dira, eta, ondorioz, mugimendua hobeto kontrolatu daiteke.

- Tamainak garrantzia du. Kontuan hartu behar da dollyak DSLR kamera bat daramala bere gainean, eta kamera hauek pisu handia har dezakete erabiltzen duten objektiboaren baitan. Dolly txikiek, oinarri txikiagoak dituztenez, gehiago bibratzen dute, eta, ondorioz, kasu askotan mugimenduak ez dira behar bezain naturalak.

Aurkitu diren sistema guztien artean ez dugu bakar bat ere topatu urruneko kontrol bat erabiliz kontrolatzen dena. Sistema guztiek pultsadore edo etengailuekin funtzionatzen dute, ez gure sistemak izan behar duen moduan, Android aplikazio bat erabiliz. [Pucci, 2014] [ymkangyt, 2012]

[1] *Interbalometroa:* kamerari atxikitzen zaion gailu bat da, argazki batetik besterako denbora definitu eta argazkia ateratzea automatizatzeko.

3.1.2 Sistema komertzialak

Aurreko atalean azaldutako sistemekin konparatuz sistema hauek askoz emaitza hobeak ematen dituzte, dudarik gabe. Sistema hauek oso garestiak izateaz gain sistema itxiak dira, hau da, oso konplikatua izango litzateke gure aplikazioa sortzea sistema horiek kontrolatzeko. Beraz, ez dugu denbora gehiegi inbertituko sistema hauek aztertzen, baina gutxienez aurreko sistemekin alderatuta zein hobekuntza egin daitezken ikusten saiatuko gara.

Sistema hauek oinarri oso sendoak erabiltzen dituzte lurrean tinko kokatzeko, pisu handia eusten duten tripode bereziak normalean. Egonkortasun handia emateaz gain, altuera aldatzeko aukera ere ematen du tripode-sistemek. Sistema kontrolatzearen ikuspegitik badira Android aplikazio bat erabiliz kontrolatzen diren sistemak. Aplikazioak nahiko oinarrizkoak dira, eta aukera asko eskaintzen baino gehiago, erabilgarritasunean zentratzen dira. [TIMELAPSE, 2010] [Perception, 2015]

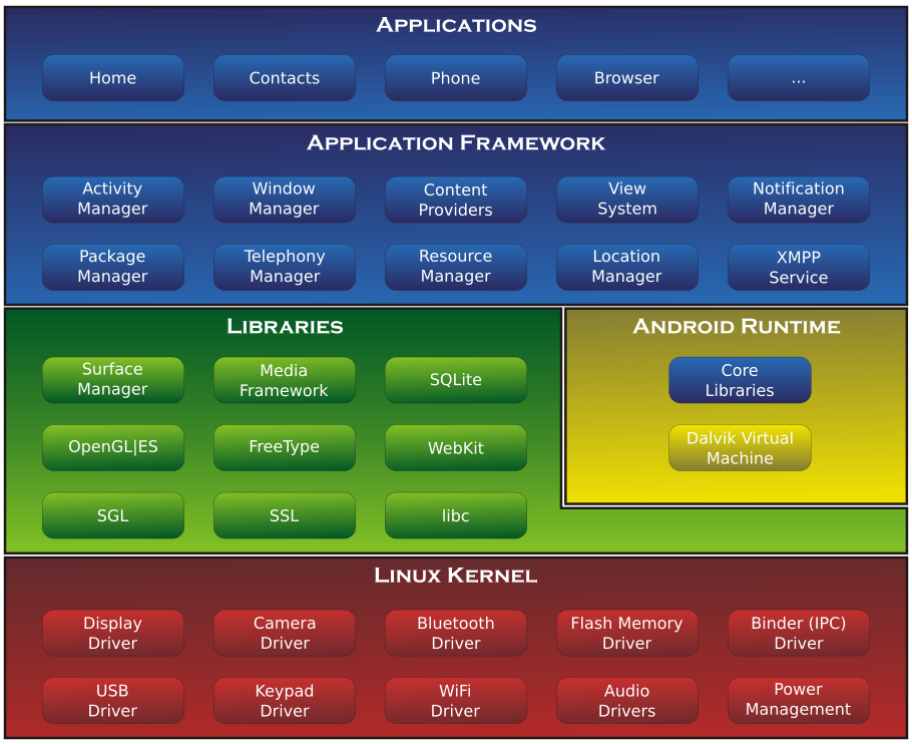
3.2 Teknologiak

Hasiera-hasieratik egindako planteamenduan, definituak ditugu proiektuaren garapenean erabiliko ziren teknologia batzuk. Horiek eta sistema gauzatzeko beharko diren beste teknologia batzuk ere atal honetan aztertzen dira.

3.2.1 Android

Nahiz eta garatu nahi den sistema beste teknologia batekin ere garatzea posible izan, hasieratik hartutako erabakia da aplikazioa Android sistema eragilearentzat garatzea. Horren arrazoi nagusia Android sistema eragilearentzako programatzen ikastea da, oso zabaldua dagoen sistema delako eta aukera asko eskaintzen dituelako. Gainera, 2015ko hasierako datuen arabera, Android sistema eragilea daramaten gailuek mugikor adimentsu guztien %78a osatzen dute [IDC, 2015].

Android sistema eragilearentzat programatzeko lengoaia nagusia Java da, nahiz eta XML bezalako lengoaiek ere garrantzi handia duten. Nahiz eta ez den proiektu honen helburua Androiden barne-egiturari buruzko xehetasunezko azalpenak ematea, sistemaren ikuspegi orokor bat edukitzeko datu batzuk bildu ditugu. 3.1 irudian ikus daiteke sistemaren arkitekturaren diagrama bat.



3.1 Irudia: Android sistema eragilearen arkitektura

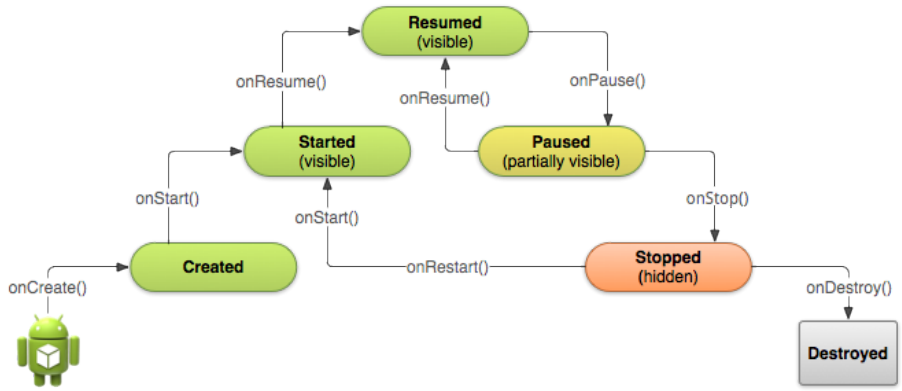
.

Funtsean honako geruza hauek ditu:

- Aplikazioak: Javan idatziak, *Dalvik* izeneko makina birtual batean exekutatzen dira.
- Aplikazioen ingurunea: Android gailuaren oinarrizko funtzioak kudeatzen ditu. Garatzaileen eskura hainbat baliabide jartzen ditu aplikazioak garatzeko.
- Liburutegiak eta Android exekuzio unea: Linux kernelaren eta aplikazio-ingurunearen arteko lotura egiten du. Alde batetik, *Android exekuzio uneak* Dalvik makina birtuala kudeatzen du, Android aplikazio bakoitza horietako makina batean exekutatzen baita. Horretaz gain, aplikazioaren ingurunerako C/C++ liburutegi multzoak eskuragai jartzen ditu.
- Linux kernela: geruza oinarrizkoena, hardwarearen eta bere gainetik dauden geruza guztien arteko lotura egiten du. Horrenbestez, hardware-elementu bakoitzaren kontroladoreak barneratzen ditu, baita energiaren kudeaketa egin ere.

Interesgarria da ikustea aplikazioak zein punturaino dauden babestuak hardwarearekiko. Hori da Androiden programatzearen abantaila nagusietako bat, oso goi-mailako lengoaia izanik, abstrakzio maila ere oso altu batean lan egitea. Androidek definituak ditu lan egiteko osagai eta sistema batzuk; garrantzitsuenak aipatuko ditugu:

- Jarduerak edo *activities* aplikazioaren zati agerikoenak dira. Erabiltzailea jarduera batetik bestera ibili ohi da aplikazioan zehar, eta jarduera bakoitzak aukera desberdinak eskaintzen dizkio. Jarduerak sortzea prozesu nahiko garestia denez, *Activity manager* izeneko jarduera-kudeatzailea dugu hori kontrolatzeko. Jarduera bat zenbait egoeratatik igarotzen da bere bizi-zikloan zehar, eta egoera horietatik igarotzean nahi duguna egiteko programatu dezakegu. 3.2 irudian jarduera baten egoera-aldaketak ikus daitezke.
- Zerbitzuak edo *services* bigarren planoan exekutatzen dira, eta ohikoena pantailan erakutsi behar ez diren eginkizunak egiteko erabiltzea izaten da.
- Eduki-hornitzaileak edo *content providers* aplikazioen artean informazioa trukatzeko interfazeak dira.
- Difusio-hargailuak edo *broadcast receivers*, *observer* patroiari jarraitzen dioten mekanismoak dira. Gertakari baten aurrean dagokion kodea exekutatzeko balio du.



3.2 Irudia: *Activity* baten bizi-zikloa.

Androiderako aplikazioak garatzeko aukerei erreparatuz, aurrerapen handiak egin dira oso denbora gutxian. Gaur egun, *Android Studio* izeneko garapen-ingurunea daukagu. Doakoa da eta Windows, Linux edo Mac OS sistema eragileentzat bateragarria. Garapen-ingurune honek beharrezko liburutegi eta osagaiak ditu, aplikazioa programatu, probatu eta argitaratzeko.

Android bertsioak Android sistema eragile bat denez, baditu bere bertsio ezberdinak eta eguneraketak. Ateratzen den bertsio bakoitzerako funtzionalitate eta baliabide berriak *API* [2] maila batean finkatzen dira, eta maila bakoitzari Android bertsio bat dagokio. Proiektu hau garatzen ari zen mementoan, 22 mailako APIa zegoen eskuragai, Android 5.1 Lollipop bertsioari zegokiona. Aplikazioa garatzen hasterako unean, berebiziko garrantzia du aukeratzen dugun APIak, zenbat eta maila altuagoa erabili, geroz eta baliabide gehiago izango baititugu garatzeko unean, baina geroz eta Android gailu gutxiagotan erabili ahalko da. Android gailu guztiek ez baitute sistema eragilearen azken bertsioa. Gure kasuan, ez da helburua jende guztiarengana iristea, eta, berez nahikoa genuke gure gailuarekin bateragarria izanez gero. Hala ere, ez dugu azken bertsioarentzako garatuko, erdibideko bertsio bat aukeratuko dugu, ustekaberik izanez gero, beste gailu batez ordezkatzeko aukera izateko. Hori kontuan izanik, Android 4.1 bertsio minimoa jartzea erabaki dugu (16 mailako APIa), interfazearen kudeaketan abantailak eskaintzen dituelako eta martxan dauden Android gailu guztien ia %90entzat bateragarria izango delako.

Android eta Bluetooth Androidek lehenengoz 5 mailako APIan jarri zuen Bluetooth euskarria, funtzionalitate nagusi guztiak barneratuz. Hurrengo bertsioetan gauza gutxi

[2] *API* maila bakoitzak garapen-ingurunearen eguneraketa bati egiten dio erreferentzia.

aldatu da 18 mailako APIra arte (Android 4.3). Bertsio horretan guztiz aldatu da Bluetootharen funtzionamendua, eta funtzionalitate berri asko gehitu dira. Hala ere, proiektuaren irismenerako nahikoa ditugu oinarrizko funtzionalitateak, eta funtzio berri horiek ez dira kontuan hartu aplikazioa garatzeko Android bertsioa aukeratzerako garaian.

3.2.2 Arduino

Egileen hitzetan, Arduino hardware libreko plataforma bat da, plaka bat eta mikrokontrolagailu batek osatzen dute, softwarearen garapenerako ingurune batez gain. Prezio baxuak eta garapen-komunitate oso sutsu bat edukitzeak nabarmentzen du Arduino proiektua.

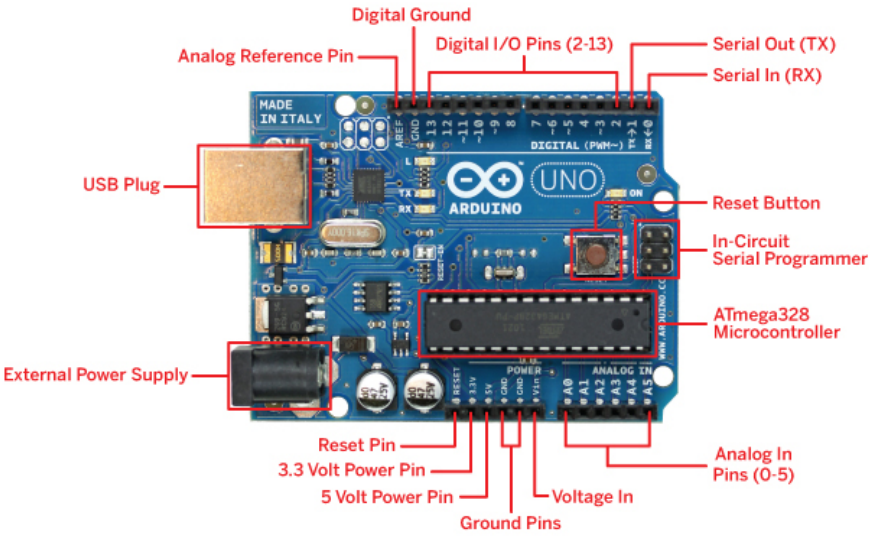
Garapenerako garaian, mikrokontrolagailuak programatzeko, *Wiring* izeneko programazio-lengoaian oinarritzen den Arduino lengoaia erabiltzen da, eta *Processing* proiektuan oinarritutako programazio-ingurune bat dauka horrek. Garapen-ingurunea Arduinoren webgune ofizialetik deskargatu daiteke doan. Behin programatuta, txartelak guztiz autonomoak izan daitezke.

Arduino txartelak erabiliz nahiko erraza da oso konplexuak ez diren sistema elektronikoak sortzea. Sarrera/irteerarako sentsore eta eragingailu asko daude eskuragai, eta baita Arduinorentzat bereziki prestatutako moduluak ere. Bereziki Arduinorentzat prestatutako osagaien artean garrantzi berezia dute *shield* izeneko osagaiek. Shield horiek Arduinoaren gainean jar daitezke bata bestearen gainean dorre bat osatuz eta hainbat aukera ematen dituzte; besteak beste, komunikazio-ahalmenak zabaltzea, GPS lokalizazioa, etab. Baita motorrak kontrolatzea ere. Aurreraxeago aztertuko dugu gure motorrak kontrolatzeko zer erabiliko dugun.

Hainbat Arduino modelo ezberdin badaude ere, hemen aztertutakoa Arduino UNO modeloa da; alde batetik, proiektua hastean eskura genuelako, eta bestetik, gure proiektuaren helburuetarako nahikoa delako. Arduino Uno ATmega328 mikrokontrolagailuan oinarritzen da. 14 sarrera-irteera digital –hauetatik 6 *PWM*[3] irteera bezala erabil daitezke– eta 6 sarrera analogiko ditu. USB konexioa erabiltzen du aplikazioak instalatzeko eta elikadura USB kablearen bitartez edota 7-12V kanpo-elikadura konektorearen bitartez egin daiteke. 3.3 irudian Arduino UNO modeloaren osagaien irudi bat ikus daiteke.

Arduino eta motorrak Arduino plaka batekin motorrak kontrolatzea ez da beste eragingailu batzuk kontrolatzea bezain erraza, motorrek korronte elektriko handiagoa eskatzen

[3] *PWM* Pulse Width Modulation edo pultsu zabalera bidezko modulazioa.



3.3 Irudia: Arduino UNOaren osagaiak.

dute eta Arduino plaka bere horretan ez da nahikoa. Zorionez, motorrak kontrolatzeko hardware berezia dago, Arduinorentzat prestatua: motorrak kontrolatzeko *shield-ak*. Marka eta modelo ezberdinak daude, batzuk Arduino proiektuaren barruan ofizialak direnak eta beste batzuk, berriz, beste enpresa batzuek sortu dituztenak. Gure kasuan, *Arduino Motor Shield* aukeratu dugu, Arduinoren *shield* ofiziala delako eta komunitatearen laguntza jasotzeko errazagoa dela iruditu zaigulako, arazoak izanez gero.

Arduino Motor Shieldak bi DC motor edota *urratsez urratseko motor*[4] bat kontrolatu dezake aldi berean. 5-12 Veko lan-boltajea du, eta 2A (kanal bakoitzeko) eta 4A (bi kanalak batera) tarteko korronte maximoa eman dezake kanpoko elikadura jarriz gero.

Arduino eta Bluetooth Arduino UNO plakek ez dute Bluetooth modulu txertaturik; ondorioz, kanpo-modulu bat gehitu behar zaio ahalmen hori izateko. Kasu honetan ere ez dago modelo bakar bat, eta daudenen artean azterketa bat egin eta aukeratu egin behar izan da. Azkenean, aukeratutako Bluetooth modulua HC-05 modeloa izan da. Modelo hau aukeratzeko arrazoiak bi izan dira: lehenengoa eta garrantzitsuena, gure beharretarako nahikoa izatea. Guk sortu behar dugun komunikaziorako abiadurak onargarriak dira eta egin behar ditugun konfigurazioak egiteko aukerak baditu. Bestalde, eta garapenari begira,

[4] *Urratsez Urratseko motorra* gailu elektromekaniko bat da, pultsu elektrikoak mugimendu angular diskretu bihurtzen dituena.

modelo hau oso erabilia da Arduino munduan, eta, honenbestez, laguntza aurkitzea nahiko erraza izango da.

Modulua erabiltzeko zati konplikatuena hasierako konfigurazioa da; beste gauzen artean, moduluaren izena eta parekatzeko pasahitza ezarri behar dira. Modulua konfiguratzeko, *AT komando moduan* piztu behar da; horretarako, moduluaren KEY hankatxoa HIGH tentsioarekin elikatu behar da, eta ondoren elikadura nagusia jarri. Behin modulua komando moduan dugularik, TX eta RX hankatxoak baliatuko ditugu serie-lerro bidez komandoak bidaltzeko moduluari eta beharrezko aldaketak egiteko konfigurazioan. Konfiguraziorako komandoen laburpen bat 3.1 taulan ikus daiteke.

Modulua konfiguratu ondoren, KEY hankatxoa elikatzeari utziko diogu. Hortik aurrera, elikadura hankatxoak eta TX eta RX hankatxoak soilik erabiliko ditugu komunikazioa gauzatzeko.

Komandoa	Funtzioa
AT	Test UART Connection
AT+RESET	Reset Device
AT+VERSION	Query firmware version
AT+ORGL	Restore settings to Factory Defaults
AT+ADDR	Query Device Bluetooth Address
AT+NAME	Query/Set Device Name
AT+RNAME	Query Remote Bluetooth Device's
AT+ROLE	Query/Set Device Role
AT+CLASS	Query/Set Class of Device CoD
AT+IAC	Query/Set Inquire Access Code
AT+INQM	Query/Set Inquire Access Mode
AT+PSWDAT+PIN	Query/Set Pairing Passkey
AT+UART	Query/Set UART parameter
AT+CMODE	Query/Set Connection Mode
AT+BIND	Query/Set Binding Bluetooth Address
AT+POLAR	Query/Set LED Output Polarity
AT+PIO	Set/Reset a User I/O pin

3.1 Taula: HC-05 Bluetooth moduloaren AT komandoak.

3.2.3 Bluetooth

Bluetooth aski ezaguna den haririk gabeko komunikazio-teknologia da, mundu mailako estandar bat. Bere abantaila nagusien artean, energia-kostu txikia eta gailuen arteko

parekatze-sistema daude. Xehetasun teknikoetan asko sartu gabe, aipatzearren, 100 metro bitarteko Bluetooth konexioak sortu daitezkeela eta 24Mbit/s-ko transferentzia-tasak lortu.

Bilaketa eta parekatze-sistema gauzak asko errazten dituen sistema da. Alde batetik, Bluetooth modulu bakoitza bere izenagatik identifikatzen da; ondoren, modulu horrekin parekatzeko eskaera egin eta pasahitzak elkartrukatzen dira. Aurrerantzean bien arteko konexioa zuzenean egiteko aukera egongo da.

3.3 Eragingailuak

Gure sistemak izango duen eragingailu nagusia motorra da. Motorrak mota eta funtzio askotarikoak daude. Gure kasurako bi mota identifikatu dira egokienak izan daitezkeenak. Alde batetik, erredukzio-kutxa duten motorrak ditugu, eta bestetik, urratsez urratseko motorrak.

Erredukzio-kutxa duten motorrak DC motor sinpleak dira, baina diferentzia batekin: motor elektrikoaren biraketa erredukzio-sistema batekin konektatzen da, abiadura gutxitu eta *torkea*[5] handitzeko. Gure kasuan hori komeni zaigu, mugimenduak motelak baina seguruak izan behar baitute.

Urratsez urratseko motorrek berriz, sistema konplexuagoa dute; xehetasun gehiegitan sartu gabe, pultsuka kontrolatzen dira motor hauek, eta pultsu bakoitzean sortzen den desplazamendua ezaguna da. Motor hauen abantaila nagusia biraketaren desplazamendua neurtu egin daitekeela da, baina bestalde garestiagoak dira eta programatzeko ere dezente konplexuagoak.

Proiektuaren hasiera partean dagoeneko lortu genuen erredukzio-kutxa daukan motor bat, eta motor hori erabiltzea erabaki da programatzeko errazagoa delako eta ez duelako kostu handiagorik suposatuko. Hala ere, arazorik bagenu eta azkenean motor bat erosi beharko balitz, pausokako motor bat erosiko genuke eskaintzen duen zehaztasunagatik.

3.4 Teknika audiobisualak

Proiektu honen helburuetako bat time-lapse sekuentziak lortzea da; ez gara time-lapse horien kalitate edota ikusgarritasunaz kezkatuko, baina beharrezkoak diren ezagutza minimo

[5] *Torkea* ardatz batekiko perpendikularra den indar batek sortzen duen momentua.

batzuk azalduko ditugu time-lapse horiek sortzeko.

Time-lapseak argazki-sekuentziak dira. Argazkiak bata bestearen atzetik ateratzen dira denbora luzean zehar eta ondoren bideo bat bailiran erreproduzitu mugimendu azkarrak lortzeko. Argazki-sekuentzia horiek ateratzerako garaian, aldagai nagusiak bi dira: argazki kopurua eta argazki batetik bestera dagoen denbora-tartea.

- Argazki kopuruak, nolabait, azken emaitzaren iraupena definitzen du. Hau da, geroz eta argazki kopuru handiagoa izan, orduan eta fotograma gehiago izango ditu azken emaitzak.
- Argazki batetik bestera dagoen denbora tarteak time-lapsearen abiadura definitzen du. Argazki batetik bestera geroz eta tarte handiagoa egon, geroz eta azelerazio handiagoa lortuko da azken emaitzan.

Beraz, bi horiek dira gure aplikazioa sortzen dugunean kontuan hartu beharko ditugun parametroak. Argazki kopurua aukeratzea ez da erabiltzailearentzat oso erosoa, kalkuluak egin beharko bailituzke azken emaitzaren iraupena jakiteko. Beraz, diseinua egiterako garaian pentsatu beharko da lan hori nola erraztu. Argazkien tarteko denbora, ordea, berez erabiltzen den balio bat da; segundotan eman ohi da eta ohikoena 1-30 segundo tartea da. Asko jota 60 segundoko tarteak.

4. KAPITULUA

Sistemaren funtzionalitateen analisia

Kapitulu honetan garatuko den produktuaren funtzioen azterketa sakona egiten da, diseinuari ekin baino lehen kontuan hartu beharrekoak zehazteko. Lehendabizi, sistemak izango dituen funtzionalitateen analisia egiten da, produktuaren irismena birdefinituz, eta, ondoren, sistema osatzen duten zatien analisi zehatzagoa.

4.1 Funtzionalitateen analisia

Sistemaren erabiltzaileen profila ezaguna bazaigu ere, analisi egokiagoa egiteko asmoz, argazkilaritza munduko bi profesionalekin bildu gara: alde batetik, Ogeita4 Camera Solutions-eko Xabier Aguado-rekin, eta bestetik, Iban Retegi argazkilariarekin. Ditugun muga ekonomiko eta teknikoak alde batera utzita, gure sistemak zer egitea gustatuko litzaiekeen galdetu zaie. Emandako ideiak interesgarriak izan badira ere, gehienak gure proiektuaren irismenetik asko aldentzen dira, baina ez denak. Ideia horiek sailkatu egin dira: egingo direnak, egin nahi direnak eta etorkizunerako ideia bezala geratu direnak bereizi dira. Hasieratik definitua dugun oinarrizko lerrotik (time-lapsea egin) aurrera egingo diren funtzionalitate-gehikuntzak softwarearen hobekuntzekin egin beharrekoak dira. Ez dugu hardwarearen hobekuntza beharko duten funtzionalitate gehigarririk garatuko.

Oinarrizko lerroa:

- Time-lapsea egin: hasieratik definitua dugun funtzionalitatea.
- Dollya eskuz kontrolatu: sistema guztiz automatikoa izan ordez, dollyaren mugimendua eta kameraren kliskagailua denbora errealean kontrolatu ahal izatea.

Funtzionalitate gehigarriak:

- Time-lapsearen aukerak zabaldu
  - Mugimendua aukerazkoa: nahi izanez gero desplazamendua desaktibatzeko aukera.
  - Time-lapsearen abiadura-aukerak: abiadura aukeratzeko argazki arteko denbora-tartea aukeratu beharrean, azelerazio-faktore bat aukeratu ahal izatea.
- Kamera-modeloa aukeratzeko aukera
- Funtzionamendu manualaren abiadura hautatu
- Desplazamendua mugatu

Irismenetik kanpo:

- Ateratzen ari diren argazkiak Android dispositiboan bistaratu
- Argazkiak Android dispositibora deskargatu
- Aldez aldeko desplazamenduaz gain errotazio-puntu bat
- DSLR kameraren bideoa kontrolatzea
- DSLR kameraren diafragma kontrolatzea

Hemen definitutako funtzioak erabiltzaileak izango dituen aukerak dira, eta, erabiltzaileak Android aplikazioa soilik erabiliko duenez, funtzionalitate hauen analisia kapitulu honetako 4.4.1 atalean egiten da. Irismenetik kanpo dauden funtzionalitateak ez dira hemen landuko, baina horietako batzuk 10.3 atalean aztertuko dira.

4.2 Dollya

Dollyaren funtzioa nahiko erraza da azaltzeko, ez ordea gauzatzeko. Helburu nagusia DSLR kameraren oinarria izatea da. Kamera plataforma higikor baten gainean egongo da finkatuta, eta plataforma hori errail batzuen gainean desplazatuko da alde batetik bestera. Ahal den neurrian plataformaren mugimenduak egonkorra izan behar du, hau da, ez du dardararik izan behar. Kamera eramateaz gain, beharren araberako hardware osagaiak ere dollyan jarriko dira: etengailuak edota motorra esaterako.

4.3 Hardwarea

Hardware aldetik bi osagai nagusi ditugu: bata Android mugikor adimenduna eta bestetik Arduinoa.

Android mugikorrari buruz ez dago gauza asko esateko: mugikorrak aplikazioa instalatzeko gutxienezkoa den Android bertsioa behar du eta Bluetooth ahalmena.

Arduinoari buruz gehiago dago aztertzeko. Arduinoak hardware-osagaien burmuina izan behar du eta gai izan beharko du Android gailuak bidalitako aginduak ulertu eta eginkizunak aurrera eramateko akatsik egin gabe. Kontuan hartu behar ditugu Arduino UNO plakak dituen mugak: 14 sarrera/irteera digital, horietako 6 PWM seinalea sor dezaketenak, eta 6 sarrera analogiko.

Eginkizunak aurrera eramateko, hardware-osagai hauek beharko direla identifikatu dugu:

- Arduino Motor Shield: motorrak kontrolatzeko beharrezko osagaia. Sei S/I digital hankatxo erabiltzen ditu eta bi sarrera analogiko. *Shieldak* definiturik ditu erabiltzen dituen hankatxoak eta ezin dira aldatu. Erabiltzen dituenak hauek dira: 3, 8, 9, 11, 12, 13, A0 eta A1.

- Bluetooth HC-05 modulua: modulu honek bi hankatxo erabiltzen ditu, guk aukeratu ditzakegunak.

Hardware-osagaiek erabiltzen dituzten hankatxoez aparte, kontuan hartzeko beste batzuk ere badira. Honako hauek dira identifikatu diren inguruneko beste osagaiek dituzten beharrak:

- USB: aplikazioen arazketa USB bitartez egiten denez, 0 eta 1 hankatxoak horretarako erreserbatuak daude; beraz, ezin dira erabili, ez hasiera batean behintzat.
- Bide-amaierak[1]: gutxienez 2 hankatxo okupatuko dituzte eta bidearen alde bakoitza detektatzeko erabiliko dira.
- Kameraren kontrola: kameraren kontrolerako seinalea sortzeko beste hankatxo bat erabiliko da.

Laburbilduz, eskura ditugun 14 S/I digitaleko hankatxoetatik 13 erabiliko ditugu. Ez genuke ustekabekorik izan behar, nahiko analisi zehatza egin baitugu; hala ere, hankatxo bat soberan dugu zerbait gehiago gehitu beharra izango bagenu ere.

4.4 Softwarea

Softwarearen ikuspegitik, bi aplikazio ditugu: Android aplikazioa eta Arduino aplikazioa.

4.4.1 Android

Android aplikazioa erabiltzaileak zuzenean erabiliko duen sistemaren zati bakarra da, eta beraz, asko zainduko den atala da honakoa. Alde batetik, aplikazioaren errendimendua ahal den eraginkorrena izaten saiatuko gara, eta, bestalde, erabiltzailearekiko elkarrekintza erraztuko da interfaze sinple eta intuitibo batekin. Aplikazioaren kalitate berme bezala, Android garatzaileen web ofizialean, *Best practices*[2] [Developers, 2015b] ataleko gomendioak jarraituko dira garapen-prozesuan.

Aplikazioaren helburua erabiltzaileak egindako aukerak ondo interpretatu eta arduinoari transmititzea da. Horretarako, aplikazioak eskainiko duen funtzio bakoitza banan-banan aztertu da, funtzio bakoitzak dituen beharrak identifikatuz.

Time-lapsea

Hau da erabilpen nagusia eta gehien zaindu behar den atala ere bai. Funtzionalitate honek honakoa lortu behar du: iraupen jakin bateko eta abiadura finkatu bateko time-lapse bat

[1] *Bide-amaierak* ibilbide baten amaiera detektatzeko etengailuak dira.

[2] *Best practices* Googlek urteroko I/O (garatzaileentzako kongresu bat) bakoitzean publikatzen duen gomendio zerrenda bat da, aplikazio hobeak garatzen laguntzeko.

egiteko informazioa transmititzea. Horretaz gain, kameraren desplazamenduaren noranzkoa ere adierazi behar da.

Hasteko, funtzionalitate honek arduinoari bidali behar dion informazioa identifikatu da: aterako den argazki kopurua, argazki batetik besterako duen itxaron denbora (*frame-tartea* hemendik aurrera), mugimendua egongo den ala ez adieraztea eta mugimenduaren norabidea (eskuinetik ezkerrera edo alderantziz). Beraz, erabiltzaileak aplikazioan egiten dituen aukerekin informazio hori eskuratu behar da, eta erabiltzailearentzat erosoa den modu bat aurkitu hori lortzeko.

- Argazki kopurua: lehenago aipatu bezala, argazki kopurua ez da iraupena kalkulatzeko oso modu erosoa erabiltzailearentzat, aukeratutako argazki kopurua frame-tartearekin alderatu beharko bailuke eta kalkuluak egin. Horren aurrean, erabiltzaileari time-lapsearen kapturaren denbora erreala eskatuko zaio ordu eta minututan, eta denbora hori betetzeko beharrezko argazki kopurua sistemak berak kalkulatuko du.

- Frame-tartea: argazki batetik bestera dagoen itxaron-denbora balioak lortzeko, balio finko batzuen artean emango da aukeratzeko, time-lapse munduan nahiko ohikoak diren denbora-balio batzuk. Denbora-tartea definitu beharrean, azelerazio-faktore bat hautatzeko aukera ere emango zaio erabiltzaileari; kasu honetan, erabiltzaileak aukeratzen duen balioarekin kalkulatuko du sistemak zein den denbora-tartea. Erabiltzaileak bi aukeretako bat hautatu beharko du, eta sistemak egiten dituen kalkuluak guztiz gardenak izango dira harentzat.

- Mugimenduaren norabidea: time-lapseak irauten duen bitartean gertatzen den desplazamendua kontrolatzeko hiru aukera garbi daude: ez mugiu, ezkerretik eskuinera mugitu edota eskuinetik ezkerrera. Erabiltzaileak hiru aukeretako bat hautatu beharko du.

- Mugimendu mugatua: mugimendua bidearen alde batetik besteraino egin beharrean definitutako tarte batean egiteko, erabiltzaileari hasiera- eta amaiera-mugak definitzeko aukera emango zaio.

Azkenik, time-lapsea hasi edota gelditzeko aukera ere eman behar diogu erabiltzaileari.

Eskuzko kontrola

Eskuzko kontrola erabiltzean denbora errealean erreakzionatu behar du sistemak. Berez ez daude aukera asko; alde batera mugitu, bestera mugitu edota argazkia atera. Aukera horietaz gain beste batzuk gehitu dira baliagarriak direlakoan: dollya ertz bateraino eraman edota bestera. Portaera hori eteteko, gelditzeko aukera eman behar da.

Erabiltzailearentzat ahalik eta intuitiboena izate aldera, pantaila ukitu eta mementoan erreakzionatu beharko luke sistemak; 5 kapituluan sakonago aztertuko da aukera egokiena.

Desplazamenduaren abiadura kontrolatzeko aukera ere eskuzko kontrolean sartzen da; honetarako, erabiltzaileari balio batzuen artean hautatzeko aukera emango zaio, eta balio hori mantenduko da eskuzko kontroleko funtzio guztientzat.

Kamera-modeloa aukeratzea

Honen helburu nagusia sistema kamera-modelo bat baino gehiagorentzat baliagarria izatea da. Horretarako, kamera bakoitzak dituen ezaugarriak aztertu behar dira. Kamera-marka eta modelo guztiek ez dute modu berean funtzionatzen, eta beraz, ezinezkoa da guztientzat baliagarria izatea. Horren aurrean, kamera-marka ezagunenen artean aukera emango da, nahiz eta agian modelo jakin batzuek funtzionatu ez.

Hizkuntza aukeratzea

Lokalizazio aukerak ere oso garrantzitsuak dira gaur egun aplikazioak garatzeko unean; nahiz eta gure kasuan ez den behar-beharrezko funtzionalitate bat, gure burua behartu nahi izan dugu aukera hau garatzera, aplikazioaren kalitateari begira balio erantsi handia ematen baitio. Honetarako, aplikazioaren ezarpenen artean hizkuntza aldatzeko aukera emango da, eta erabiltzaileak euskara, gaztelania eta ingelesaren artean aukeratzeko ahalmena izango du.

4.4.2 Arduino

Arduino aplikazioa garatzeko unean ere, aplikazioaren errendimendu optimoa lortzeko saiakera egingo da. Aplikazio honek ez du erabiltzailearekin zuzeneko harremanik izango baina erabiltzailearen aginduak gauzatuko ditu. Hasiera batean, Android aplikazioak esaten dionari itxaron beharko dio aplikazioak, eta, nahiz eta aplikazio honek ez duen bere kabuz erabaki berririk hartu behar, ez da guztiz Android aplikazioaren mendean egongo.

Arduino aplikazioak izango du sistemaren alde fisikoaren berri, une oroko egoera jakingo

du eta bera arduratuko da sistemaren funtzionamendu egokiaz. Gerta liteke, kasu batzuetan, iristen zaion informazioa bete behar ez izatea: adibidez, dollya bidearen ezker aldeko mugan baldin badago eta iritsitako agindua ezkerrerago joatea bada. Kasu horiek kontuan hartu behar dira eta horien aurrean zentzuzko erantzuna eman, eta ez jarraitu itsu-itsuan jasotako informazioa. Zer esanik ez dago horrelako kasuak kontuan hartzen ez badira arazoak egon litezkeela hardwarearekin eta osagaiak hondatu. Honenbestez, Arduino aplikazioa lehenengo eta behin sistemaren integritate fisikoa bermatzeaz arduratuko da, eta ondoren, Android aplikazioak bidalitako aginduak betetzeaz.

Ditugun funtzionalitateak betetzeko Arduino aplikazioak bi zati garrantzitsu ditu: mezua tratatzea, alde betetik, eta jasotako aginduak aurrera eramatea, bestetik. Mezuaren tratamendua berdina da mezu guztietarako, baina mezuaren aurrean egin beharrekoa aldatzen da agindu bakoitzarentzat.

Mezu bakoitzaren prozesamenduak zenbait atal izango ditu, eta azaldutako ordenan jarraituz egin behar dira:

1. Mezua jaso.
2. Jasotako mezua prozesatu eta agindua ulertu.
3. Jasotako agindua uneko egoerarekin alderatu.
4. Aginduari jaramon egin behar zaiola ondorioztatu bada, aginduak bete.

Mezurik jasotzen ez den bitartean edota aplikazioak arazoren bat detektatzen ez duen bitartean bere egoera mantendu egingo da. Mezuan jasotako agindua aurrera eramateko hainbat zeregin desberdin identifikatu dira; horietako zeregin sinple batzuk, zeregin konplexuagoen zati ere badira:

- Eskuinera edo ezkerrera mugitu: motorra norabide batean mugiarazi behar da, adierazitako norabidean eta adierazitako abiaduran.
- Gelditu: motorraren mugimendua eten eta geldirik dagoen egoeran utzi behar da.
- Argazkia atera: kameraren kliskagailuari eragin behar zaio argazkia atera dezan.

- Time-lapsea egin: dudarik gabe, ekintza konplexuena da honakoa. Aurretik azaldu diren funtzio sinpleagoak baliatuko dira time-lapsea sortzeko mugimendu eta ekintzak burutzeko, funtzio sinple horiek automatizatuko baitira. Automatizaziorako beharrezkoa den faktore bat argazki kopurua da, eta bestea egin beharreko desplazamendua. Atera behar den argazki kopuruaren arabera, argazki batetik bestera sortu behar den desplazamendua kalkulatu behar da, eta, horretarako, beharrezkoa izango da bidearen luzeraren balioa jakitea. Balio hori, hasiera batean, aplikazioaren kodean emango dugu, bidea finkoa izango baita. Argazki kopurua eta desplazamendu osoaren balioa jakinda, sistemak tarte bakoitza kalkulatuko du.
  Funtzionamendua honakoa izango da: lehenengo, dollya hasierako posizioan jarriko da; ondoren, ziklo errepikakor batean sartuko da: argazkia atera eta beharrezko desplazamendua sortu eta portaera hori errepikatuko da time-lapsearen argazki guztiak atera arte.

- Kamera-modeloa aldatu: Arduino aplikazioak beharrezkoa izango du definiturik izatea kamera mota bakoitzarentzako ezaugarri konkretuak. Kamera modelo guztiek ez dute seinale berdina erabiltzen kliskagailua aktibatzeko, horregatik da garrantzitsua aldatzeko aukera hori ematea.

Ez da ahaztu behar, mezuak jasotzen edota aginduak betetzen aritzeaz aparte, arduinoak hardwarearen egoera kontrolatzen egon behar duela une oro, eta, ustekabekorik gertatzen bada, aurre egin behar diela.

4.5 Erabilpen kasuak

Egindako analisiaren ondoren, honako hauek dira identifikatu ditugun erabilpen kasuak:

- Time-lapsea egiteko aukerak ezarri: nahi den time-lapsea sortzeko beharrezkoak diren parametroak aukeratzeko ahalmena. Time-lapsearen iraupena, frame-tartea eta mugimenduaren aukerak.

- Time-lapsea hasi: time-lapsea hautatutako aukerekin abian jarri.

- Time-lapsea gelditu: martxan dagoen time-lapsea geldiarazi.

- Dollya eskuz kontrolatu: honetarako zenbait aukera bakan identifikatu dira; dollya alde bateraino mugitu, pixkanaka mugitu edota abiadura aukeratu, adibidez (erabilpen kasu hau osatzen duten funtzionalitate guztiak 4.1 irudian ikusi daitezke).

- Ezarpenak aldatu: aplikazioarekin eta hardwarearekin zerikusia duten ezarpenak aldatzeko aukera. Hizkuntza aldatzeko eta kamera-modeloa aldatzeko aukerak emango dira.

4.1 irudian identifikatutako erabilpen kasuen diagrama daukagu:

4.1 Irudia: Erabilpen kasuen diagrama.

5. KAPITULUA

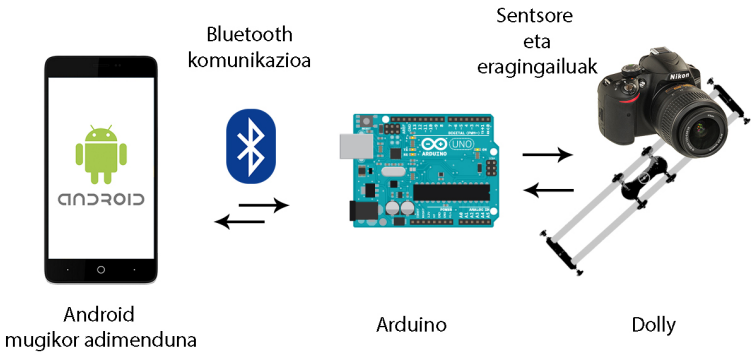
Softwarearen diseinua

Kapitulu honetan egindako sistemaren diseinuaren xehetasunak azaltzen dira, baita sistema osoaren elementu guztien arteko erlazioa ere. Lehenengo, sistema guztiak izango duen diseinuaren azalpen txiki bat emango dugu, argigarri gisa, eta ondoren, egingo ditugun aplikazioen klase eta egiturak aztertuko ditugu.

5.1 Ikuspegi orokorra

Sistemaren osagai bakoitza aztertzen hasi aurretik, sistema osoaren diseinua gainetik aztertuko dugu, elementu bakoitza beste zein elementurekin erlazionatuta dagoen argi ikusteko.

Lehenago esan bezala, gure sistemak aginterako atal bat eta atal eragile bat ditu. Aginteatala Android mugikor adimendun bat izango da eta bertan gure aplikazioa instalatuko dugu. Bluetooth bidez, Android aplikazioa Arduinoarekin komunikatuko da, alde eragilearen arduraduna. Azkenik, Arduinoak sentsore era eragingailu ezberdinak baliatuz, dollya mugiaraziko du eta argazki-kamera kontrolatu. Arduinoak, sentsoreen bidez jasotzen duen informazioa ere Android dispositibora igorriko du aginte-atalari egoera aldaketen berri emateko. 5.1 irudian ikus daiteke sistema osoaren arkitekturaren eskema bat.



5.1 Irudia: Sistemaren arkitekturaren eskema.

5.2 D-Lappse app-a

Garatuko dugun Android aplikazioari D-Lappse izena jarri diogu. Sisteman, paper oso garrantzitsua jokatzen du. Alde batetik Bluetooth komunikazioa kudeatuko duelako: Arduino gailuarekin konektatu eta bien arteko irakurketa/idazketa eragiketak egin beharko ditu. Bestalde, sistemaren eta erabiltzailearen arteko elkarrekintza bideratzearen arduraduna izango da. Beraz, beharrezkoa da aplikazioaren diseinu eraginkor bat diseinatzea.

Aplikazioaren egitura

Aplikazioak dituen klase eta metodoak azaltzen hasi aurretik, erabiltzailearen interfazeak duen egitura orokorra azalduko dugu, elementu bakoitzaren arrazoia hobeto ulertzeko.

Analisiaren atalean aztertutako beharren arabera, *UIak*[1] bi elementu nagusi behar ditu: alde batetik time-lapseak egiteko aukerak emango dituena eta, beste aldetik, eskuzko kontrola egiteko beharrezko elementuak dituena. Atal hauetako bakoitzak pantaila osoa hartuko du; beraz, bi pantaila izango ditugula esan daiteke eta pantaila horien artean nabigatzeko aukera izango dugu. Bi atal horien artean azkar nabigatzea ere komenigarria da. Beraz, fitxa-sisteman (Tab) oinarritutako egitura bat diseinatu da. Atal batetik bestera joateko, behatza pantailaren gainean desplazatuz edota fitxa bakoitzaren gainean sakatuz egin daiteke.

Aplikazioaren helburu nagusia time-lapseak egitea denez, lehenengo Tab-ean jarri dira

[1] *UI* edo *User Interface* erabiltzailearentzat zuzendutako interfaze grafikoa da.

horrekin erlazionatutako elementuak eta bigarrenean kontrol manualarekin erlazionatutakoak. Pantaila bakoitzaren goiko aldea parametroak finkatzeko erabiliko da eta beheko partean, aginduak bidaltzeko botoiak izango ditugu. Bi pantaila horietatik aparte, menu botoia sakatzean, beste aukera batzuk zabalduko dira.

Azaldutako UIko elementuen eta UIko elementu ez direnen portaera kontrolatzeko beharrezkoak diren klase eta metodoak diseinatu dira.

Gure aplikazioak ez duenez datu baserik behar, klase guztiak errepresentazio- eta negozio-logika geruzetan barneratzen dira; bost klase nagusi ditu gure aplikazioak.

- `MainActivity, ManualFragment` eta `TimelapseFragment` klaseek UIaren kontrola eramaten dute.
- `ConnectionMannager` eta `Connection` klaseek, berriz, negozio-logikaren zama handiena daramate.

Arkitektura honek modularizazioa errazten du, eta, modu horretan, aplikazioaren zatiren bat aldatu beharko balitz, ez luke bateragarritasun-arazorik izango. Klase nagusi horietaz gain, badira beste klase batzuk ere. Horien artean, `SettingsActivity` aplikazioaren ezarpenak kudeatzeko erabiliko dena edota `Constants` klasea, klaseen arteko aldagai konstanteak definitzeko.
Javaz idatzitako klaseez aparte, badira beste baliabide batzuk ere, aplikazioaren UIaren layout-a definitzeko erabiltzen direnak edota aipatutako klaseek erreferentziatuko dituzten datuak definitzeko fitxategiak; besteak beste tituluak, testuak edota parametroen balioak. Baliabide horiek XML formatua dute eta oso era errazean definitu daitezke aplikazioak erabiliko dituen datu edota egiturak.

Hurrengo orrietan klase bakoitzaren azalpen zabalagoa egin da. Klase bakoitzak metodo asko ditu, horietako batzuk oso sinpleak eta azaltzea ere merezi ez dutenak, beraz, klase bakoitza azaltzerakoan, metodo garrantzitsuenak soilik azalduko dira. Hala ere, nahi izanez gero, A eranskinean aplikazioaren klase-diagrama osoa aurkitu daiteke, metodo guztiak ikusi ahal izateko.

`MainActivity`

Erabiltzailearen interfazea kudeatzearen arduradun nagusia da. Hala ere, interfazearen elementu oso gutxi kudeatzen ditu zuzenean. Jarduera sortzen denean, `TimelapseFragment` eta `ManualFragment` klaseei deitzen zaie eta klase hauek `Tab` izeneko egitura batean sartzen ditu. Modu horretan, erabiltzaileak interfazearen elementuak modu eroso batean

ikusteko ahalmena izango du. Jarduera honek, beraz, `TimelapseFragment` eta `ManualFragment` klaseek definitzen dituzten UI zatiak kudeatzen ditu.
Horretaz gain, `Fragment`-en eta negozio-logikaren arteko lotura kudeatzen du jarduera honek. UIko elementuek sortzen dituzten eskaerak hartzen ditu eta negozio-logikari igorri. Negozio-logikak sortutako eskaerak eta UIan islatu behar direnak ere hemen kudeatzen dira eta behar den bezala igortzen zaio dagokion `Fragment`-ari.
Kontuan izan behar da, `MainActivity` `TimelapseFragment` eta `ManualFragment` klaseen gurasoa dela eta ondorioz, `Activity` hau itxiko balitz, aplikazioaren UI guztia itxiko litzakeela. Bestalde, klase honetatik era erraz eta eraginkorrean kudeatu daitezke bere ume diren klaseen elementuak. Interfazearen elementuen kudeaketa guztia klase honetatik egin bazitekeen ere, elementu eta metodo asko daudenez, egokiagoa iruditu zaigu klase honetatik ateratzea kode hori era ordenatuago batean egituratzeko.

Hauek dira `MainActivity` klaseak dituen metodo garrantzitsuenak:

- `onCreate`: aplikazioaren muin izango diren aldagaiak hasieratzeaz gain, Bluetooth-a kudeatuko duen `ConnectionManager` haria abiarazten da eta UIa kudeatzeko `SectionPageAdapter`-a ere hasieratzen da.

- `setupHandler`: harien eta UIaren arteko komunikazioa ahalbidetuko duen Handler-a hasieratzen da.

- `onCreateOptionMenu` eta `onOptionItemSelected`: metodo hauek menu botoia sakatzean mezua zabaltzeko portaera definitzen dute eta baita aukera bakoitza sakatzean izan beharreko portaera ere.

- `sendCommand`: Arduinoari bidali behar diren mezuak kudeatzeko metodoa.

`TimelapseFragment`

UIaren time-lapsearen atala kontrolatzen du klase honek. Erabiltzaileak time-lapsea egiteko parametroak definituko ditu eta parametro horien balioak eskuratu eta tratatzea izango da klase honen helburu nagusia. Parametroak eskuratutakoan, laburpen bat eskainiko zaio UIan erabiltzaileari time-lapseak izango dituen ezaugarriak erakusteko. Ezaugarri horiekin ados dagoenean, erabiltzaileak aginduak bidali behar ditu; horretarako, klase hau `MainActivity` klasearekin komunikatuko da behar den mezua transmititzeko. Honako hauek dira UIan erakutsiko diren elementuak:

- Time-lapsearen iraupena: erabiltzaileak alde batetik orduak eta beste aldetik minutuak aukeratu ahalko ditu *number picker* izeneko widget batzuetan. Behatza zenbakien gainean mugituz, balioak aukeratu ahalko dira. Erabiltzailearentzat intuitiboa izate aldera, minutuen balioa bi digiturekin adieraziko da beti (adib. 02 minutu 2 minutu izan beharrean).

- Frame-tartea: frame tartea aukeratzeko, zuzenean denbora balio bat aukeratu daiteke definitutako balio batzuen artean, edota bestela azelerazio balio bat aukeratu daiteke, hemen ere aurrezarritako balioen artean aukeratuz. Bata edo bestea aukeratzeko bi *radio button* erabiliko dira eta bata aukeratzean bestea desaktibatuko da. Balioa aukeratzeko, *spinner*-ak erabiliko dira, balio finko batzuen artean erabiltzaileak bere aukera egiteko. *Radio button* bat edo bestea aukeratzean, horietako bakoitzari dagokion *spinner-a* aktibatuko da eta bestea desaktibatuko da; modu horretan erabiltzailea ez da okertuko.

- Mugimendua: dollyak izango duen mugimendua definitzeko *checkbox* bat eta bi *radio button* erabili dira. Erabiltzaileak garbi ikusteko gertatuko dena, lehenengo mugimendua aktibatuko duen *checkboxa* aukeratu egin beharko du eta, ondoren, norabidea adierazten duten bi *radio button-ak* aktibatuko dira.

- Laburpena: time-lapsearen parametroak definitu ditugunean, parametro horien baitan sortuko litzatekeen time-lapsearen ezaugarriak adierazten ditu.

- Hasi/Gelditu botoia: time-lapsea hasi aurretik, botoiak "Hasi"testua izango du, eta hau sakatzean hasiko da time-lapsea. Time-lapsea martxan dagoen bitartean ordea, botoiaren testua aldatu egingo da eta "Stop"mezua erakutsiko du, time-lapsea gelditu nahi bada ere gelditzeko aukera izateko.

- Ezkerrera eta eskuinera botoiak: dollya bidearen ertz bateraino edo besteraino bidaltzeko botoiak.

`ManualFragment`

UIaren kontrol manualaren atala kontrolatzen du klase honek. Funtzionamendu orokorra dagoeneko azaldu dugun `TimelapseFragment`-en antzekoa da. Alde batetik, erabiltzailearen ekintzak tratatu, eta bestetik, beharrezko informazioa `MainActivity` klaseari bidali behar dio. Honako elementu hauek ditu interfazeak:

- Abiadura aukeratu: abiadura aukeratzeko, *SeekBar* bat erabili da, bertan 0tik 10erako balio bat aukeratu ahalko du erabiltzaileak.

- Ezkerrera eta eskuinera botoiak: erabiltzaileak botoi hauetako bat sakatzean, dollya alde batera edo bestera mugitzen hasiko da eta mugitzen jarraituko du botoia sakatua mantendu bitartean. Askatzean gelditu egingo da.
- Dena ezkerrera eta dena eskuinera botoiak: aurrekoaren antzeko funtzionamendua da, baina botoia askatzean dollya gelditu beharrean mugitzen jarraituko du bidearen ertzera iritsi arte.
- Gelditu botoia: botoi hau sakatzean, edozein dela ere momentuko egoera dollya gelditu egingo da.
- Argazkia atera: argazkia ateratzeko botoia sakatzearekin aski da.

`ConnectionManager`

Klase hau Bluetooth-aren kudeaketarekin zerikusia duen guztiaz arduratzen da. Asko dira klase honen zereginak baina guztiak klase bakar batean biltzea beharrezkoa iruditu zaigu diseinu egoki bat lortzeko.

Lehenengo eta behin, dispositiboaren Bluetooth gailua egiaztatzen da, gailua detektatzen den eta aktibatua dagoen ikusteko. Ez baldin badago aktibatua, aktibatzeko eskaera egingo da. Ondoren, Arduino gailuaren bilaketa hasiko da, aurkitzen denean, ea parekatuta dagoen ala ez egiaztatzen da. Parekatuta ez balego parekatzeko eskaera egiten da eta dagoeneko parekatuta egonez gero, konektatzeko saiakera egingo litzateke.

Honako hauek dira klase honen metodo garrantzitsuenak:

- `configureReceievers`: metodo honetan gure *receiver*-ak iragaztea nahi ditugun gertakizunak aukeratzen ditugu.
- `onReceive`: *receiver-ak* jasotzen dituen gertakarietako baten aurrean behar den ekintza egin; dispositibo bat parekatu izana edota konexioa galdu dela detektatu daiteke.
- `createSocket`: Arduino dispositiboa aurkitzen denean, *socket* bat irekitzen da komunikaziorako.
- `runSocket`: sortutako *socketa* ireki eta konexio berri bat sortu.

`Connection`

`ConnectionManager` klaseak sortzen duen klasea da honakoa. *Socket* ireki bat pasatzen zaio eta hari berri batean exekutatzen da `Connection`-a.

Hauek dira klase honen metodo garrantzitsuenak:

- `run`: Mezuak jasotzeko metodoa, amaigabeko begizta batean mantenduko da arazorik gertatu ezean.
- `write`: Mezuak bidaltzeko metodoa, bidali nahi den mezua *socketaren* bitartez transmititzen du.

Mezuak idazteko garaian ez dago arazorik, mezu bat hartu eta *socketaren* bitartez bidaltzea besterik ez da. Baina mezu bat jasotzeko, *socketa* irekita mantendu behar da eta denbora guztian entzuteko moduan. *Socketa* irekita mantenduko bagenu denbora guztian, ezingo genuke mezurik bidali, entzute-egoeran blokeatuta edukiko genuke eta. Horretarako, bi `Connection` desberdin ditugu abian hari desberdinetan; bata, mezuak idazteko erabiliko dugu eta, bestea, berriz, mezuak jasotzeko.

`Settings Activity`

`SettingsActivity` klasea Android Studio garapen inguruneak automatikoki sortzen duen klase bat da. Klase honen helburua aplikazioaren barne konfigurazioko beharrezko aukerak eskaintzea da. Klase honetan erabiltzen diren aldagai gehienak XML fitxategietan gordetzen dira eta ezarpen horiek mantentzeko, Android-ek eskaintzen duen barne funtzio batzuk erabiltzen dira. Ezarpenen jarduera hau, menu botoia sakatu eta erakusten diren aukeren artean, ezarpenak aukera sakatzean exekutatzen da. Pantaila guztia betetzen du ezarpenen jarduerak eta zerrenda bat erakusten du aukera ezberdinekin. Honako aukera hauek eskaintzen ditu `SettingsActivity`-k:

- Kamera aukeratu: Kamera aukeratzeko, aukeran dauden kamera-modeloen zerrenda bat erakutsiko da eta horietako bat aukeratu beharko da.
- Dollyaren izena: Bluetooth moduluak izen bat dauka identifikatzeko, Android gailuak izen hori erabiliko du bilaketa egin eta gailua parekatzeko. Edozein arrazoi dela eta, izen hori aldatu egin beharko balitz, hemendik egingo litzateke.
- Fotogramak segundoko: Azken emaitza zein frame-maiztasunekin egingo den adierazi daiteke. Hiru aukera izango dira eta bat aukeratuko du erabiltzaileak: 24, 25 eta 30.

- Bidearen luzera: Erabiltzaileak, dollyaren bidearen luzera sartuko du zenbakizko balioetan, zentimetrotan.
- Hizkuntza hautatu: Aukeran dauden hizkuntzen zerrenda irekiko da eta horien artean bat aukeratuko da.

5.3 Sekuentzia-diagramak

Android aplikazioaren diseinua eta klaseen artean dagoen informazio-fluxua argi ikusteko, sekuentzia-diagramak osatu ditugu; ez da erabilpen-kasu bakoitzarentzako sekuentzia diagrama bat egin, erabilpen-kasu batzuk besteen oso antzekoak baitira. Diagrama hauek programatzerako garaian erabiliko ditugun gidalerroak dira, eta, honenbestez, esanguratsuenak iruditu zaizkigun sekuentzia-diagramak egin ditugu. Egindako diagramekin, aplikazioak beharko dituen informazio-fluxu egitura guztiak azaltzen dira, eta, azaltzen ez diren kasuak, hauei aldaketa txikiak aplikatuz definituko lirateke.

Time-lapsea hasi

Time-lapsea kudeatzeko pantailan gaudela eta konexioa ezarrita dugula egin daiteke ekintza hau, ez bestela. Time-lapsea abian baldin badago ere ezingo genuke berriro hasi, lehenengo abian dagoena gelditu beharko genuke berri bat hasteko. Erabiltzaileak interfazeko *Hasi* botoia sakatzen duenean hasten da exekuzio hau; lehenengo `TimelapseFragment`-ak detektatzen du botoia sakatu egin dela eta *fragment* honek beharrezko parametroak jaso eta `MainActivity`-ri deitzen dio. Ondoren, `MainActivity`-k egiaztapenak egiten ditu: egiaztapenen batean time-lapsea egiteko baldintzak betetzen ez direla detektatzen bada, erabiltzaileari behar duen abisua erakusten zaio interfazean, bestela, `Connection`-aren laguntzaz agindua bidaltzen da. 5.2 irudian ikus daiteke sekuentzia-diagrama.

Time-lapsea gelditu

Time-lapsea gelditzeko, noski lehenago time-lapsea hasi egin behar izan da; horretaz gain, agindu hau bidaltzeko Bluetooth konexioa ere beharrezkoa da. Agindu hau time-lapsearen pantailatik edota kontrol manualaren pantailatik bidali daiteke, bi kasuetan *Gelditu* botoia baitago (diagrama egiterakoan `TimelapseFragment` aukeratu dugu). Normalean, erabiltzailearen papera ahalik eta sinpleen mantendu dugu, baina, kasu honetan, beharrezkoa iruditu zaigu erabiltzaileari konfirmazio bat eskatzea agindua bidali aurretik, modu horretan, time-lapsea ez geratzeko ustekabean. Erabiltzaileak konfirmazio-leihoan ez gelditze-

ko erantzuten badu bere horretan gelditzen da exekuzioa, baiezkoa erantzuten badu ordea exekuzioa gelditzeko agindua bidaltzen da. 5.3 irudian ikus daiteke sekuentzia diagrama.

Ezkerrera joan

Funtzionalitate hau eskuinera joan edota argazkia ateratzearen oso antzekoa da. Agindua interfazean ematen da eta ondoren agindu hori bidaltzea besterik ez da, egiaztapen gehiagorik gabe. Noski, agindua bidali ahal izateko, Bluetooth konexioak ezarrita behar du eta time-lapserik ezin da egon abian. 5.4 irudian ikus daiteke sekuentzia-diagrama.

Hizkuntza aukeratu

Edozein mementotan egin dezakegu hizkuntza aldaketa, ez da ez konexiorik edota beste baldintzarik bete behar hizkuntza aldatu ahal izateko. Lehenengo eta behin, erabiltzaileak menua ireki eta ezarpenen aukera sakatu behar du, ezarpenen pantailara sartzeko. Hori kudeatzeaz `MainActivity` arduratzen da eta ezarpenak sakatzean `SettingsActivity` klasea sortzen du. Ondoren, ezarpenen pantailan, hizkuntzaren gainean sakatuko du erabiltzaileak eta aukeran dauden hizkuntzak erakutsiko zaizkio. Azkenik, erabiltzaileak hizkuntza bat aukeratzen duenean, aplikazioaren hizkuntza aldatuko da eta pantailak berriro kargatuko dira hizkuntza egokia erakusteko. 5.5 irudian ikus daiteke sekuentzia-diagrama.

Konexioa ezarri

Aplikazioaren funtsezko eginkizuna da honakoa, gainontzeko funtzionalitate ia guztiak honen mende baitaude. Gainera, bi modu daude konexioa ezartzeko: ohikoena, erabiltzailearen parte hartzerik gabe, eta bestea, erabiltzaileak zuzenean aginduta. Bien arteko diferentzia bakarra konexioa egiteko eskaera non egiten den da: `MainActivity`-k `ConnectionManager` sortzen duen unean, Bluetooth bilaketa hasten da eta dollya aurkitzean automatikoki egiten da konexio eskaera. Konexio horrek huts egiten badu ordea, erabiltzaileak eskuz hasi behar du berriro ere bilaketa hori, horretarako, menuko *Konektatu* botoia sakatu behar du. 5.6 diagraman, lehenengo aukera azaltzen da; bertan, aplikazioa exekutatzen den unean sortzen den `MainActivity`-rekin hasten da fluxua. `MainActivity`-k `ConnectionManager` sortzen du eta honek `BroadcastReceiver` objektu bat instantziatu. `BroadcastReceiver` horrek Bluetootharen egiaztapen guztiak egiten ditu, eta azkenean, gailua aurkitzen badu, `Connection` klaseari deitzen dio konexio bat ezartzeko. 5.6 irudian ikus daiteke sekuentzia-diagrama.

5.4 Dollyarentzako driverra

Dollyaren driverra Arduinoak erabiltzen duen softwarea da eta Androidena baina askoz sinpleagoa da diseinu aldetik. Android aplikazio guztiek bi osagai nagusi izaten dituzte: Setup eta Loop.

`Setup` funtzioan aplikazioak erabili behar dituen Arduinoaren S/I guztiak konfiguratzen dira; baita erabiliko diren aldagaiak hasieratu ere. Funtzio hau Arduinoak bere aplikazioa hasten duen momentuan exekutatzen da eta ez da berriro exekutatzen programa berrabiarazten ez bada.

`Loop` funtzioa etengabe exekutatzen ari den funtzio bat da. Funtzioak dituen agindu guztiak exekutatu ondoren, berriz hasieratik exekutatzen hasten da eta bertan, aplikazioaren funtzionamendua kontrolatzeko elementu guztiak egon behar dute. Hala ere, funtzio honetatik kanpo ere guk behar ditugun metodoak definituko ditugu. Modu horretan, loop-etik dei egin ahalko diegu metodo laguntzaile horiei. Argi dago, funtzionamendu konplexu bat lortzeko, egoera ezberdinak izango dituela gure aplikazioak; beraz, egoera horiek egiaztatu eta aldatzeko mekanismoak sortu beharko ditugu. Egoerak aldatzeko arrazoiak bi izan daitezke: alde batetik, Bluetooth bidez jasotzen den mezu batek aginduta alda dezake eta beste aldetik, Arduinora konektatutako sentsoreren batek detektatutako informazioak aldatu lezake.

5.7 irudian Arduino aplikazioaren exekuzioaren diagrama bat ikus daiteke, egoerak zein unetan egiten diren ikusteko. Eskeman ikus daitekeen moduan, Bluetooth mezu bat jasotzea ez da nahikoa zuzenean egoera aldatzeko; lehenengo, jasotako informazioa Arduinoaren sentsoreen informazioarekin alderatu behar da, eta ondoren egoera aldatu behar den aztertu.

5.5 Komunikazio-protokoloa

Gure sistemaren bi aplikazioen artean komunikazioa Bluetooth bidez ezartzen da. Baina, komunikazio horretan zehar, aplikazio baten mezuek beste aplikazioan behar duten erantzuna izateko, beste komunikazio-protokolo bat ezarri beharra daukagu.

Bluetooth konexioa kudeatzearen arduradun nagusia Android sistema eragilea izateak segurtasun geruza nahiko sendoa ematen digu. Horrenbestez, gure aplikazioen arteko protokoloa ahalik eta sinpleen mantendu dugu alferrikako datuak ez bidaltzearren. Mezu

gehienak oso tamaina txikikoak izango dira, agindu bat eta kasu batzuetan agindu hori betetzeko beharrezkoa den informazio osagarria bidali beharko da. Komando eta datu osagarriak, beraz, mezua osatuko dute eta ez dute beti tamaina bera izango. Mezuan doan informazio osagarria, komandoaren araberakoa izango da; hau da, komando bakoitzak informazio gehigarri jakin bat eramango du. Hori ikusirik, mezuaren tratamendua errazteko, mezuaren hasieran bidaliko den lehenengo gauza komandoa bera izango da eta komando guztiek tamaina bera izango dute. Segidan, beharrezko datu gehigarriak gehituko dira.

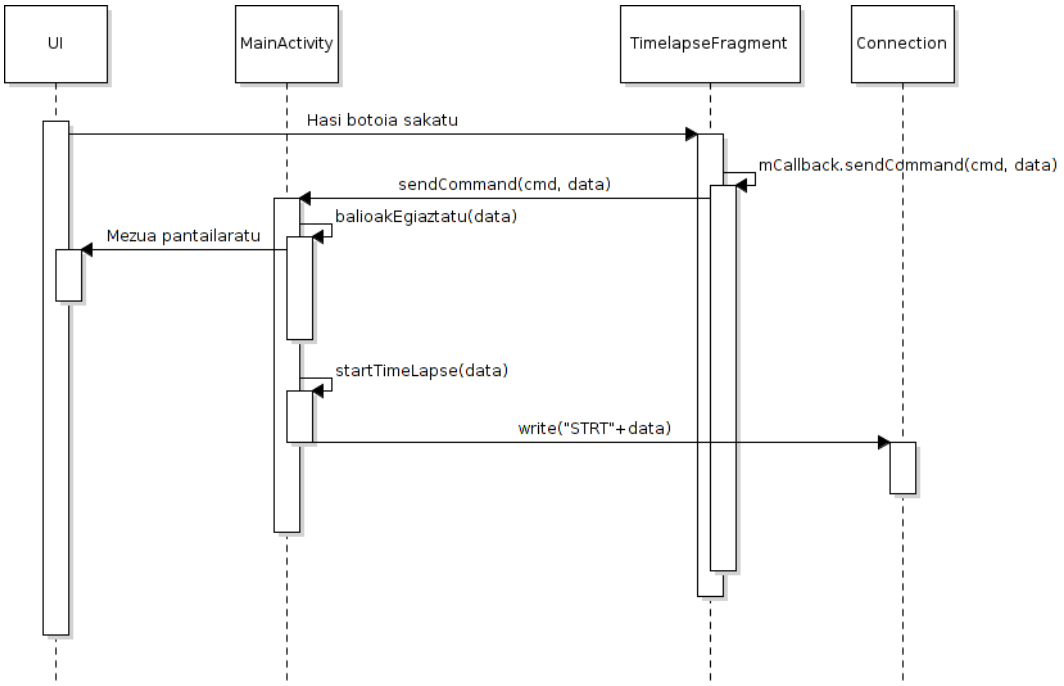
Jarraian, Android eta Arduino aplikazioen artean bidaltzen diren mezuak zerrendatuta ageri dira. Dokumentu honetan azalpen sinplea ematen da, baina komunikazio-protokoloaren xehetasun guztiak jakiteko B eranskinean begiratu daiteke.

Android aplikazioak bidalitakoak:

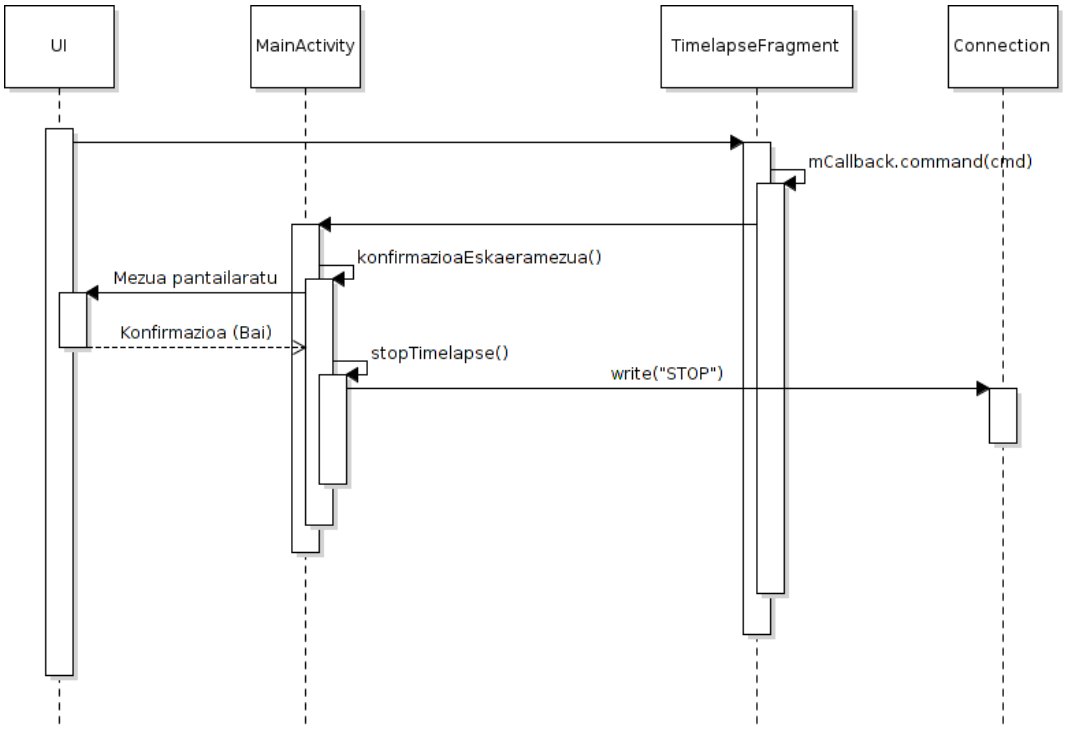
- **STRT**, time-lapsea hasteko agindua. 3 parametro bidaltzen ditu: argazki kopurua, frame-tartea eta norabidea.
- **RIGH**, eskuinera joateko agindua. Parametro bat bidaltzen du: mugimendu-abiadura.
- **LEFT**, ezkerrera joateko agindua. Parametro bat bidaltzen du: mugimendu-abiadura.
- **STOP**, gelditzeko agindua. Ez du parametrorik bidaltzen.
- **SHOT**, argazkia ateratzeko agindua. Ez du parametrorik bidaltzen.
- **CONF**, konfigurazioa bidaltzeko agindua. 2 parametro bidaltzen ditu: kamera modeloa eta bidearen luzera.

Arduino aplikazioak bidalitakoak:

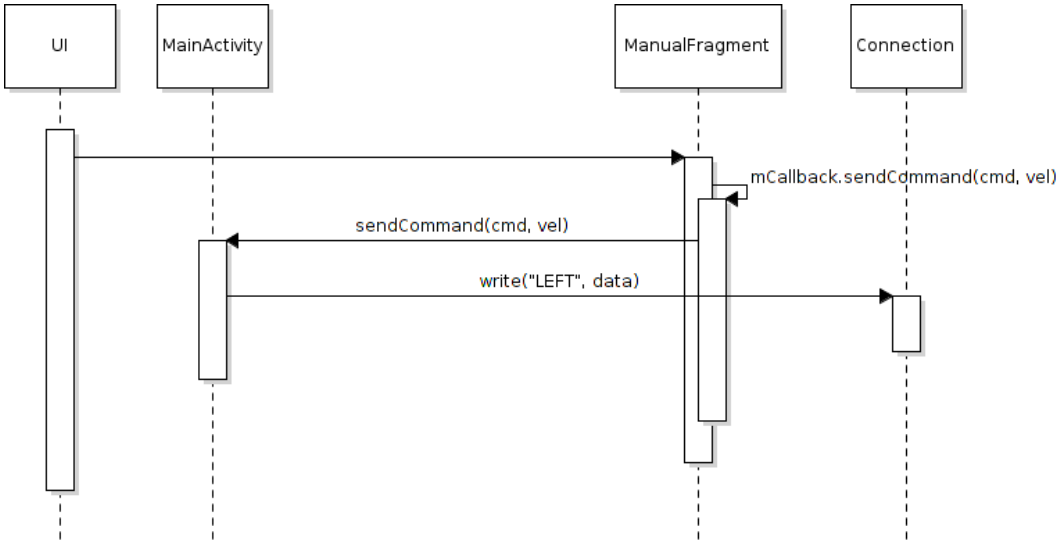
- **CONF**, konfigurazioa ondo ezarri den edo ez abisatzeko. Parametro bakarra darama: ondo ezarri den ala ez.
- **DEND**, dollya bidearen ertz batera iritsi denean bidaltzen da. Parametro bakarra darama: zein aldetara iritsi den dollya.
- **FAIL**, time-lapseak huts egin duela adierazten du. Ez du parametrorik bidaltzen.



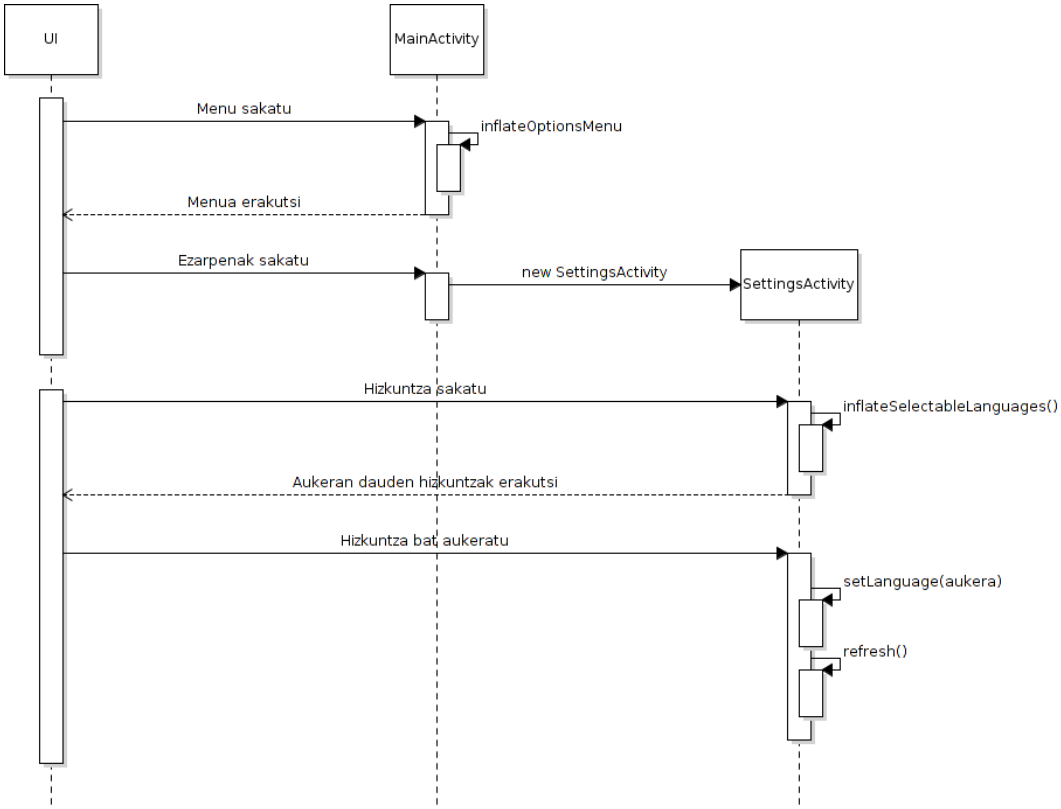
5.2 Irudia: *Time-lapsea hasi* sekuentzia diagrama.



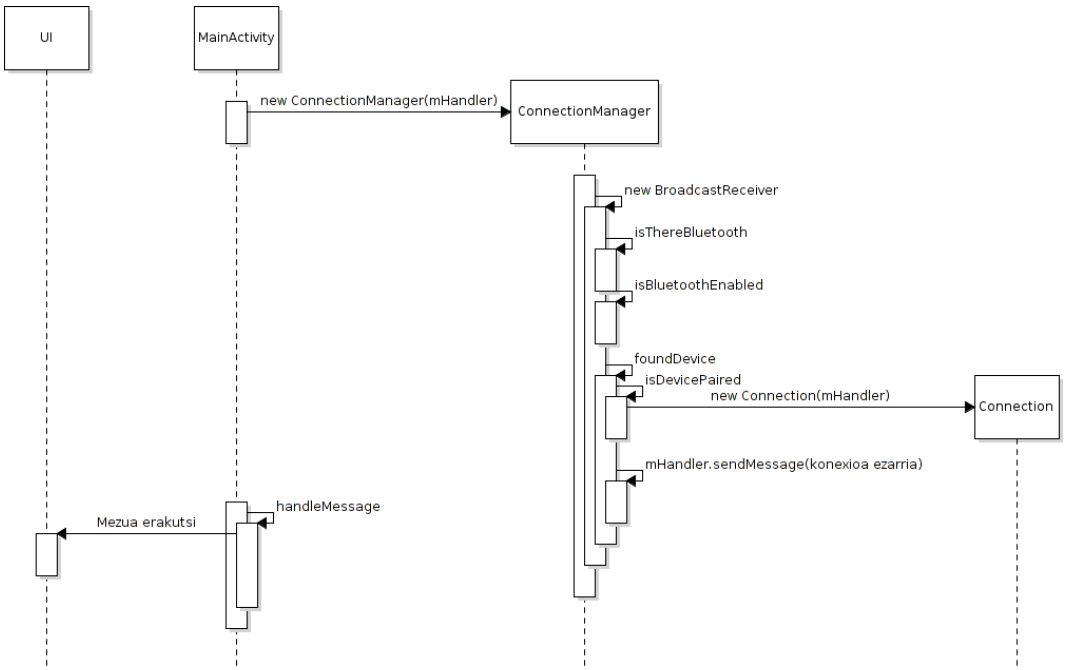
5.3 Irudia: *Time-lapsea gelditu* sekuentzia diagrama.



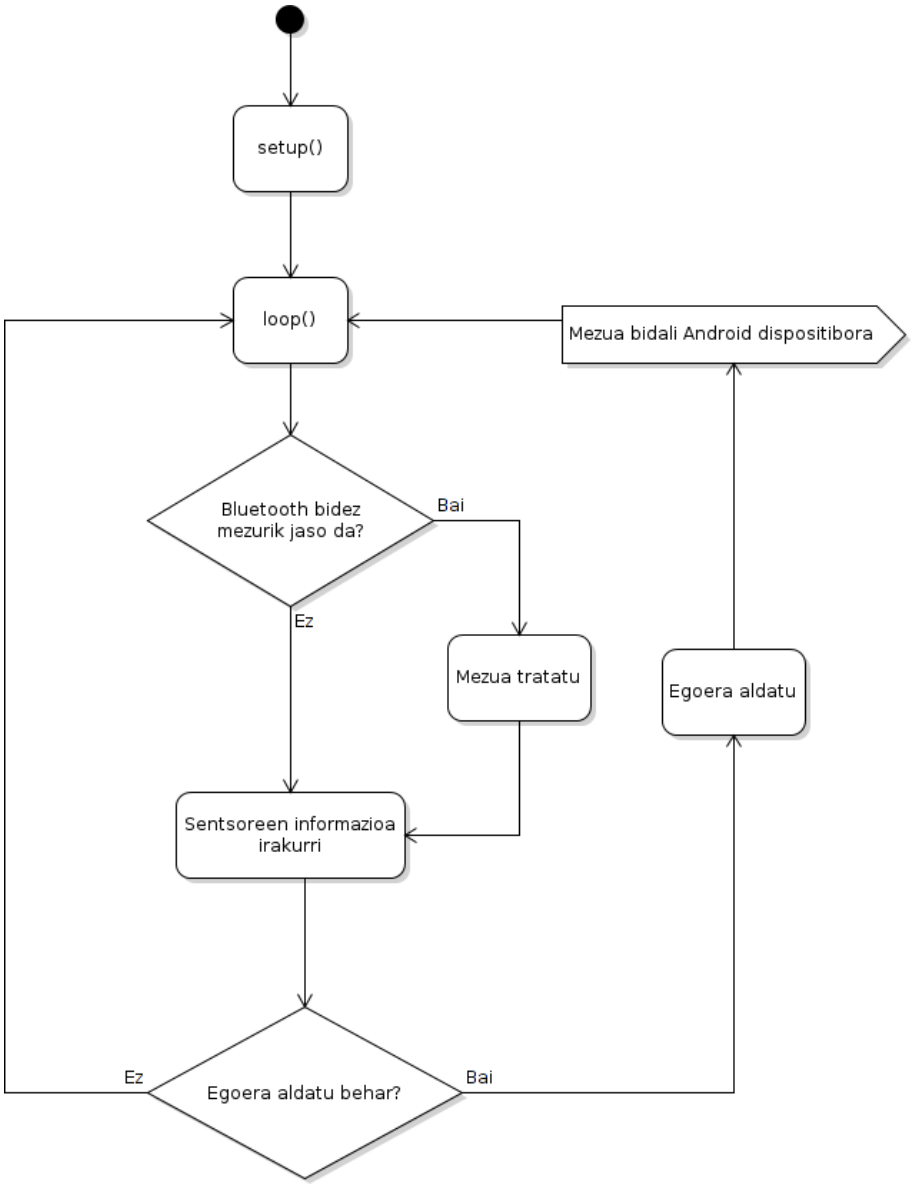
5.4 Irudia: *Ezkerrera joan* sekuentzia-diagrama.



5.5 Irudia: *Hizkuntza aukeratu* sekuentzia-diagrama.



5.6 Irudia: *Konexioa ezarri* sekuentzia-diagrama.



5.7 Irudia: Sistemaren arkitekturaren eskema.

6. KAPITULUA

Hardwarearen diseinua eta garapena

6.1 Hardwarearen diseinua

6.1.1 Arduino

Gure hardware pieza nagusia Arduino Uno plaka da, eta gainontzeko osagaiak plaka honi konektatzeko, konexioak zein eratan egin behar diren zehaztea ezinbestekoa dugu. Hasieratik garbi genekien nola lotuko genituen osagai batzuk: adibidez, Arduino Motor Shield-a, Arduino Uno plakaren gainean konektatzen da, eta motorra kontrolatzeko kableak, berriz *Shiled*-ak helburu horretarako dituen konektoreetan. Baina, beste osagai guztiak, zein eratan konektatu gure esku geratzen da. 6.1 taulan ikusi daiteke Arduinoaren pin (edo hankatxo) bakoitza zein osagai kontrolatzeko erabili den.

Horretaz gain, kontuan izan behar da osagai guztiak ez direla zuzenean Arduinoaren pinetara konektatuko; osagai batzuek erresistentziak edota zirkuitu konplexuagoak behar dituzte funtzionamendu egokia lortzeko. Osagai bakoitza zein eratan eta beste zein osagairekin konektatuko dugun definitzeko, 6.1 irudian ikus daitekeen eskema egin dugu. Nahiz eta behin-behineko diseinua den, beharrezko osagai guztiak aurkeztea lortu da eta garapena egiterako garaian, ideia nahiko garbi bat izateko balio digu.

Analisi zatian identifikatu ez genituen osagaiak erresistentziak dira. Diseinua egiteko momentuan ohartu gara elementu batzuen funtzionamendu egokirako horien beharra genuela eta guztira hiru erresistentzia erabili ditugu. Eskeman azaltzen diren hiru erresistentziak

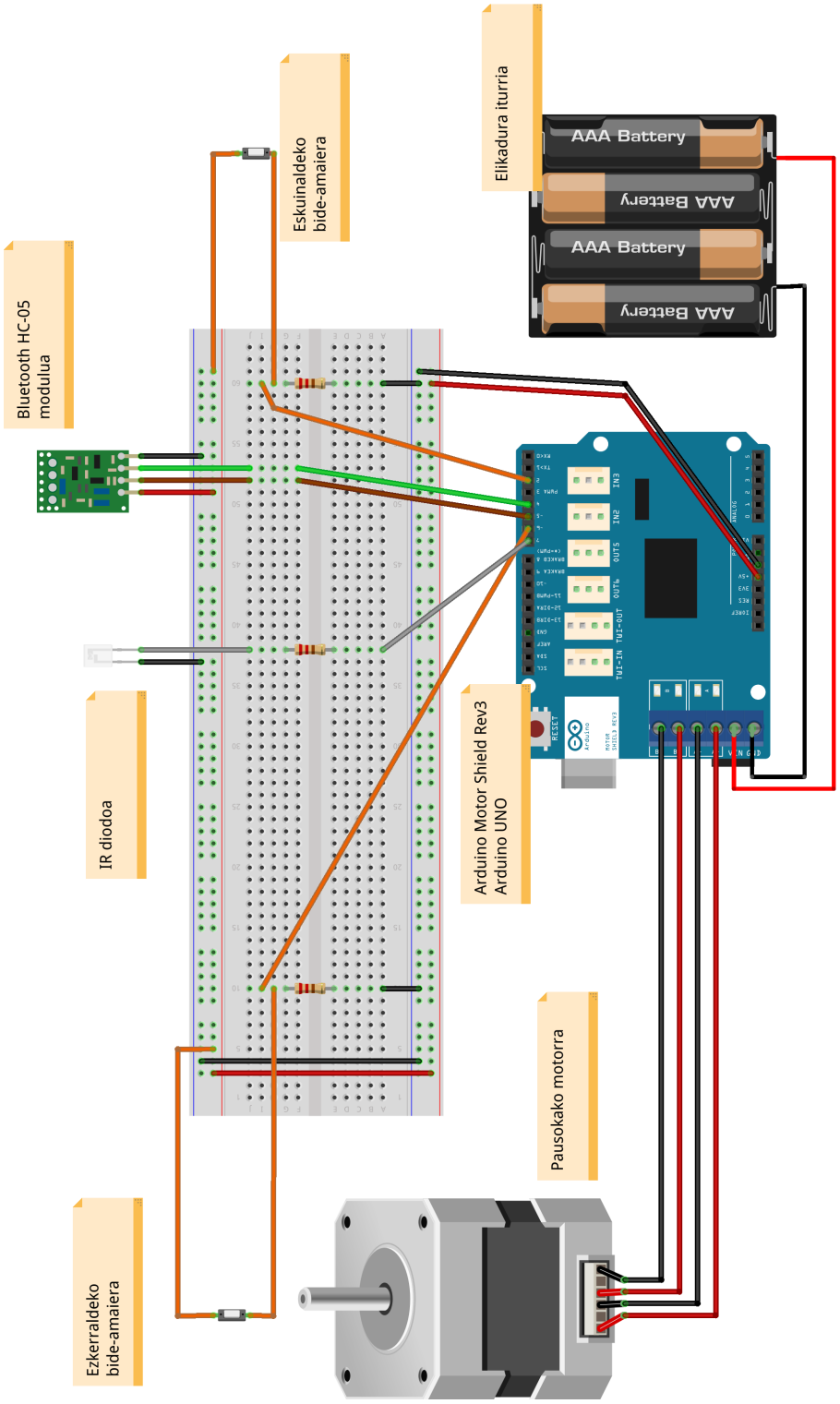
6.1 Taula: Arduinoaren pin bakoitzaren funtzioa.

Pin zenbakia	Funtzioa
0	USB serie lerroa (Rx)
1	USB serie lerroa (Tx)
2	Eskuineko bide-amaierako pultsadorea
3	Arduino Motor Shield (PWM A)
4	Bluetooth serie lerroa (Rx)
5	Bluetooth serie lerroa (Tx)
6	Ezkerreko bide-amaierako pultsadorea
7	IR diodoa
8	Arduino Motor Shield (Brake B)
9	Arduino Motor Shield (Brake A)
10	
11	Arduino Motor Shield (PWM B)
12	Arduino Motor Shield (Dir A)
13	Arduino Motor Shield (Dir A)
A0	Arduino Motor Shield (Current sensor A)
A1	Arduino Motor Shield (Current sensor B)
A2	
A3	
A4	
A5	

1000 ohm-eko erresistentziak dira. IR diodoaren kasuan, diodoari behar duen tentsioa emateko jarri da, eta bide-amaieren kasuan, tentsio altua ala baxua zuzen detektatu ahal izateko erabili dira, pultsagailuaren egoeraren arabera.

6.1.2 Dolly

Ez ditugu objektu fisiko-mekaniko bat diseinatzeko beharrezkoak diren konpetentziak eta, honenbestez, ezin izan da diseinu zehatz eta definitiboa egin. Aurrekarietan aztertutako beste diseinuetan oinarrituz, dollyaren diseinu oso sinpleak egin ditugu, inondik ere azken emaitzarekin bat etorriko direnak. Hala ere, eta diseinu-lan gehiena garapen prozesuarekin batera egingo genuela bagenekien ere, parte interesatuei ideia azaldu eta kontzeptua ulertzeko diseinu sinple batzuk egin dira. Diseinu horiek justu-justu bere funtzioa bete dute eta ez dira dokumentu honetan sartuko, proiektuari begira, informazio esanguratsurik ematen ez duela iruditu baitzaigu.



6.1 Irudia: Arduinoaren konexioen eskema.

6.2 Garapen prozesua

Arduino aplikazioa garatzen joan garen heinean joan gara hardware berria gehitzen eta probak egiten. Zorionez, diseinua egiterakoan inbertitutako denbora ondo erabilia izan zen eta garapena nahiko erraz egin da.

Hardwarea dollyan txertatzea ere ez da lan handia izan, toki nahikoa izan dugu eta beharrezko elementu guztiak erraz sartu dira. Dollyaren garapena egiterakoan, laguntza asko jaso dugu. Informatikari gisa, atal asko gure ahalmenetatik kanpo gelditzen ziren, eta beraz, ahalik eta dolly sinpleena garatzea zen helburua. Hala ere, familiartekoetatik jasotako laguntzari esker, helburu hori lortzeko guk inbertitu beharreko denbora asko jaitsi da eta emaitza asko hobetu da. Gauza guztien gainetik, dolly funtzional bat garatu nahi izan da eta estetikoki izango duen itxurari jaramon gutxi egin diogu.

Dollyak egurrezko xafla bat du hardwarearen oinarri gisa, eta horren azpian, aluminiozko pieza batzuekin, 4 gurpiltxo lotzen zaizkio oinarriari. Gurpiltxo horiek errail batzuen gainean aurrera eta atzera mugi daitezke, urratsez urratseko motorra erabiliz. Motorrak engranaje baten bidez finko dagoen zinta batean zehar indarra egiten du dollya aurrera eta atzera desplazatzeko.

Arduinoa egurrezko xaflaren gainean kokatuta dago eta dollyan kokatuta dauden osagaiei konektatuta kable bidez. Arduinoak urratsez urratseko motorra, bide amaierak, Blueooth HC-05 modulua eta kamera kontrolatzen ditu. Arduinoa eta bere osagaiak elikatzeko, berez telefono mugikorrak kargatzeko erabiltzen den *power bank* bat erabiltzen da. 6.2 eta 6.3 irudietan ikus daiteke dollyaren azken itxura.

6.2 Irudia: Dollya.

6.3 Irudia: Dollya gertutik.

7. KAPITULUA

Softwarearen inplementazioa

Kapitulu honetan sistemaren garapeneko hainbat aspektu tekniko landuko dira. Sistema garatzeko erabili diren teknologien inguruko ezagutza minimo edo orokor bat edukitzea ezinbestekoa izango da hemen azaldutakoa ondo ulertzeko.

7.1 Lizentzia

Proiektu honetako kode guztia *Apache* lizentziapean garatu da, *Apache License, Version 2.0*. Beraz, kode honen erabilera edo argitaratzea lizentzia horrek finkatzen dituen mugen eta betebeharren barruan egin behar da. Proiektuaren iturri-kodeak lizentziaren kopia bat du eta *Apache*-ren lizentziaren erabileraren txantiloia ondorengo estekan aurkitu daiteke.

https://www.apache.org/licenses/LICENSE-2.0.html

7.2 D-Lappse aplikazioa

Atal hau izan da dudarik gabe proiektuaren atalik korapilatsuena. Android garapena eremu ezezaguna genuen eta erronka handia izan da aplikazio hau Android gailu baterako garatzea. Aplikazioaren garapen guztia nola egin den azaltzen ez genuke sekula bukatuko; beraz, aplikazioaren puntu kritikoenak aztertuko ditugu, funtzionamendu orokorra

ulertzeko. Hala ere, aplikazioaren iturburu-kodean beharrezko ohar eta komentario guztiak aurkitu daitezke aplikazioaren zati bakoitza ulertu ahal izateko.

Aplikazioaren garapen osoa Android Studio garapen-ingurunean egin da; honek Android garapenerako behar diren tresna guztiak eskaintzen dizkigu. Baliabide eta kode fitxategiak antolatzeko modua ere garapen inguruneak finkatzen digu; alde batetik, `java` karpetan iturburu kode nagusia dugu, Javaz idatzia. Bestalde, `res` karpetan, kode gehigarria daukagu, aplikazioaren egitura finkatzeko edota `String` balioak finkatzeko, XML fitxategietan.

Ez da ahaztu behar, aplikazioaren garapen-prozesuan, lehenago aipatutako *Android Best Practices* Android garapenerako gomendioak jarraitu ditugula. Horretarako, laguntza handiena Android garapenerako webgune ofizialean aurkitu dugu [Developers, 2015a], bertako gomendioak jarraituz aplikazio optimizatu bat lortzeko. Gomendio horiei jarraituz, beste gauzen artean, aipatzekoa da aplikazioaren probak egiteko eta gertatzen dena ondo ulertzeko, kode guztia `Log` izeneko funtzio batzuez josita dagoela. `Log` hauekin, aplikazioaren gertaera eta erroreen berri ematen zaio Android Studio-ren araztaileari.

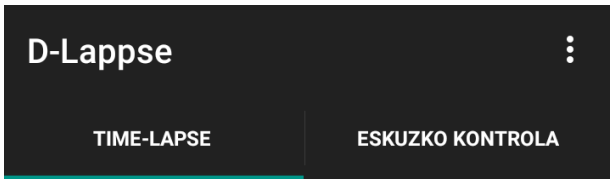
Garatutako aplikazioaren kodea azaltzeko, klase bakoitza banan-banan deskribatu beharrean, aplikazioa exekutatzen den unetik aurrera ekingo diogu azalpenari, momentu bakoitzean behar den funtzionamendua lortzeko beharrezkoa izan den kodea azalduz.

7.2.1 Aplikazioa martxan jarri

Aplikazioa exekutatu eta lehenengo exekutatzen den jarduera `MainActivity` da; jarduera guztiak metodo bera erabiliz hasten dira, `onCreate()` metodoa. Aplikazioa hasi eta erabiltzaile batek berehala ikusi nahi duena interfaze grafikoa da; beraz, hori da lehenengo eginbeharretako bat, eta hori azalduko dugu lehenik.

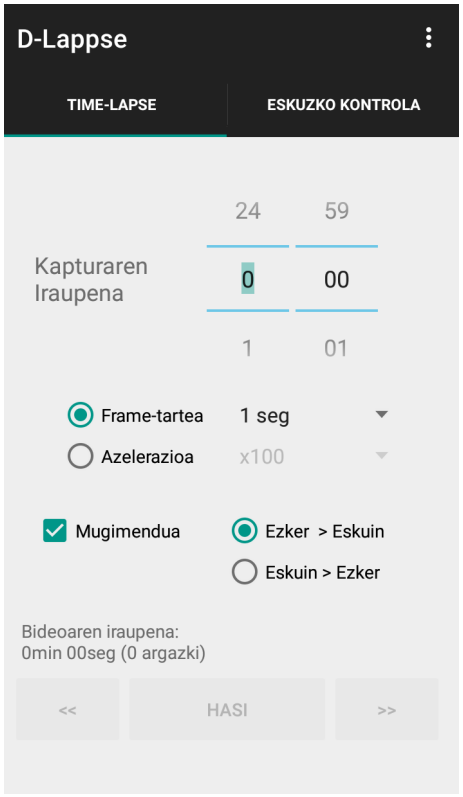
`MainActivity`-k ez ditu interfazeko elementu asko zuzenean kudeatzen, baina interfazeko elementu nagusien kudeaketaren arduraduna da. Horretarako, `SectionPagerAdapter` klasea baliatzen da, `MainActivity` klasean kabiatua aurkitzen dena. Klase hori erabiliz, pantailaren goialdeko *ActionBar-ean* bi fitxa sortzen dira eta horietako bakoitzari `Fragment` bat esleitzen zaio. Errazago ulertzeko, 7.1 irudian ikus daiteke `MainActivity` jarduerak kudeatzen duen *ActionBar-a*.

Interfazearekin jarraituz, `MainActivity`-k gehitzen dituen `Fragment`-ek, interfazeko elementu gehiago dituzte, eta erabiltzaileari eskaini behar zaizkion aukera gehienak eskaintzen dituzte. `TimelapseFragment`-ek time-lapsea egiteko beharrezkoak diren konfigura-

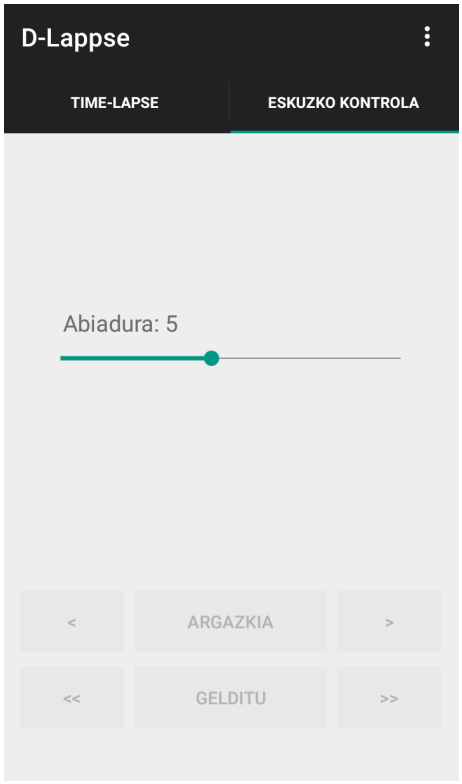


7.1 Irudia: ActionBar-a time-lapse eta eskuzko kontrola fitxak dituelarik.

zio parametroak jaso eta hauek tratatzen ditu eta `ManualFragment`-ek, berriz, eskuzko kontrolarekin erlazionatutako atazak burutzen ditu. Sortzen diren bi pantaila horien funtzio eta antolaketa ondo ulertzeko, 7.2 eta 7.3 irudiak ditugu.



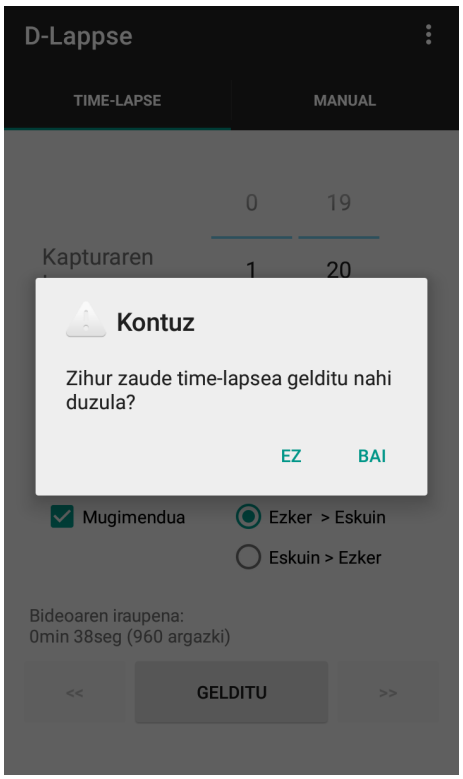
7.2 Irudia: *Time-lapse* pantaila



7.3 Irudia: *Eskuzko kontrola* pantaila

Erabiltzailearekin elkarrekintza gehien duten osagaiak dira; beraz, erabiltzaileari ahalik eta erraztasun gehien ematen saiatu gara. Horien artean, time-lapsea egiteko aukerak aukeratzen ditugun bitartean, automatikoki laburpen bat erakusten zaio erabiltzaileari. Hautatutako balioekin, time-lapseak izango duen argazki kopurua eta bideoaren iraupena adieraziz. Horretaz gain, erabilgarri ez dauden aukerak indargabetu egiten dira eta modu horretan, erabiltzaileak badaki aukera hori ez dagoela erabilgarri. Azken aukera hori oso baliagarria da dollyarekin komunikatzeko botoietan; konexiorik ez badago, botoiak in-

dargabeturik mantenduko dira, eta konexioa lortzean aktibatu egiten dira. Horretaz gain, time-lapsea egiten ari den bitartean ere aukera gehienak desaktibatu egiten dira eta time-lapsea gelditzeko aukera soilik uzten da. Hala ere, eta time-lapsea nahi gabe ez gelditzeko, *Gelditu* sakatzean konfirmazio leiho bat erakusten da; benetan gelditu nahi den galdetuz (ikus 7.4 irudia).



7.4 Irudia: Time-lapsea gelditzeko konfirmazio eskaera.

Ohar leiho hori zabaltzeko, `AlertDialog` klasea erabiltzen da. Klase honen metodoei esker, oso modu errazean aplikazioaren exekuzioa pausatu daiteke eta erantzunaren zain gelditu. Erantzuten denaren arabera gauza bat edo bestea egiten da; kasu honetan, time-lapsea gelditu edota ezer ez aldatu.

7.2.2 Konexioa ezarri

`MainActivity` aplikazioa, interfazea bistaratzeko duen arduraz gain, `ConnectionManager` haria exekutatzeaz ere arduratzen da. Klase hori hari moduan exekutatzea *Android Best Practices* gomendioei jarraiki egin da, konexioa kudeatzeak sor ditzaken mantsotzeak, aplikazioaren errendimendu orokorrari eta, batez ere, interfaze grafikoari ez diezaion era-

san. Konexioaren kudeaketa aplikazioaren alde kritikoenetakoa da, konexioa ezarri eta gertatzen diren aldaketak eta mezu trukeak akatsik gabe egin behar baitira. Gainera, erabiltzaile askorentzat konexioarena ez da erraz ulertzen den gaia, eta beraz, erabiltzailearentzat ahalik eta gardenen mantentzen saiatu gara; konexioaren kudeaketa guztia eta kudeaketa horri buruz behar-beharrezko abisuak eta eskaerak soilik egiten zaizkio erabiltzaileari.

Konexioa ezartzeko, dollyak daraman Arduinoaren Bluetooth modulua bilatu behar da, baina horretan hasi baino lehen, Bluetootharen inguruan konprobaketa batzuk egiten dira. Lehenik eta behin, Android dispositiboak Bluetooth ahalmena duen egiaztatu behar da. Edukiko ez balu, ez legoke aurrera jarraitzerik. Behin hori egiaztatuta, hurrengo pausoa Bluetooth-a aktibatuta dagoen begiratzea da. Kasu honetan ere, aktibatuta ez badago ezin da aurrera jarraitu. Aktibatuta ez badago, erabiltzaileari abisua ematen zaio, Bluetooth-a aktibatu behar duela jakinarazteko. Hasiera batean, abisua jarri beharrean, automatikoki Bluetooth aktibazio eskaera bat egitea pentsatu zen, modu horretan Bluetooth-a aktibatua ez balego, pantailan leiho berri bat azalduko litzateke aktibatu ala ez erabakitzeko. Azkenean, aukera hori baztertzea erabaki zen, erabiltzaileari bat-batean espero ez duen leiho bat azaltzea intrusiboegia iruditu zaigu eta. Hala ere, aukera horretaz baliatzen da beste momentu batean, menuko *Konektatu* botoia sakatzen denean, Bluetooth-a aktibatua ez badago aktibatzeko eskaera egiten baitzaio.

`ConnectionManager`-aren osagai nagusia `Receiver`-a da, Androiden `BroadcastReceiver` klasearen instantzia bat. Objektu hau Android sistema eragileak kudeatzen dituen ekintza-eskaerei entzuteko erabiltzen da eta, kasu honetan, Bluetooth-aren inguruko aldaketak detektatzeko erabili da. Objektua instantziatu ondoren, `receiver`-a konfiguratu egin dugu, guk behar ditugun ekintzak kudeatzeko. Bi iturri desberdin dituzte entzun behar ditugun eskaerek, alde batetik `BluetoothAdapterrarekin` eralazionatutakoak eta beste aldetik, `BluetoothDevice`-arekin erlazionatutakoak. `BluetoothAdapter` gure Android dispositiboaren Bluetooth gailua kudeatzeko erabiltzen da eta `BluetoothDevice` berriz aurkitzen diren gailuen instantziak izango lirateke.
Guztira sei eskaera mota entzuten ditu `receiver`-ak:

`BluetoothDevice`

- ACTION_FOUND: Blueooth gailu bat aurkitzen den bakoitzean jasotzen da.
- ACTION_BOND_STATE_CHANGED: Bluetooth gailu baten parekatze-egoera aldatu denean jasotzen da.

- ACTION_ACL_DISCONNECTED: Bluetooth gailu batetik deskonektatzean jasotzen da.

`BluetoothAdapter`

- `ACTION_DISCOVERY_STARTED`: beste gailuen bilaketa hasten denean jasotzen da.
- `ACTION_DISCOVERY_FINISHED`: gailuen bilaketa amaitzean jasotzen da.
- `ACTION_STATE_CHANGED`: Android dispositiboaren Bluetooth gailuaren egoera aldatzen denean jasotzen da; aktibatu edo desaktibatu egin den, adibidez.

Receiver-a konfiguratu eta zuzenean abiarazten da, eta etengabe funtzionatzen jarraitzen du sistemaren abisuak detektatzeko. Azaldu ditugun sistemaren eskaera horiek detektatuz, bilatzen ari garen gailuarekin konektatzea lortzen da. Esan bezala, gure gailuarekin konektatzeko lehenengo pausua gailua detektatzea da. Horretarako, detektatutako gailuaren izena egiaztatzen da eta bilatzen ari garen gailuaren izenarekin bat egiten duen begiratu behar da. Bat egiten badu, bi aukera ditugu: gailua dagoeneko parekatua egotea edota oraindik parekatu gabe. Parekatu gabe egongo balitz, parekatzeko eskaera egingo genuke eta ondoren, gailuarekin konektatzen saiatuko ginateke. Konektatzeko, lehenengo *socket* bat ireki behar dugu. Horretarako, `createSocket` metodoa erabili dugu. Metodo honek aurkitutako gailuaren `BluetoothDevice` instantzia bat jasotzen du eta *socket* bat sortu. Ondoren, `runSocket` eta `manageConnectedSocket` metodoak erabili ditugu konexioa ezarri eta kudeatzeko. Konexioa ezartzeko, `Connection` klasea erabili dugu eta klase horren instantzia bat sortu ondoren klase horrek eskaintzen dizkigun metodoak erabili ahal izateko. Baina konexio horretan zehar komunikazioa bideratzea ez da hain erraza; idazteko ez dago arazo handirik, nahikoa da `write` metodoa erabiltzea; baina *socket*-aren bitartez datozen mezuak entzuteko, etengabe entzuten egon beharra dago eta etengabe entzuten egongo bagina, ezingo genuke mezurik bidali. Hori konpontzeko, `Connection` klasearen beste instantzia bat sortu dugu eta beste hari batean exekutatu gelditu gabe entzuten egon dadin (ikus 7.1 kodea).

7.1 Kodea: ManageConnectedSocket metodoa.

```
1  private void manageConnectedSocket(BluetoothSocket socket){
2      connection = new Connection(socket, mHandler);
3      Log.v(TAG, "Konexioa ezarri da");
4
5      mHandler.obtainMessage(Constants.SUCCESS_CONNECT).sendToTarget();
6
```

```
7           Thread listenThread = new Thread(new Connection(socket, mHandler));
8           listenThread.start();
9       }
```

7.2.3 Informazio-fluxua

Klaseen arteko komunikazioa ezinbestekoa da gure aplikazioan, informazioa aplikazioaren osagai desberdinek jasotzen baitute; interfaze grafikotik jasotzen diren aginduak edota Bluetooth bitartez jasotzen diren mezuak, adibidez. Aplikazioaren diseinua egiterakoan azaldu bezala, informazio guztia `MainActivity` jardueratik igaroko da eta aplikazioaren informazio-fluxua kudeatzeaz arduratuko da. Bi arazo mota identifikatu ditugu gure aplikazioan klase batetik bestera informazioa pasatzeko: alde batetik, interfazeko elementuen informazioa `TimelapseFragment` eta `ManualFragment` klaseetatik, `MainActivity`-ari pasatzea eta beste aldetik, `ConnectionManager` eta `Connection` harietatik `MainActivity`-ra. Zorionez, Androidek nahiko definituak ditu kasu bakoitzean erabili behar diren tresnak.

Errazenetik hasiko gara. `TimelapseFragment` eta `ManualFragment` klaseak, `MainActivity` klasearen umeak dira, eta honenbestez, klase umeetan interfaze bat definitu behar da klase gurasoaren metodoekin. Ondoren, `FragmentListener` klaseaz baliatuz, *callback* objektu bat sortuko dugu eta objektu horren bitartez klase gurasoaren metodoei dei egin, behar ditugun aldagaiak pasatuz.

`MainActivity` eta harien arteko komunikazioa pixka bat konplexuagoa da. Kontuan izan behar da batera exekutatzen ari diren prozesuak direla, eta honenbestez, prozesu batek besteari mezuren bat bidaltzen badio eta hau okupatuta badago, erantzuna ez dela berehalakoa izango. Hala ere, gure kasuan behintzat, ez dugu inongo arazorik izan eta ekintza guztiak berehala exekutatzen dira. Komunikazio hau gauzatzeko, Androiden `Handler` klaseaz baliatu gara, eta `mHandler` objektua instantziatu dugu. `MainActivity` klasean objektu hau konfiguratzen da eta jasoko dituen mezuak identifikatzen dira, baita jasotakoan egin behar dena definitu ere. Ondoren, `ConnectionManager` eta `Connection` hariak sortzerakoan, objektu hau pasatzen zaie eta, modu horretan, hari horietan eskura egongo da objektu hori. Horrela, `ConnectionManager` hariak konexioa ezartzen duenean edo `Connection` hariak mezu bat jasotzen duenean, `mHandler` objektuaren bitartez igortzen da mezu hori eta `MainActivity`-ari iristen zaio. 7.2 kodean ikus daiteke erabilitako *handler*-aren funtzionamendua, jasotzen duen mezuaren arabera.

7.2 Kodea: Handler-aren mezu tratamendua.

```
private void setupHandler(){
       mHandler = new Handler(Looper.getMainLooper()) {
           @Override
           public void handleMessage(Message inputMessage) {
               switch(inputMessage.what){
                   case Constants.SUCCESS_CONNECT:
                       connection = ConnectionManager.connection;
                       Toast.makeText(getApplicationContext(),
    ctx.getString(R.string.toast_connection_success), Toast.LENGTH_SHORT).show();
                       sendCommand(Constants.SEND_CONFIG, null);
                       shouldExecuteOnResume=true;
                       if(state == Constants.STATUS_NOT_READY){
                           state=Constants.STATUS_READY;
                       }
                       enableAllButtons();
                       break;
                   case Constants.ENABLE_BT:
                       enableBt();
                       break;
                   case Constants.CONNECTION_LOST:
                       connectionLost();
                       break;
                   case Constants.MESSAGE_READ:
                       String msg = (String) inputMessage.obj;
                       if(msg.equals("CONFOK")){
                           Log.v(TAG, "Konfigurazioa ondo ezarri da");
                       }else if(msg.equals("CONFKO")){
                           Log.e(TAG, "Konfigurazioa ezin izan da ezarri");

                   (...)
               }
           }
       };
```

Azkenik, `MainActivity` klasea informazio-fluxuaren bilgunea denez, bertan sortu dugu `sendCommand` metodoa. Metodo honek komando bat eta parametro gehigarriak (komandoak behar baditu) jasotzen ditu eta horien arabera mezu bat igortzen du irekita dagoen konexioan zehar.

7.2.4 Lokalizazioa

Lokalizazioa asko zaindu nahi izan den beste aspektua izan da. Oraingoan ere, *Android Best Practices* gomendioei jarraituz, metodologia jakin bat erabili dugu aplikazioa garatzerakoan. Ideia nahiko sinplea da, erabiltzaileari eskaintzen zaion testu guztia `values`

karpetan dagoen `Strings` XML fitxategian idazten da, eta, ondoren, behar denean, testu hori erreferentziatzen dugu. Sistema honek lan gehiago eskatzen du programatzerako unean, baina behin aplikazioaren testu guztiak ondo erreferentziatuak ditugunean, oso erraza da hizkuntza berriak gehitzea. Hizkuntza berri bat gehitzeko, beste `values` karpeta bat sortu behar da hizkuntza bakoitzerako; `values-eu` euskararentzako eta `values-es` gaztelaniarentzako gure kasuan. Ondoren, `Strings` XML fitxategi bat sortu eta testua erreferentziatzeko erabili den identifikadore bera erabiliz, hizkuntza bakoitzarentzat testu alternatiboa sartzen da.

Hala ere, ez dira denak abantailak: Android sistemak, telefonoaren hizkuntza zein den detektatzen du eta, automatikoki, aplikazioa hizkuntza horretan exekutatzen du. Horrek hasiera batean, abantaila dirudien arren, gure kasuan ez da hala gertatzen. Nahiz eta Android 4.0tik aurrera Androidek euskararen euskarria eduki, telefono gehienek ez dute aukera hori. Horrenbestez, telefonoa euskaraz ezin bada konfiguratu gure aplikazioa ere ezingo litzateke euskaraz ikusi. Beraz, beste modu bat bilatu behar izan dugu hori konpontzeko.

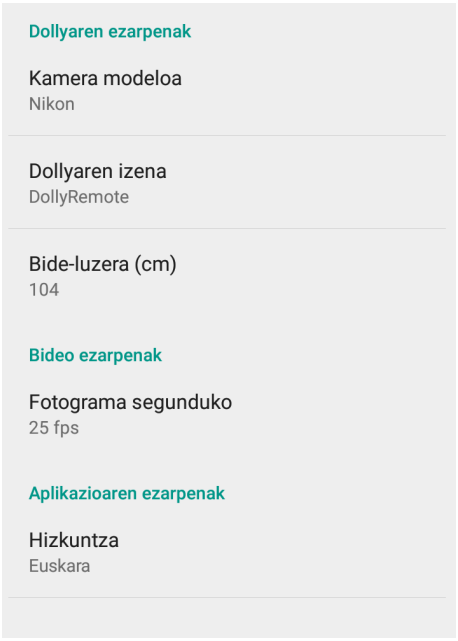
Alde batetik, automatikoki hizkuntza ezartzea ekidin dugu, lokalizazioa aplikazioaren ezarpenetatik aukeratuz, eta bestetik, erabiltzaileak nahi duen hizkuntza aukeratzeko, ezarpenetan, aldatzeko aukera emanez. 7.3 kodean ikus daiteke lokalizazioa behartzeko sortu dugun kodea. Kode hori hiru momentutan exekutatzen da: aplikazioa irekitzen denean, ezarpenetan hizkuntza aldatzen dugunean eta ezarpen pantailatik pantaila nagusira joaten garenean. Modu horretan, ezarpenetan aukeratzen dugun hizkuntza beti bistaratua izango dugu eta ez dugu aplikazioa berrabiarazi beharko.

7.3 Kodea: Lokalizazioa ezartzeko metodoa.

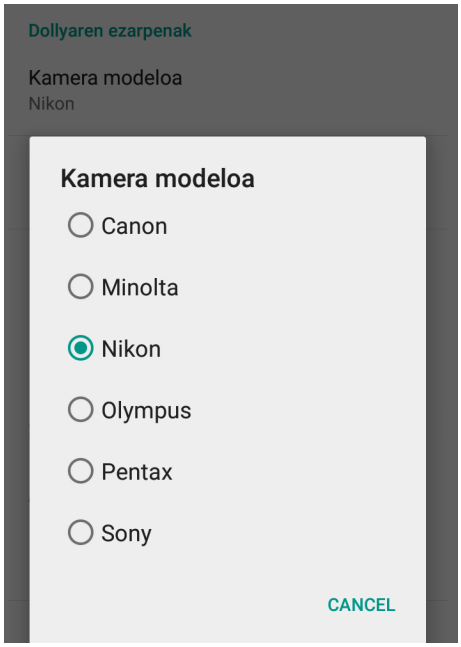
```
1  public void setLocale(String lang) {
2          myLocale = new Locale(lang);
3          Resources res = getResources();
4          DisplayMetrics dm = res.getDisplayMetrics();
5          Configuration conf = res.getConfiguration();
6          conf.locale = myLocale;
7          res.updateConfiguration(conf, dm);
8          Intent refresh = new Intent(this, MainActivity.class);
9          startActivity(refresh);
10         finish();
11     }
```

7.2.5 Ezarpenak

Hasiera batean, atal hau gehiegi lantzea pentsatu ez bagenuen ere, garapenak aurrera egin ahala sistemaren konfigurazioko zenbait aukera erabiltzaileari ematea erabaki dugu. Honek gure aplikazioari balio erantsi bat ematen diola iruditu zaigu; izan ere, konfigurazio parametro batzuk, erabilgarriak direnak noski, erabiltzaileari eskaintzeak, sistema malguago bat sortzeko balio du. Ez ditugu itsu-itsuan ahalik eta ezarpen gehien atera, baizik eta erabiltzailearentzat baliagarriak izan daitezkeenak soilik. Horien artean, nahiko aipagarriak dira dollyaren bidearen luzera hautatzeko aukera, edota bilatu behar dugun dollyak daraman Bluetooth moduluaren izena aukeratzea. Modu honetan, dollyak aldaketaren bat izango balu, edota dolly-ari beste bide bat jarriko bagenio, ez litzateke aplikazioa aldatu beharko, ezarpenak aldatu besterik ez. Ezarpen hauetaz gain, badira beste batzuk ere, hasieratik erabakita genuen kamera modeloa aldatzeko aukera, esate baterako. 7.5 eta D.8 irudietan ikus daiteke ezarpenen pantailaren itxura.



7.5 Irudia: Ezarpenen pantaila.



7.6 Irudia: Kamera modeloa aukeratzeko leihoa.

7.3 Dollyaren driverra

Dollyarentzako driverra garatzeko, Arduinorentzat bereziki prestatutako garapen ingurunea erabili dugu. Nahiko IDE sinplea da eta laguntza gutxi eskaintzen du, baina Ardui-

norentzat bereziki prestatua denez, abantaila interesgarri batzuk ematen dizkigu. Hona hemen interesgarrienak: kodearen konpilazioa aplikaziotik bertatik egiteko aukera ematen digu eta serie-lerroko kontsola ere eskaintzen digu.

Kode nagusia fitxategi bakar batean kokatzen da, `main.ino` fitxategian. Horretaz gain, eta Arduinoren liburutegitaz gain, beste bi liburutegi erabili ditugu: `StepperMotor` eta `MultiCamaraIrControl`. `StepperMotor` liburutegian urratsez urratseko motorra kontrolatzeko erabili dugun kodea daukagu; Arduino Motor Shield-arentzat liburutegi egokiturik aurkitu ez dugunez, guk sortu dugu liburutegi hau, beharrezkoak ikusi ditugun funtzioak inplementatuz.

Arduino aplikazioaren garapena, osagaiz osagai egin da, hau da, kontrolatu beharreko osagai bat hartu eta hori kontrolatzeko beharrezkoa den kodea garatu, ondoren hurrengoarekin jarraitzeko. Hasiera batean, hiru ziren garapenari begira konplexuenak ziruditen osagaiak: motorraren kontrola, kameraren kontrola eta Bluetooth bidezko komunikazioa.

Buruhauste gehien eman digun elementua motorra izan da; hasiera batean erabiltzea pentsatu genuen motorra aldatu egin behar izan dugu eta. Motor hori maneiatzeko, gure kontroladoreak ez zuen nahikoa ahalmen eta horren ordez, urratsez urratseko motor bat erabili behar izan dugu. Baina urratsez urratseko motorra kontrolatzea ere ez da erraza izan, motor bereziak dira, eta hauek kontrolatzeko beharrezko liburutegiak ez ditugu topatu. Honenbestez, guk garatu behar izan dugu liburutegi egoki bat motorra kontrolatzeko. Liburutegi hori garatzea ez da erraza izan, ikerlan ugari eskatu baitu; hala ere, azkenean merezi izan du: liburutegia guk geuk inplementatzean guk behar genituen funtzioak garatu ditugu, eta, modu horretan, aplikazioa garatzerako garaian, gauzak asko erraztu dizkigu. Hauek dira liburutegiak eskaintzen dizkigun funtzioak:

7.4 Kodea: StepperMotor liburutegiaren funtzioa publikoak. (`StepperMotor.h`)

```
1    StepperMotor();
2    StepperMotor(int pin_pwm_chA, int pin_pwm_chB, int pin_dir_chA, int pin_dir_chB, int
      pin_brake_chA, int pin_brake_chB, int _steps);
3    void speed(int rpm, int dir);
4    int step(int dir);
5    int steps(int steeps, int dir);
6    int steps(int steeps, int rpm, int dir);
7    void stop();
```

Liburutegi hau garatzerako garaian, hainbeste lagundu digun Arduino komunitateari laguntza hori itzultzeko aukera ikusi dugu. *Arduino Motor Shield*-arentzat liburutegi ofizialik ez dagoenez, gure liburutegia komunitateari laguntzeko erabiltzea erabaki dugu,

eta horretan pentsatuz, beste urratsez urratseko motorrekin bateragarria izateko funtzioak gehitu ditugu. 7.4 kodeko 1 eta 2 lerroetan ikusten den moduan, bi eraikitzaile eskaintzen ditu liburutegiak, bata parametrorik gabe eta bestea parametroekin. Parametrorik gabekoak, gure proiekturako bereziki pentsatutako konfigurazioa duen motorrari dagokion klasea sortzeko balio du, beste eraikitzaileak ordea, beste motor-modelo bat konfiguratzeko balioko digu. Horretaz gain, gure proiekturako baliagarriak suertatuko zaizkigun beste funtzio batzuk ere sortu ditugu, mugimendu kontrolatuak sortu ahal izateko. Gure kode nagusian, *StepperMotor* klasearen instantzia bat sortzen dugu, eta ondoren, objektuaren metodoak erabili ditugu behar izan dugunean oso modu errazean.

Hurrengo urratsa kamera kontrolatzea izan da; hasiera batean nahiko konplexua iruditu bazitzaigun ere, nahiko sinplea izan da. 4. kapituluan azaldutako aurrekariak aztertzerakoan, Sebastian Setz-ek bere *Camera Ir Control Library* proiektuan garatutako liburutegi bat topatu dugu [Setz, 2011], erabilera libreko lizentziapean, beraz, liburutegi horretaz baliatu gara. Nahikoa erraza izan da liburutegia ulertu eta erabiltzea; kamera-modelo bakoitzerako eraikitzaile bat eskaintzen digu eta, ondoren, metodo bati deitu behar diogu argazkia ateratzeko. kamera-modelo ezberdinak ditugunez, kamera-modelo bakoitzeko objektu bat sortu behar izan dugu eta ondoren, aukeratutako kamera-modeloaren baitan objektu bat edo beste erabili, 7.5 kodean azaltzen den moduan.

7.5 Kodea: kamera-modelo bakoitzarentzat argazkia ateratzeko deia. (`main.ino`)

```
1  void argazkiaAtera(){
2    if(kamera_modeloa == "NI"){
3      nikon.shutterNow();
4    }else if(kamera_modeloa == "CA"){
5      canon.shutterNow();
6    }else if(kamera_modeloa == "OL"){
7      olympus.shutterNow();
8    }else if(kamera_modeloa == "PE"){
9      pentax.shutterNow();
10   }else if(kamera_modeloa == "SO"){
11     sony.shutterNow();
12   }else if(kamera_modeloa == "MI"){
13     minolta.shutterNow();
14   }
15 }
```

Bluetooth komunikazioa ezartzea ere nahiko sinplea izan da Arduinoan. HC-05 Bluetooth moduluak puntu ireki bat bezala funtzionatzen du eta Android aplikazioa da bertara konektatu eta Bluetooth komunikazioa kudeatzen duena. Arduino aplikazioan egin behar izan dugun bakarra, serie-lerro bidezko komunikazio bat ezartzea izan da Bluetooth mo-

dulua konektatutako portuetan. Modu horretan, Android aplikazioak konexioa ezartzen duenean, informazioa idatzi eta irakurri egin daiteke bertan.

Aipatu ditugun osagaiak behar bezala funtzionatzeko, Arduinoan konektatuta dauden pinen arabera egin behar dira aldagaien erazagupenak, 7.6 kodean ikus daiteke zein modutan egin diren osagai horien erazagupenak eta liburutegien inportazioak.

7.6 Kodea: Osagaien erazagupenak. (`main.ino`)

```
#include <multiCameraIrControl.h>
#include <StepperMotor.h>
#include <SoftwareSerial.h>

// BLUETOOTH SERIALa KONFIGURATU
#define RxD 4
#define TxD 5
SoftwareSerial BTSerial(TxD, RxD);
// PAUSOKAKO MOTORRA KONFIGURATU
StepperMotor motorra;
// kamera-modelo EZBEDINAK LORTU
Nikon nikon(7);
Canon canon(7);
Olympus olympus(7);
Sony sony(7);
Minolta minolta(7);
Pentax pentax(7);
```

Osagaien erazagupenak eta metodo laguntzaileak alde batera utzita, aplikazioa bi zati nagusitan banatzen da: hasieraketak eta funtzionamendua. Hasieraketak `setup` funtzioan egiten dira eta hori behin bakarrik exekutatzen da. Ondoren, amaigabeko begizta batean sartzen da: `loop` funtzioa. Begizta honetan etengabe funtzionatuko du aplikazioak behar diren ataza guztiak burutzeko. Horretarako, egoera desberdinak erabili ditugu portaera egokiak lortzeko; begizta nagusiak honako portaera hau jarraitzen du: lehenengo kanpotik datorren informazioa egiaztatzen du eta ondoren, egoeraren arabera, portaera bat edo bestea hartuko du. Begizta nagusi honek beste metodo laguntzaile batzuk erabiltzen ditu eginkizun desberdinetarako, begizta nagusia ahalik eta argien ikus dezagun. Kanpoinformazioa jasotzeko lehenengo Bluetooth modulua egiaztatzen du, eta mezu bat jaso bada, mezua tratatzeko `mezuaTratatu` (7.7 kodean) metodoari dei egiten dio.

Mezua tratatzeko, lehenik eta behin komandoa identifikatu beharra dago. Horretarako, gure komunikazio protokoloan definitua dugun moduan, mezuaren lehenengo byteak aztertzen dira. Ondoren, komando bakoitzaren baitan, tamaina ezberdinetako parametroak

jasotzen ditugu. Tratatzeko komando konplexuena `START` komandoa da, time-lapsea hasteko agindua. Agindu honek hiru parametro ditu eta mota ezberdinetako balioak nahasirik ditu gainera. Horietako balio bakoitza jaso eta behar den motako aldagaietan gorde behar da. 7.7 kodean ikus daiteke komandoa nola aztertzen den eta ondoren, horren baitan, mezu bakoitza modu ezberdinean tratatzen dela.

7.7 Kodea: Mezua tratatzeko funtzioa. (`main.ino`)

```
void mezuaTratatu(String msg) {
  String command = msg.substring(0, 4);
  Serial.print("Komandoa: " + command + "\n");
  Serial.print("Mezu osoa: " + msg);
  switch (komadoaLortu(command)) {
    case STOP_:
      Serial.print("Gelditu egingo da\n");
      egoera = GELDIK;
      break;
    case START_:
      {
        String numPhotosString = msg.substring(4, 9);
        String frameIntervalString = msg.substring(9, 14);
        String movDirString = msg.substring(14, 16);

        numPhotos = numPhotosString.toInt();
        double frameInterval_double = frameIntervalString.toFloat();
        frameInterval_double = frameInterval_double * 1000;
        frameInterval = (int) frameInterval_double;

        (...)
```

Mezua tratatu ondoren, albo-etengailuak irakurtzen dira, eta ondoren, egoera egiaztatzeari ekiten zaio. Etengailuaren irakurketek garrantzi berezia dute mugimendua egin behar den kasuetan, ez baitugu nahi, albo bateko etengailua zapaldua dagoenean, alde horretara gehiago joaterik. Beraz, jasotako informazioarekin egoera batean sartuko da eta beharrezko eragingailuei aginduak bidaliko zaizkie. 7.8 kodean ikus dezakegu egoera bakoitza zein modutan tratatzen den. Oraingo honetan ere, portaera konplexuena time-lapsearena da, beharrezko portaera lortzeko 3 fase igaro behar baititu: hasierako desplazamendua, argazkia atera eta amaierako desplazamendua. Hiru horien konbinazioarekin lortuko da time-lapsearen mugimendu sinkronizatua argazkiekiko. Azkenik, laugarren aukera bat ere badugu, mugimendurik gabeko time-lapsea sortzeko.

7.8 Kodea: Egoeraren tratamendua. (`main.ino`)

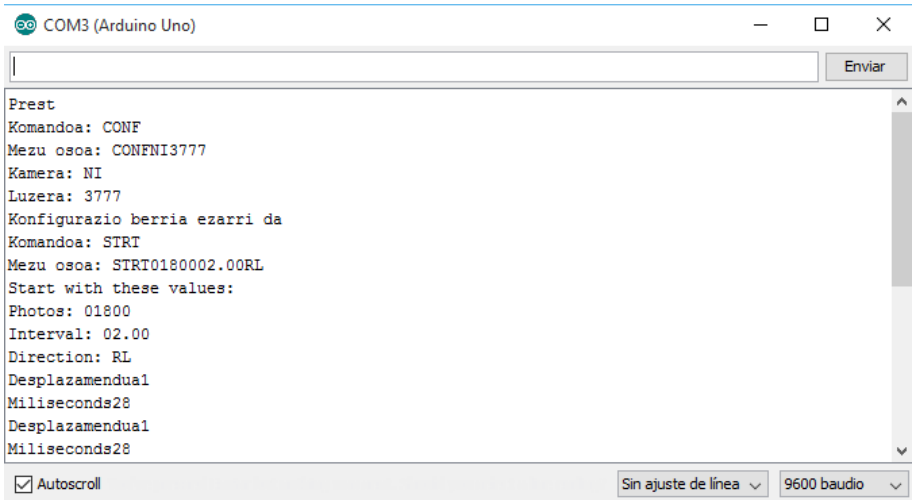
```
(...)

eskubiBotoia = digitalRead(eskubiBotoiaPin);
ezkerBotoia = digitalRead(ezkerBotoiaPin);

/* Egoera tratatu */
switch (egoera) {
  case GELDIK:
    motorra.stop();
    break;
  case EZKERRERA:
      if(ezkerBotoia == LOW){
        motorra.speed(rpm, 0);
      }else{
        motorra.stop();
        BTSerial.write("DENDL\n");
        egoera=GELDIK;
      }
      break;
  case ESKUBIRA:
    if(eskubiBotoia == LOW){
      motorra.speed(rpm, 1);
    }else{
        motorra.stop();
        BTSerial.write("DENDR\n");
        egoera=GELDIK;
    }
    break;
  case TIME_LAPSE:
      if(((ezkerBotoia == HIGH) && (movDir==0)) || ((eskubiBotoia == HIGH)&&(movDir==1))){
          BTSerial.print("FIN_\n");
          egoera=GELDIK;
          break;
      }
      switch (timelapse_egoera) {
          case 0:
              desplazamendua = (bide_luzera / numPhotos)/2;
              if(egindakoPausoKop <= desplazamendua){
                  miliseconds = miliseconds + motorra.step(movDir);
                  egindakoPausoKop++;
                  break;
              }else{
                  egindakoPausoKop=0;
                  delay((frameInterval/2) - miliseconds);
                  timelapse_egoera = 1;
                  break;
              }
              break;
          case 1:
```

```
51                  argazkiaAtera();
52                  delay((frameInterval/2) - miliseconds);
53                  miliseconds=0;
54                  timelapse_egoera = 2;
55                  break;
56              case 2:
57                  desplazamendua = (bide_luzera / numPhotos)/2;
58                  if(egindakoPausoKop <= desplazamendua){
59                      miliseconds = miliseconds + motorra.step(movDir);
60                      egindakoPausoKop++;
61                  }else{
62                      egindakoPausoKop=0;
63                      miliseconds=0;
64                      timelapse_egoera = 0;
65                  }
66                  break;
67              case 3:
68                  argazkiaAtera();
69                  delay(frameInterval);
70                  break;
71              }
72          break;
73    } //switch egoera amaiera
74  } // end loop
```

7.8 kodeko 17, 26 eta 32 lerroetan ikus daitekeen moduan, Bluetooth bidez mezuak bidaltzen dira behar denean, eta modu horretan, Android aplikazioari sistemaren egoeraren berri ematen zaio. Horretaz gain, garapenerako oso garrantzitsua izan den USB bidez serie-lerro bidezko mezu asko txertatu ditugu, aplikazioa probatzerakoan akatsak detektatzeko behar-beharrezkoak izan baititugu. Arduinoren IDEak berak serie-lerro bidezko kontsola eskaintzen digunez, nahiko erosoa suertatu zaigu lan horretarako. 7.7 irudian ikus daiteke nola ageri den aplikazioaren exekuzioa kontsolan.



7.7 Irudia: Serie-lerroaren kontsola.

8. KAPITULUA

Sistemaren ebaluazioa eta hobekuntzak

Kapitulu honetan garatutako sistemaren gaitasunak ikusteko egindako probak azalduko dira, baita proba horiek egindakoan detektatutako arazoak eta emandako konponbideak ere.

Sistemaren garapenaren hasiera-hasieratik egin ditugu probak, osagai berri bat edota funtzionalitate berri bakoitza inplementatzean, frogatu egin dugu ea espero zen emaitza lortzen genuen. Modu horretan, funtzionalitate berriak garatzean ez ditugu akats berdinak errepikatu eta gero eta emankorragoak izan gara. Sistema bere osotasunean garatu dugunean, proba sakonagoak egin ditugu, sistemaren funtzionamendu normaletik at dauden egoerak behartuz.

Egindako probak bi multzo nagusitan banatu ditzakegu: sistemaren funtzionamendu egokia bermatzeko probak eta erabiltzaileari behar den informazioa ematen zaiola bermatzeko probak; hau da, erabiltzaileak ezer arraroa ez sumatzea aplikazioan eta galdua ez sentitzea.

Sistemak behar bezala funtzionatzen duela frogatzeko nahiko proba sinpleak egin ditugu: time-lapseak egin, konfigurazio ezberdinak erabiliz; eskuzko kontrola probatu; bidearen ertz batera eraman dollya eta alde horretara gehiago joateko agindua bidali... Proba horien ondoren, nahiko emaitza onak izan ditugu, sistemaren portaera ona da, kontrol manualak berehala erantzuten du eta time-lapseak ere finkatutako denbora eta argazki kopurua betetzen du. Hala ere, aurkitu ditugu arazo batzuk. Horietako batek eskuzko kontrolarekin du zerikusia, eskuzko kontrolean aukeratzen den abiadurarekin hain zuzen ere.

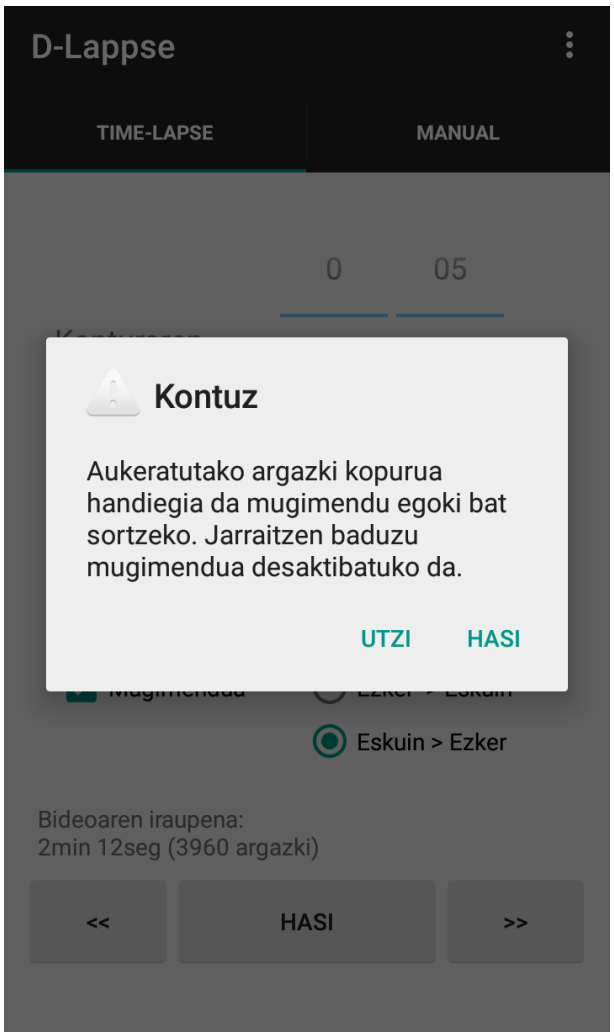
- Desplazamenduaren abiadura maximo eta minimoak: abiadura maximoan jartzen badugu (10 balioa), urratsez urratseko motorrak ez du portaera zuzena erakusten eta minimoan ere (1 balioa) berdina gertatzen da. Honen arrazoia motorraren abiadura maximoa eta abiadura minimoa gainditzen ditugula da eta ezin dira abiadura horiek sortu. Hori konpontzeko, Arduinoaren aplikaziora jo dugu, bertan abiaduraren balio horri aplikatzen zaion biderkatzailea aldatu dugu, abiadura maximoaren eta minimoaren arteko diferentzia txikituz.

Aurkitu ditugun gainontzeko arazoak time-lapsearekin dute erlazioa, time-lapsea egiteko aukeratzen diren balioekin hain zuzen ere. Bi arazo sor ditzakete aukeratutako balioek:

- Desplazamendua azkarregia izatea: time-lapsea egiterakoan, dollyak bidearen luzera osoa hartzen du mugimendua sortzeko, eta arazoa denbora balio txikiak aukeratzen direnean sortzen da. Denbora balio txikiegia hartzen bada, argazkia atera behar den posizio batetik hurrengo posiziora dagoen distantzia handiagoa da, eta distantzia hori egiteko denbora frame-tartearen denbora baino handiagoa bada, ez du nahikoa denbora izango desplazamendua sortzeko. Hori konpontzeko, zoritxarrez ez dugu aukera askorik: hardwarea hobetuz lortuko genuke, baina proiektuaren irismenetik kanpo geratzen da. Hala ere, oso konfigurazio arraroek sortzen dute honelako kasu bat, ez baita batere ohikoa denbora oso laburreko time-lapseak egitea. Honenbestez, kasu hauetarako erabaki duguna, frame-tartea luzatzea izan da. Hasiera batean, segundo bat baino motzagoak ziren frame-tarteak eskaintzen ziren, baina, hau ikusita, frame-tarte minimoa segundo batekoa izatea erabaki da. Horrela, ziur gaude desplazamendua sortzeko denbora nahikoa izango dugula.

- Desplazamendua txikiegia izatea: kasu honetan, pausokako motorrak duen mugarekin topo egin dugu, pausokako motorraren desplazamendu minimoa "pauso"bat da eta gure bide guztia egiteko 3777 pauso direla identifikatu dugu. Ondorioz, bideak dituen pauso kopuruak baina argazki gehiago eskatzen dituen time-lapseak, pauso bat baino txikiagoa den desplazamendua eskatuko luke, eta hori ezin dugu egin. Kasu honetan ere, hardwareak mugaturik gaude, eta hau konpontzeko egin dugun mugimendua desaktibatzea izan da. Noski, erabiltzaileari abisu bat jarriko zaio aukeratu duen konfigurazioa ez dela egokia adieraziz eta mugimendua desaktibatu egingo dela (ikus 8.1 irudia).

Sistemaren funtzionamendu zuzena alde batera utzita, aplikazioak egoera ezberdinen aurrean nola erantzuten duen ikusteko probak egin ditugu. Hasiera hasieratik zaindu dugun



8.1 Irudia: Mugimendua ezin da burutu.

atala izan da honakoa, aplikazioa kontsistentea izatea eta erabiltzaileari behar duen informazioa ematea. Hala ere beti agertu daitezke pentsatu ez ditugun kasuak. Informazioa bi modutan ematen da: alde batetik, testu-mezuak ditugu, eta beste aldetik, interfazeko elementuen itxura-aldaketak. Adibidez, time-lapsea hastean, *Gelditu* botoia izan ezik, gainerako botoiak desaktibatu egiten dira eta modu horretan erabiltzaileak badaki time-lapsea martxan dagoela eta ezin duela beste agindurik bidali. Time-lapsea hasi eta gelditu bezalako probak egitean ez da arazorik sumatu; modu egokian desaktibatzen dira botoiak eta baita aktibatu ere.

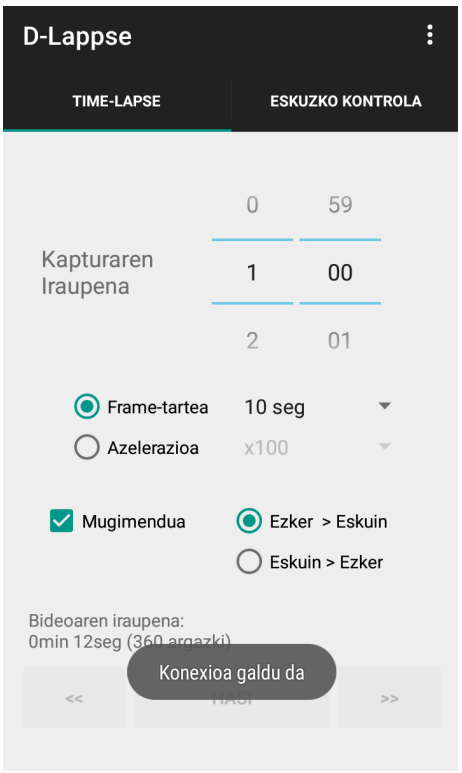
Aplikazioa baldintzatua dago guztiz Bluetooth konexiora, aplikazioak ezin baitu agindurik bidali edota informaziorik jaso konexioa ez badu. Beraz, Bluetooth konexioaren egoeraren baitan ekintza desberdinak daude egiteko. Honako proba hauek egin dira:

- Bluetooth modulua desaktibatua egotea: aplikazioa exekutatzean konexioa automatikoki ezartzen saiatzen da, baina mugikorraren Bluetooth modulua ez baldin badago aktibatuta ezin da konexio hori burutu. Kasu horretan, erabiltzaileari Bluetooth modulua aktibatu behar duela abisatzen zaio eta botoi guztiak desaktibatuta mantentzen dira. Erabiltzailearen erosotasunerako, *Konektatu* botoia sakatzen bada, Bluetooth modulua aktibatzeko eskaera egingo zaio gailuari, eta eskaera onartua bada, automatikoki hasiko da bilaketa egiten.

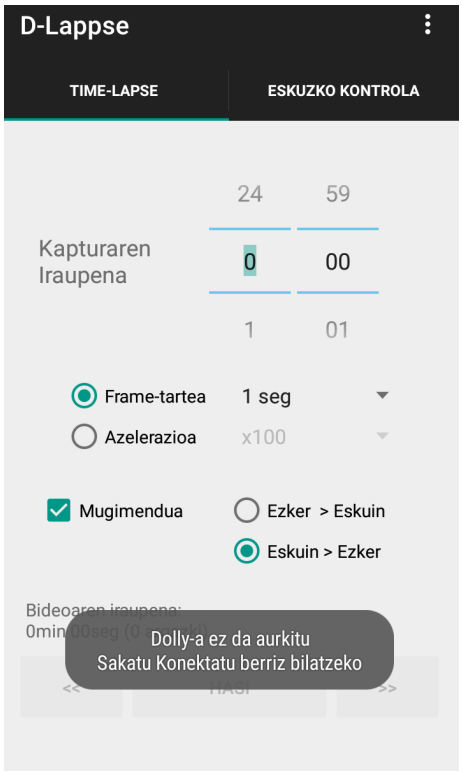
- Konexioa etetea: kasu hau arrazoi desberdinengatik gerta daiteke; Arduino aplikazioak erroreren bat izatea, Android gailua dollytik urrunegi egotea... Baina kasu guztietan egin beharrekoa berdina da, erabiltzaileari konexioa galdu dela adierazi eta botoi guztiak desaktibatzea (ikus 8.2 irudia). Aplikazioaren portaera egokia izan da kasu gehienetan, hasiera batean kontenplatu ez genuen kasu batean izan ezik. Konexioa ezarrita daukagunean eta Bluetootha desaktibatzen den kasuan ere konexioa galdu egiten da, baina beste modu batean detektatu behar da Android aplikazioan. Arazo hori konpondu beharra izan dugu Bluetooth moduluaren aldaketak detektatuz.

- Konexioa berreskuratza: honetarako, konexioa galdu dugu apropos eta ondoren konektatu botoia sakatu dugu bilaketa hasteko. Kasu guztietan ondo funtzionatu du eta arazorik gabe konexioa berriro ere jarri ahal izan dugu.

- Dollya ez aurkitzea: D-Lappse aplikazioak automatikoki egiten du dollyaren bilaketa eta aurkitzen ez baldin badu, ezin da konexiorik ezarri. Kasu honetan, erabiltzaileari abisatzen zaio ezin izan dela dollya aurkitu (8.3 irudia).



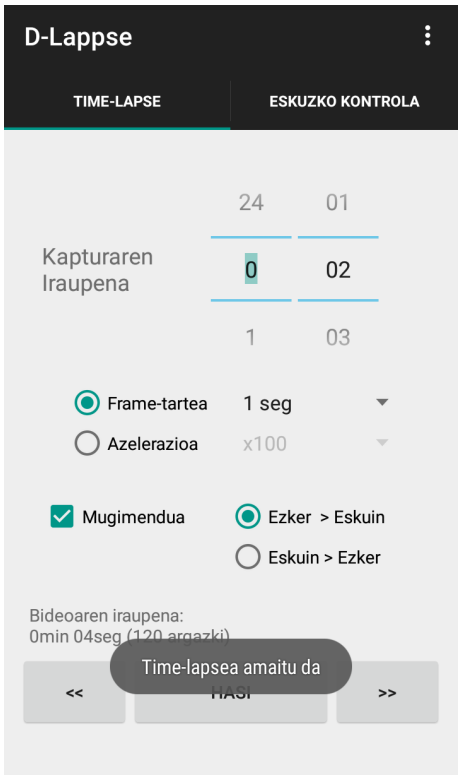
8.2 Irudia: *Konexioa galdu da* abisua.



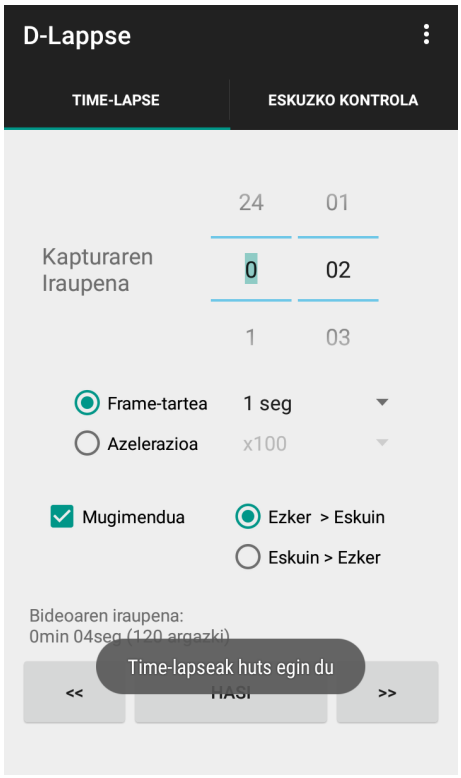
8.3 Irudia: *Dollya ez da aurkitu* abisua.

Arduino aplikazioak detektatzen dituen gertaeren aurrean ere erreakzionatu behar du Android aplikazioak. Atal hau guztiz kritikoa da erabiltzailearentzat, denbora guztian sistemaren kontrola edukitzearen inpresioa izatea nahi baitugu. Horretarako, Arduino aplikazioak mezuak bidaltzen ditu Android aplikaziora eta mezu horien bidez, erabiltzaileari abisuak emango zaizkio. Abisu garrantzitsuenetakoa time-lapsearen amaieraz abisatzen duena da, eta hau arazorik gabe egiten da. Erabiltzaileari mezu bat erakusten zaio time-lapsea amaitu dela esanez eta botoi guztiak aktibatzen dira (8.4 irudia). Dollya ertz bateraino joateko agindua bidaltzean, botoien portaera ere aldatu egiten da, time-lapsea hasteko botoia gelditzeko botoia bilakatzen da, eta ertz bateraino joateko sakatutako botoia desgaitu egiten da. Ondoren, dollya ertz horretaraino iristen denean, dollya gelditu eta Android aplikazioko botoiak bere hasierako egoerara itzultzen dira arazorik gabe.

Hasiera batean aurreikusi ez genuen egoera batek, ordea, arazoak eman dizkigu. Egoera hau dollya ertz batean kokatzen denean eta time-lapsea alde horretara egiteko agintzen dugunean gertatzen da. Dollya ez da mugitzen, Arduino aplikazioak hori bermatzen baitu, baina ez genuen aurreikusia hori gertatzean Android aplikazioak ere abisatu beharra duela. Horretarako, erabiltzaileari abisu bat jartzen zaio eta time-lapsea ezeztatu egiten da, botoi guztiak berriro ere aktibatuz, 8.5 irudian ikusten den moduan.



8.4 Irudia: *Time-lapsea amaitu da* abisua.



8.5 Irudia: *Time-lapseak huts egin du* abisua.

Android aplikazioarekin jarraituz, aplikazioaren ezarpenak behar bezala aplikatzen direla frogatu beharra daukagu, horretarako, ezarpen-konfigurazio ezberdinak aukeratu eta probak egin ditugu. Balio horietako batzuk Android aplikazioak berak erabiltzen ditu, eta horiek zuzenean aplikatzen direnez, erraz ikusten dira ondo ala gaizki aplikatzen diren; beste balio batzuk, berriz, Arduino aplikaziora bidaltzen dira. Behar diren balioak bidaltzen direla frogatzeko, Arduino aplikazioan serie-lerroaren bidez egin dugu arazketa eta jasotako balioak behar direnak direla ikusi. Balio gehienak arazorik gabe iristen dira, salbuespen batekin: frame-tartea balioak bi zifra ditu komaren ondoren eta hori ez zen behar bezala jasotzen Arduino aplikazioan. Arazoaren arrazoia identifikatzea dezente kostatu zaigu, baina azkenean lortu dugu. Zenbakiaren balio osoa eta hamartarra zatitzeko erabiltzen den komatxoan datza. Android aplikazioak koma bat erabiltzen du, (ziurrenik erabiltzen den mugikorraren lokalizazioaren eraginez) eta Arduino aplikazioak, berriz, puntua. Koma hori puntu batez ordezkatuz konpondu dugu arazoa.

Ezarpenen artean garrantzi berezia duena hizkuntzaren aukeraketa da. Hainbat proba egin dugu, Android gailuaren hizkuntza aldatu dugu eta aplikazioa hizkuntza horretan exekutatzen zela probatu dugu lehenengo; ondoren, ezarpenetatik hizkuntza aldatu eta arazorik gabe funtzionatzen duela ikusi dugu. Aplikazioa ezabatzen ez badugu behintzat hizkun-

tzaren aukeraketa mantentzen du. Proba hauek egiterakoan, aplikazioaren mezu batzuk itzuli gabe genituela ohartu gara; horiek ere itzuli ditugu eta orain aplikazioa hiru hizkuntzatan dago oso-osorik.

9. KAPITULUA

Jarraipena eta kontrola

Kapitulu honetan proiektuaren garapenean izan ditugun gorabeherak azaltzen dira, proiektuaren garapenean izandako desbideraketak, kalitatearen kontrola eta baita identifikatutako arriskuek izan duten eragina ere.

9.1 Proiektuaren garapena

Proiektuaren garapena hasiera batean definitutakoarekin bat eginez eramaten saiatu gara; gauza batzuk egitea uste baino gutxiago kostatu zaigu, eta beste batzuk, berriz, uste baino gehiago.

Lehenengo amaitu genuen atala dollyaren eraikuntza izan zen, Arduinoaren osagai guztiak bere lekuan jarri eta probak egiten hasteko ezinbestekoa baikenuen. Hasiera batean honekin hastea kostatu bazitzaigun ere, jaso dugun laguntzari esker lortu genuen dollya muntatzea gorabehera gehiegirik gabe. Ondoren, Arduinoa eta bere osagaiak jarri genizkion dollyari eta probak egiten hasi ginen osagai bakoitzaren kontrola lortzeko. Osagaiak kontrolatzean izandako arazo nagusia motorrak eman zigun; antza denez, arduinoaren *Motor Shieldak* ez zuen nahikoa indarra eta erabilpen luzea ematen bagenion, exekuzioa eten egiten zen. Hori konpontzeko, beste motor bat erabiltzea erabaki genuen: urratsez urratseko motor bat. Proba hauek egitearekin batera, egindako ikerketa guztiz baliagarria suertatu zaigu; motorraren aukeraketa egiteko, edota baita kameraren kontrola egiteko ere. Kameraren kontrola interbalometro bat erabiliz egitea pentsatu genuen hasiera batean, baina kameraren kontrola egiteko aukerak aztertzen ari ginenean, beste modu bat

aurkitu genuen egokiagoa iruditu zitzaiguna: IR diodo bat erabiliz egitea. Gainera, kamera modu horretan kontrolatzeko Arduinorentzat bereziki egindako liburutegi bat topatu genuen; beraz, kameraren kontrola modu honetan egingo genuela erabaki genuen.

Hardwarearen garapena uste baino lehenago amaitzea lortu genuen eta denborarik galdu gabe Arduino aplikazioarekin hasi ginen. Arduinorentzat programatzen hastea uste baino askoz errazagoa egin zaigu; ikasketarako estimatu genuen denbora asko murriztu zen eta ikerketa egindakoan lortutako kode-zatiak ere oso baliagarriak izan dira. Atal honetarako benetan baliagarriak izan dira Arduino komunitate irekiak eskaintzen dizkigun abantailak: sarean informazio eta kode ugari dago eta ia guztia erabilera librekoa da. Arduino aplikazioaren garapenak uste baina denbora gutxiago eraman badigu ere, are gutxiago izango zen motorrarekin izandako arazoak ez bagenitu izan. Izan ere, motorraren aldaketak, motorraren kontrol konplexuagoa (eta zehatzagoa) ekarri baitigu. Ez genuen pausokako motor bat kontrolatzeko metodo erraz edota liburutegirik aurkitu, beraz, guk egin behar izan genuen motorraren kontrolerako liburutegi bat. Arduino aplikazioari forma pixka bat ematen hasi ginenean, Android aplikazioaren garapenarekin hasi ginen, bi aplikazioen funtzioak eskutik helduta baitoaz.

Arduinorentzat programatzen erabilitako denbora urriak Androiden programatzen ikasten erabili dugunarekin kontrastatzen du, askoz denbora gehiago behar izan baitugu. Hala ere, uste baina lehenago hasi ahal izan gara aplikazioa garatzen, hardwarearekin eta Arduinoaren softwarearekin izandako aurrerapenaren ondorioz. Garatzen hasi eta lehenengotako helburua Bluetooth konexioa lortzea izan zen; modu horretan, Arduinoari mezuak bidaltzeko aukera izango genuen eta bi aplikazioen funtzionalitateak paraleloan garatzen joan. Lehenengo, funtzionalitate sinpleenak garatu genituen, alde batera mugitu edota gelditu bezalakoak. Funtzionalitate sinple horiek ondo dominatu genituenean, time-lapsea egiteko funtzionalitatea garatzen jarri ginen. Alde batetik, Android aplikazioan parametro asko zeuden erabiltzaileak aukeratzeko eta Arduino aplikazioak jasotako informazioa modu egokian interpretatu eta aurrera eraman beharra zeukan. Nahiz eta funtzionalitate konplexua izan, nahiko erraz lortu genuen funtzionaraztea, aurretik ondo landu baikenituen funtzionalitate sinpleagoak eta sistema ondo ezagutzen genuen. Beraz, lehenengo probak egiten nahiko azkar hasi ginen.

Sistemaren funtzionalitate nagusiak garatuta genituela, proba dezente egiten hasi ginen. Proba horietan, zenbait arazo detektatu genituen: lehenengo urratsa arazoa non zegoen detektatzea izaten zen eta, ondoren, konpontzea; arazo batzuek Android aplikazioarekin zerikusia zuten eta besteek, berriz, Arduino aplikazioarekin. Pixkanaka oinarrizko funtzionalitateen arazoekin amaitzen joan ginen eta Android aplikazioari heldu genion, apli-

kazioaren optimizazioan eta funtzio osagarrietan zentratzeko.

Sistemaren erabilgarritasunean eta erabiltzailearentzat ahalik eta intuitiboena bilakatzen, denbora gehien inbertitutako atala izan da. Android aplikazioan ez ezik, Arduino aplikazioan eta komunikazio protokoloan ere aldaketak egin behar izan dira, eta gauza asko konpontzeko sarera jo behar izan dugu informazio eske. Atal honek ere proba ugari egitea eskatu zigun, egoera eta aldagai asko baitaude kontuan hartzeko, eta den-denak egiaztatu nahi izan genituen.

Azkenik, aplikazioaren funtzio osagarriak gehitu ziren. Horien artean, garrantzitsuena hizkuntzaren aukeraketa izan zen, eta, hasiera-hasieratik kontuan izan badugu ere, bere lana eskatu zigun. Alde batetik, aplikazioaren testu guztiak identifikatu genituen, bat bere ere ahaztu gabe, eta ondoren, testu horiek beste hizkuntzetara itzuli. Testu guztiek hizkuntza guztietan ez dute berdina okupatzen; beraz, kontuan hartu behar izan genuen, interfazean eragina izango baitzuen. Horretaz gain, hizkuntza aukeratzean, aldaketa horiek aurrera eramatea ez da dirudiena bezain sinplea. Modurik errazena aplikazioa berrabiaraztea izango litzateke, baina hori ez da erabiltzaile baten ikuspuntutik oso erosoa. Beraz, hori ekiditeko moduak bilatu behar izan genituen ,eta, azkenean, hizkuntza aukeratzen den une berean aplikazio guztia hizkuntza horretan agertzea lortu genuen.

9.1.1 Planarekiko desbideraketak

Proiektuaren garapenean, hasierako plangintzatik alderatuz, desbideraketa batzuk izan dira, aurreko atalean azaldu dugun bezala, baina desbideraketa garrantzitsuenen arrazoiak azalduko ditugu jarraian. 9.1 irudian ikus daitezke ataza bakoitzean izandako desbiderapenak.

Garapena

Nahiz eta guztira nahiko desbiderapen txikia izan garapenean, atazez ataza begiratzen badugu, desbiderapen nahiko handiak gertatu direla ikusi daiteke. Alde batetik, hardwarearen garapenak eta Arduinoaren softwareak uste baina denbora gutxiago eraman zigun, baino Android aplikazioa garatzeak planifikatutakoa baino % 29 gehiago. Android garapena dedikazio handienetakoa duen ataza izanik, desbideraketa nahiko handia da. Honen arrazoi nagusia, Android aplikazioa hobetzen inbertitutako denbora oso handia izan dela da; xehetasun batzuk gainditzea eta aplikazioa nahi genuen kalitate mailara igotzea lan askoren fruitua izan da eta, horretarako, ordu asko sartu behar izan ditugu.

Hala ere, kontuan hartu behar da atal hori garapenaren amaieran egin dugula eta hardwa-

rearen eta Arduino aplikazioaren garapenetan izandako denbora-aurrezteak bagenekizkiela; honenbestez, denbora guztian kontrolatuta genuen egutegia eta ez gara beharrezkoa baino gehiago arriskatu softwarearen garapenera dedikatutako ordu kopurua handitzean.

Memoria

Atal honek ere desbiderapen oso handia izan du; hasiera batean estimatutako denbora asko zen, baina are gehiago behar izan dugu proiektuaren memoria idazteko. Desbiderapenaren arrazoi nagusia esperientzia falta da; proiektu nahiko handia da ohituta gaudenerako, eta hori guztia dokumentatu eta azaltzera ez gaude nahikoa trebatuak. Memoria idazten ari ginen momentuan ohartu gara planifikatu genuena baino denbora gehiago beharko genuela, baina hori egitea erabaki dugu. Azken batean, dokumentu hau da proiektuan egindako guztiaren laburpena eta nahikoa garrantzia duela iruditu zaigu dedikazio gehiago eskaintzeko.

9.2 Kalitatea

Gure proiektuarekin lortu dugun kalitatea aztertuko dugu atal honetan. Jarraian, 2.7.1 atalean definitutako kalitate-adierazleak aztertu ditugu.

Adierazle kuantitatiboak

- Plangintzaren jarraipena: nahiz eta ez dugun guztiz jarraitu finkatutako plangintza, orokorrean bete dugula esan genezake. Plangintza errespetatzearen puntu garrantzitsuena epeak betetzea dela esan genezake, eta hori, dudarik gabe bete dugu. Proiektuaren hasieratik planifikatutakoa baino pixka bat aurreratuago joan gara eta hori aprobetxatu dugu ezusteko garrantzitsurik izaten bagenuen ere. Estimatutako dedikazioak ez dira guztiz bete, kasu batzuetan dezente desbideratu baikara planifikatutako denboretatik. Hala ere, amaieran dedikazioaren desbiderapena %6koa izan da eta proiektuaren dimentsioak ikusita, nahiko desbiderapen onargarria dela iruditzen zaigu.
- Kodearen lerro-kopurua: garapen fasean etengabe kode lerroen kopurua hazten joatea adierazle interesgarritzat jo genuen, eta hala izan zen, hasiera batean behintzat. Programatzen hasi eta lehenengo egunetan kode kopurua etengabe eta azkar hazi zen, aplikazioa hazten eta aurreratzen zihoan seinale, baina garapenaren fase aurreratuetan ez zen adierazle esanguratsua izan, kode laburra baina asko pentsatutakoa

idazten baikenuen. Beraz, garapenaren azken faseetan ez genion jaramon handirik egin adierazle honi.

Adierazle kualitatiboak

- Sistemarekiko elkarrekintza: sistemarekin elkarrekintza asko zaindu den atala izan da, eta horregatik, kalitate maila on bat bermatzeko proba asko egin dira. Android aplikazioa nahiko osatuta genuenean, probak egiten hasi ginen eta ikusten genituen arazoak konpondu. Ondoren, arazorik ikusten ez genionean, zenbait erabiltzailerekin probatu da aplikazioa, guk azalpenik eman gabe erabiltzeko gai ote ziren ikusteko. Ez da arazorik egon eta erabiltzaile guztiek sistema martxan jartzea lortu dute aparteko arazorik gabe.
- Sistemaren fidagarritasuna: sistemaren fidagarritasuna baloratzeko, probak egitea ezinbestekoa da. Probak egiten hasi eta berehala konturatu ginen fidagarritasunaren hutsegiteak ondorio larriak eragin ziezazkiokeela hardwareari; beraz, fidagarritasuna ezinbesteko atala izan da hasiera hasieratik. Egin ditugun azkeneko probetan ez da fidagarritasunaren hutsegiterik detektatu, ez hardware ez eta softwarearen aldetik, eta beraz, sistema nahiko fidagarri bat garatu dugula iruditzen zaigu.
- Kodearen argitasuna: honakoa baloratzea nahiko erlatiboa izan daiteke, pertsona batzuek segituan ulertzen duten kode bat beste batzuentzat ulergaitza izan bailiteke. Hala ere, gure kasuan esfortzu nabarmena egin dugu edonork ulertzeko kodea idazten. Kodea iruzkin eta oharrez betea dago, ez gehiegi, baina atal bakoitzak zer egiten duen ulertzeko beharrezko azalpenak ematen dira. Horretaz gain, metodoek eta aldagaiek dituzten izenak ere esanguratsuak dira eta kasu askotan komentarioak ordezkatzen dituzte. Kodea zuzenean irakurrita ulertzeko argitasunaz gain, arazketa prozesua ere kontuan hartu dugu. Horretarako, kodean *log* komando asko ageri dira, horiek denak arazketari begira informazio baliagarria ematen dutenak. 9.2 irudian ikus daiteke Android aplikazioaren arazketa nola bistaratzen den Android Studio garapen ingurunean.

9.3 Arriskuak

2.8 atalean proiektuaren arriskuak identifikatu genituen. Arriskuak garapenean izandako eragina aztertuko dugu segidan.

- **Datuen galera:** ezarri genituen jarraibideak jarraitu dira proiektuaren garapenean, eta zorionez, ez dugu datu galerarik jasan. Honenbestez, inork pentsatu lezake segurtasun-kopiak egiten erabilitako denbora denbora galdua izan dela, baina inondik ere. Segurtasun-kopiak egitearen ohitura hartzean, oso denbora gutxi eramaten du eta, batez ere, sarean egindako segurtasun kopiak, gure kasuan OneDrive zerbitzuan, benetan lasaitasun handia eskaintzen du. Edozein momentutan eta edonon gaudela ere informazio hori atzitzeko aukera ematen digu ,eta, datuen galera babesteaz gain, erosotasun handia eskaintzen du sistema desberdinetatik informazio bera atzitzeko aukerak.

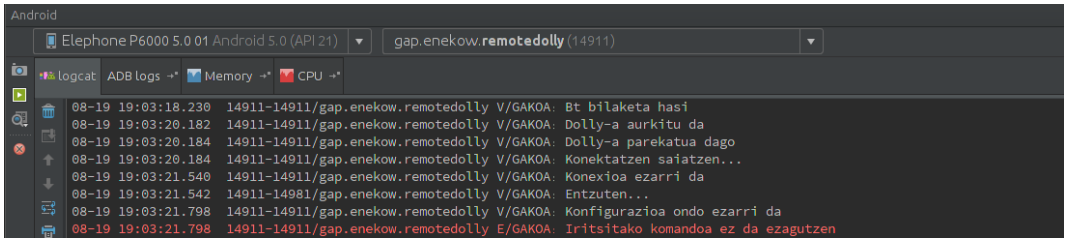
- **Hardware-ingurunea ezin prestatzea:** hasiera batean gure kezka nagusietakoa bazen ere, azkenean ez da arazo larria bilakatu. Nahiz eta arazo txiki eta buruhauste bat baino gehiago ekarri dizkigun, jaso dugun laguntzari esker garapena martxa onean joan da eta arazorik gabe amaitu.

- **Aplikazioaren garapeneko arazoak:** atal honetan Git aplikazioaren erabilera benetan kritikoa izan da. Android garapena ezagutzen ez genuen mundua zen eta garapena egiterako garaian arazo asko izan ditugu. Horien artean, denbora luzean trabatzen gaituztenetako arazo bat errorea non egin dugun aurkitzen ez dugunean gertatu ohi da. Git zerbitzuari esker, aplikazioaren bertsio ezberdinak gordetzen joan gara garapenean zehar eta trabatuta geratu garen kasu batean baino gehiagotan, atzera bota behar izan ditugu aldaketak eta aplikazioaren aurreko bertsio batera itzuli.

- **Plangintzari ez jarraitzea:** planifikatutakoari guztiz jarraitu ez badiogu ere, denbora guztian kontuan izan dugu plangintzarekiko generaman desbideraketa. Beraz, esan genezake plangintza nahiko erabilgarria suertatu zaigula, eta batez ere, hartutako erabaki batzuk benetan garrantzitsuak. Horien artean, dollyaren garapenarekin proiektuaren ia hasieratik hasteak malgutasun handia eskaini digu eta egutegia errespetatzeko guztiz erabakigarria izan da. Software garapenarekin hasi aurretik ikerketara dedikatutako denbora guztia ere benetan baliagarria izan da, denbora asko aurrezteko balio izan digun informazioa biltzea lortu baitugu.

Kodea	Atazaren deskribapena	Estimazioa	Erreala	Desbiderapena	
1	Kudeaketa	33:00	31:00	-2:00	-%6
1.1	Plangintza	14:00	14:00	0:00	%0
	Egutegia definitu	12:00	12:00	0:00	%0
	Arriskuen kudeaketa plana	2:00	2:00	0:00	%0
1.2	Jarraipena	6:00	5:00	-1:00	-%17
	Bilerak	4:00	3:00	-1:00	-%25
	Plangintzaren aldaketak	2:00	2:00	0:00	%0
1.3	Kontrola	13:00	12:00	-1:00	-%8
	Arriskuak eguneratu	3:00	2:30	-0:30	-%17
	Kalitatea kontrolatu	5:00	6:00	1:00	%20
	Aldaketak kudeatu	5:00	3:30	-1:30	-%30
2	Ikerkuntza	46:00	46:00	0:00	%0
2.1	Ikasketa	36:00	36:00	0:00	%0
2.2	Material bilaketa	10:00	10:00	0:00	%0
3	Arkitektura	35:00	29:00	-6:00	-%17
3.1	Analisia	15:00	13:00	-2:00	-%13
3.2	Diseinua	20:00	16:00	-4:00	-%20
4	Garapena	139:00	148:00	9:00	%6
4.1	Hardwarea	30:00	23:00	-7:00	-%23
	Dollya muntatu	20:00	16:00	-4:00	-%20
	Arduino eta gainerako osagarriak muntatu	10:00	7:00	-3:00	-%30
4.2	Softwarea	103:00	117:00	14:00	%14
	Lan ingurunea prestatu	3:00	2:00	-1:00	-%33
	Arduino aplikazioa programatu	30:00	25:00	-5:00	-%17
	Android aplikazioa programatu	70:00	90:00	20:00	%29
4.3	Probak	6:00	8:00	2:00	%33
5	Itxiera	90:00	110:00	20:00	%22
5.1	Memoria	80:00	100:00	20:00	%25
5.2	Defentsa	10:00	10:00	0:00	%0
		343:00	**364:00**	**21:00**	%6

9.1 Irudia: Hasierako planarekiko izandako desbiderapenak.



9.2 Irudia: Android aplikazioaren arazketa.

10. KAPITULUA

Ondorioak

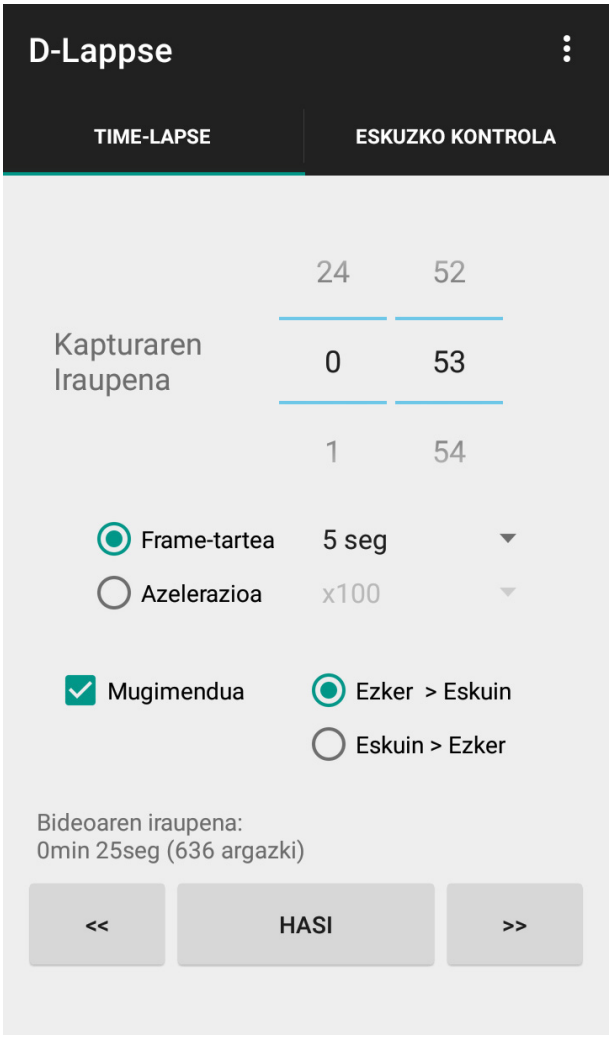
Kapitulu honetan proiektuarekin lortutako emaitzak eta ateratako ondorioak azalduko ditugu, eta azkenik, egindako lanarekin aurrera jarraituko bagenu zein aukera edukiko genituzkeen aztertuko dugu.

10.1 Lortutako emaitza

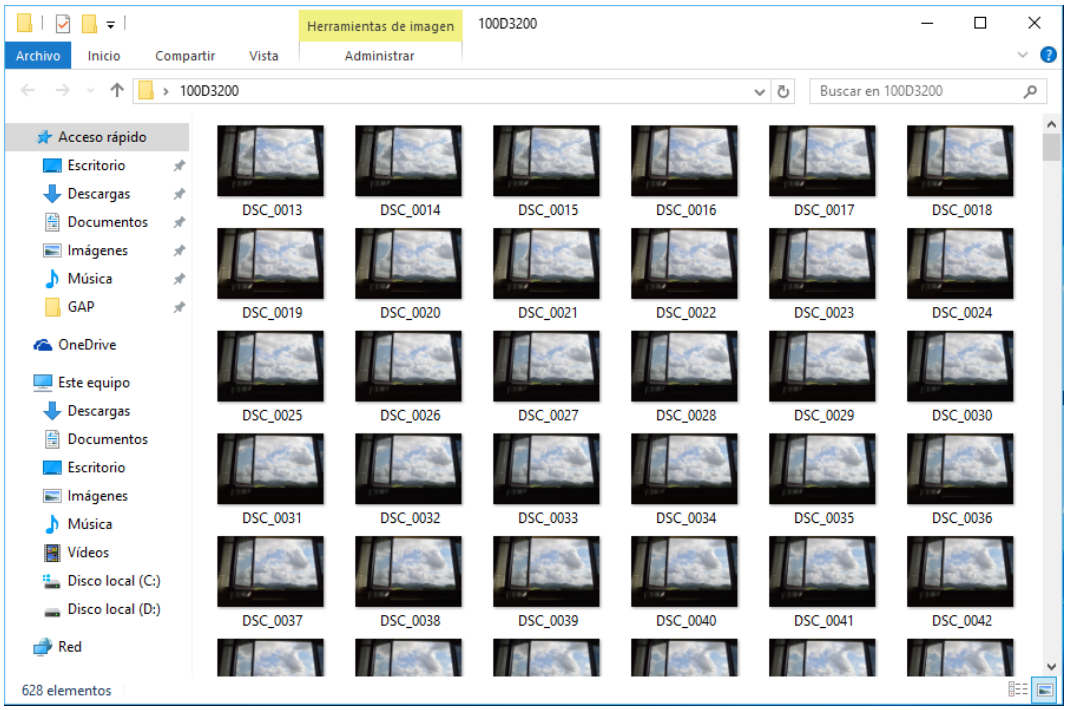
Time-lapseak egiteko sistema automatizatua garatzea lortu dugu, eta nahiko lan fina egin dugulakoan gaude gainera. D-Lappse Android aplikazioa garatu dugu, eta aplikazioarekin komunikatu eta bidalitako aginduak aurrera eramateko gai den dollya ere bai. Sistemaren helburua time-lapseak egitea da eta horixe da garatutako proiektuaren azken emaitza.

D-Lappse aplikazioan aukeratzen dira time-lapseak izan behar dituen ezaugarriak: zenbat argazki aterako diren, zenbateko tartearekin, etab. Time-lapsea egiteko agindua bidalitakoan, dollya bera arduratuko da ekintza hori aurrera eramateaz. Sistemak sortutako time-lapse bat erakusteko asmoz, honako proba praktiko hau egin dugu:

> D-Lappse aplikazioan 25 segundoko time-lapse bat egiteko parametroak ezarri ditugu, 10.1 irudian ikus daitekeen moduan.
>
> Dollya martxan jarri eta ezarritako parametroei jarraituz argazkiak egin dira. Time-lapsea amaitutakoan argazkiak kameraren memoria txartelean aurkitzen dira (10.2 irudian ikus daitezke ateratako argazki batzuk).



10.1 Irudia: 5 segundoko tartearekin 25 segundoko time-lapsea egiteko parametroak.



10.2 Irudia: Time-lapsea egitean ateratako argazkiak.

Irudia aztertuz ohar gaitezke, kamerak argazkiak atera dituela, baina ez du zehatz mehatz guk esandako argazki kopurua atera, argazki batzuk gutxiago atera ditu. Honen arrazoia motorraren zehaztasunean dago, ez baitu zehazki mugitu beharrekoa mugitzen, baina 636 argazkitik 8 argazkiko hutsegitea oso txikia da. Azken emaitzan 0.32 segundo gutxiagoko iraupena izango du 25 segundoko time-lapseak.

Ateratako argazkiak hartu eta time-lapse bideoa sortu dugu. Hurrengo estekan ikusi daiteke proba hau egiterakoan sortutako time-lapsea.

`https://vimeo.com/dlappse/leihotik`

Lortutako emaitzarekin benetan gustura gaude. Gure sistemak ez ditu sistema profesionalek dituzten mugimendu eta aukera guztiak, baina mugimendu natural eta suabeak egiten ditu, eta garrantzitsuena: erabiltzaile ez-aditu batek ere arazorik gabe kontrolatu dezake sistema. `https://vimeo.com/dlappse` helbidean egindako time-lapse gehiago ikus daitezke.

10.2 Ebaluazio pertsonala eta ikasitako lezioak

Proiektu honetan gradu osoan zehar eskuratutako gaitasunak frogan jarri ditugu. Ez soilik diseinuan edota programazioan ikasitakoa, baizik eta edozein proiektu mota aurrera eramaterakoan gerta litezkeen arazoei ere aurre egin behar izan diegu. Proiektua proposatzean bagenekien buruhauste asko eman ziezagukeen proiektu bat zela honakoa, baina gure burua erronkari aurre egiteko prest ikusten genuen. Azkenean, nahiz eta motibazio gutxiko momentuak ere igaro, proiektua aurrera ateratzea lortu dugu eta oso gustura gaude egindako lanarekin, nahiz eta oraindik badauden xehetasun batzuk hobetzeko.

Proiektuaren bizi-zikloan zehar badira ikasi ditugun zenbait gauza, horietako batzuk dagoeneko bagenekizkienak, baina orain arte beren garrantziaz ohartu gabeak gehienak. Segidan azalduko ditugu proiektu honetan zehar ikasi ditugun eta interesgarriak izan litezkeen zenbait lezio.

- Aste gutxitan ordu asko sartu beharra izan dugu proiektuan zehar eta une askotan lanak gain hartzen gaituen sentsazioa izan dugu. Hori ekiditeko eta motibazioa altu mantentzeko, egunero helburu batzuk finkatzea ideia ona da. Ez dira oso helburu konplexuak izan behar, egun horretan bertan amaitzeko astia emango dizkigun atazak baizik. Helburu horiek betetakoan, helburu gehiago finkatu ditzakegu edota hurrengo egunean jarraitzeko utzi.

- Kodea bilatzea denbora asko aurrezten lagunduko digun eginkizuna da. Informatikari garen heinean, dagoeneko bagenekien hori, baina garrantzia berezia hartzen du Android eta Arduino sistementzat garatzen dugunean. Android eta Arduino izaera askeko plataformak izanik, beren inguruan sortu den garatzaile-komunitateak ere izaera hori dauka oro har. Kode asko dago sarean eta gauza gehienak asmatuta daude; beraz, denbora-tarte bat zehazki kode-bilaketari eskaintzea ez da ahaztu beharreko gauza.

- Sareko zerbitzuetan egindako segurtasun-kopiak benetan baliagarriak dira. Ez soilik segurtasun maila altua eskaintzen duten zerbitzuak izan ohi direlako, baizik eta edozein momentutan edonon informazio guztia eskura edukitzea oso baliagarria suertatu daitekeelako. Batez ere, espero ez den uneren batean informazio hori eskuragarri ez edukitzeak suertatu daitekeen aukeraren bat ez aprobetxatzea ekar lezakeelako.

- Denboraren kudeaketa egunera eramateak duen garrantziaz ere ohartu gara. Ez soilik denborak errespetatu edota desbideraketak ikusteko, baizik eta denbora aurreztu

bada momenturen batean, denbora hori beste zerbaitetan erabiltzeko aukera dugulako. Gure kasuan, hardwarea eraikitzeko eta Arduino programazioa ikasteko pentsatutakoa baino denbora gutxiago behar izan dugu eta aurreztutako denbora hori beste gauza batzuetara dedikatu dugu. Modu honetan D-Lappse aplikazioak funtzionalitate osagarri gehiago izatea lortu dugu.

- Proiektuaren bizi-zikloa nahiko luzea izan da, eta hasieratik gauzak dokumentatzea dirudiena baina garrantzitsuagoa da, gauzak behar bezala dokumentatzen ez badira ahaztu edo galdu egin daitezke eta. Hasieratik saiatu gara dena dokumentatuta eramaten, eta bai proiektuaren garapen-prozesuan eta bai memoria idazterako garaian oso baliagarria izan zaigu hau. Hala ere, zer gauza dokumentatzen diren ondo pentsatu beharra dago, ondoren baliagarriak izango diren gauzak izan behar dute, dokumentatzen duguna erabilgarria ez bada denbora-galera suposatuko baitu.

- Proiektu honek duen dimentsioko memoriarik ez dugu inoiz garatu eta esperientzia falta hori nabaritu dugu. Ikasitako gauza garrantzitsu bat hasieratik egitura bat zehaztea da. Hau da, nolakoa izango den dokumentuaren hezurdura guztia. Ez badugu garbi jarraitu behar dugun egitura, ondoren, idatzitakoa alde batetik bestera mugitzen denbora asko galduko baitugu.

10.3 Etorkizuneko aukerak

Sortu dugun sistemak aukera dezente eskaintzen baditu ere, oraindik ere badira hobetzeko aspektuak. Horietako batzuk nahiko zeregin sinpleak dira, baina denbora kontuak direla eta ezin izan ditugu inplementatu. Beste aukera batzuk, oraingoz, gure irismenetik kanpo dauden ideiak dira, baina etorkizunerako aukera interesgarriak izan daitezke.

- D-Lappse aplikazioan laguntza eskaintzea: ez da ataza konplexua printzipioz, baina denbora eskaini behar zaio. Laguntza hori menuko beste elementu bat bezala erakutsi daiteke eta hori sakatzean beste pantaila batera eramango gintuzke. Pantaila horretan, sistema martxan jartzeko gida azkar bat erakutsiko litzateke, erabileragida bat izango balitz bezala.

- Time-lapsearen mugimendu mugatua: honako aukera hau egiteko asmoa geneukan, baina ez dugu denborarik izan honetarako ere.

- Time-lapsea amaitzean abisua: nahiz eta oraingo D-Lappse aplikazioak abisua ematen duen time-lapsea amaitzean, pantailari begira egon behar dugu abisu hori ikusteko. Horren ordez, soinu bidezko abisu bat emango balitz praktikoagoa izango litzateke.

- Kameraren kontrol handiagoa: gure sistemak kamera kontrolatzen du, baina argazkiak ateratzeko agindua soilik bidali dezakegu. Interesgarria izango litzateke kameraren beste ezaugarriak ere kontrolatu ahal izatea aplikaziotik: besteak beste, argazkiak ikusteko aukera edota bideoa grabatzeko aukera. Zoritxarrez, oraingoz, Arduino kontrolatzaile batekin hori egiterik ez dago, sistema eragile garatuago bat behar baita aukera horiek guztiak kontrolatzeko.

- Dollyan hobekuntzak: gure dollya prototipo bat da, eta nahiko aukera mugatuak ditu. Dollyak hobekuntzak jasango balitu, adibidez kamerarentzat beste errotazio-puntu bat, D-Lappse aplikaziotik hori aprobetxatzeko aukera izango genuke eta sistema osatuagoa lortu.

- Produktu komertzial gisa kokatzea: gaur egun oso modan dauden *crowdfunding*[1] webguneetan arrakasta izango lukeen produktu bat da D-Lappse gure ustez. Abantaila nagusi bat daukagu: D-Lappse app-a dagoeneko egina daukagu eta hardwarearekiko independentea da, hau da, komunikazio-protokoloa errespetatzen duen edozein hardwarerekin funtzionatzeko pentsatua dago. Produktu komertzial bat bilakatzeko, dollya eta hardwarea modu errazago eta eraginkor batean garatzeko bideak aurkitu beharko lirateke eta komunikazio-protokoloarekin bateragarri mantendu.

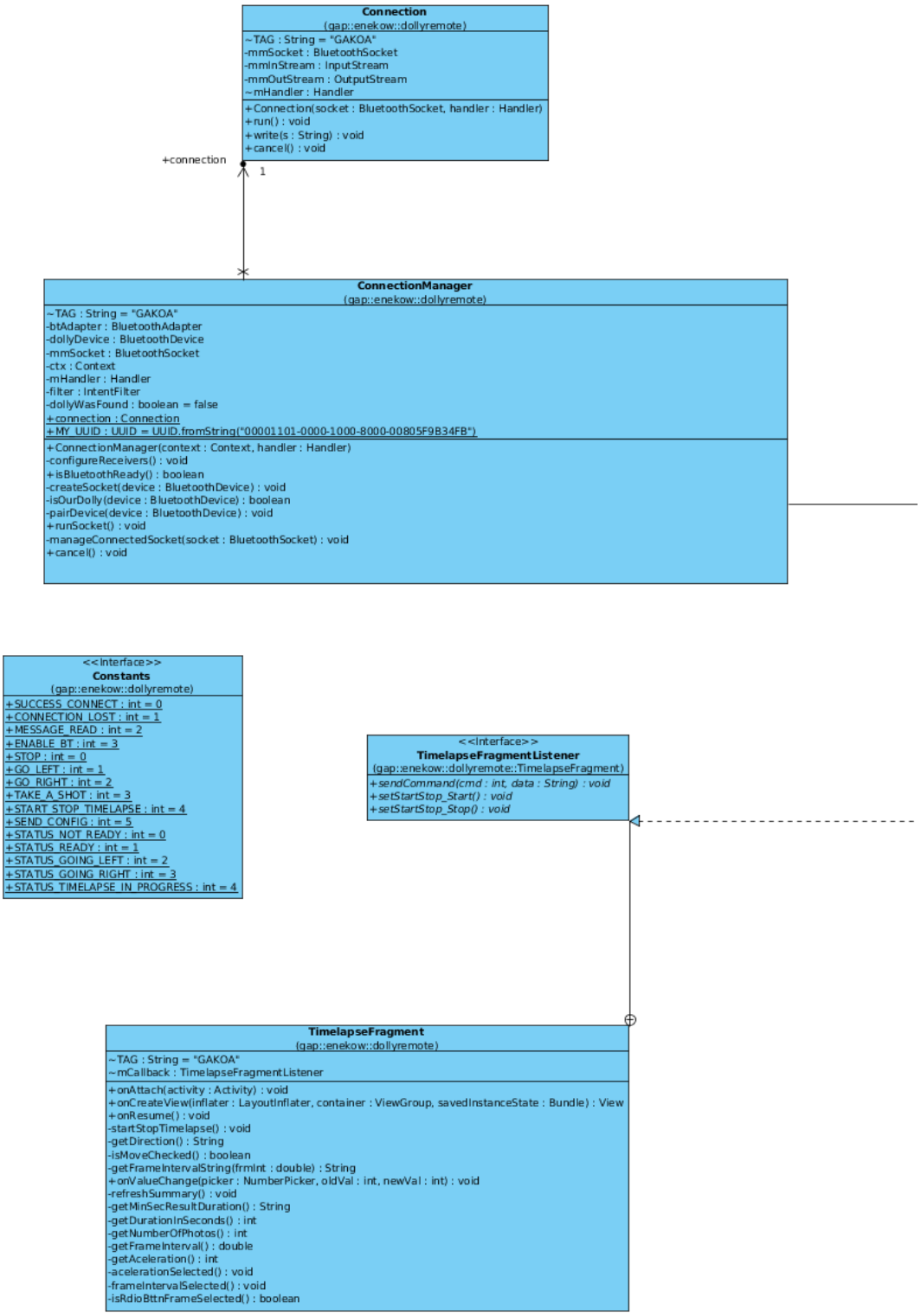
- *StepperMotor* liburutegiaren berrerabilpena: nahiz eta ez den zuzenean gure proiektuaren etorkizuneko aukera bat, Arduinorentzat sortutako *StepperMotor* liburutegia Arduinoren foro ofizialean eskegi dugu eta litekeena da, etorkizunean norbaiten Arduino proiektuaren zati izatea.

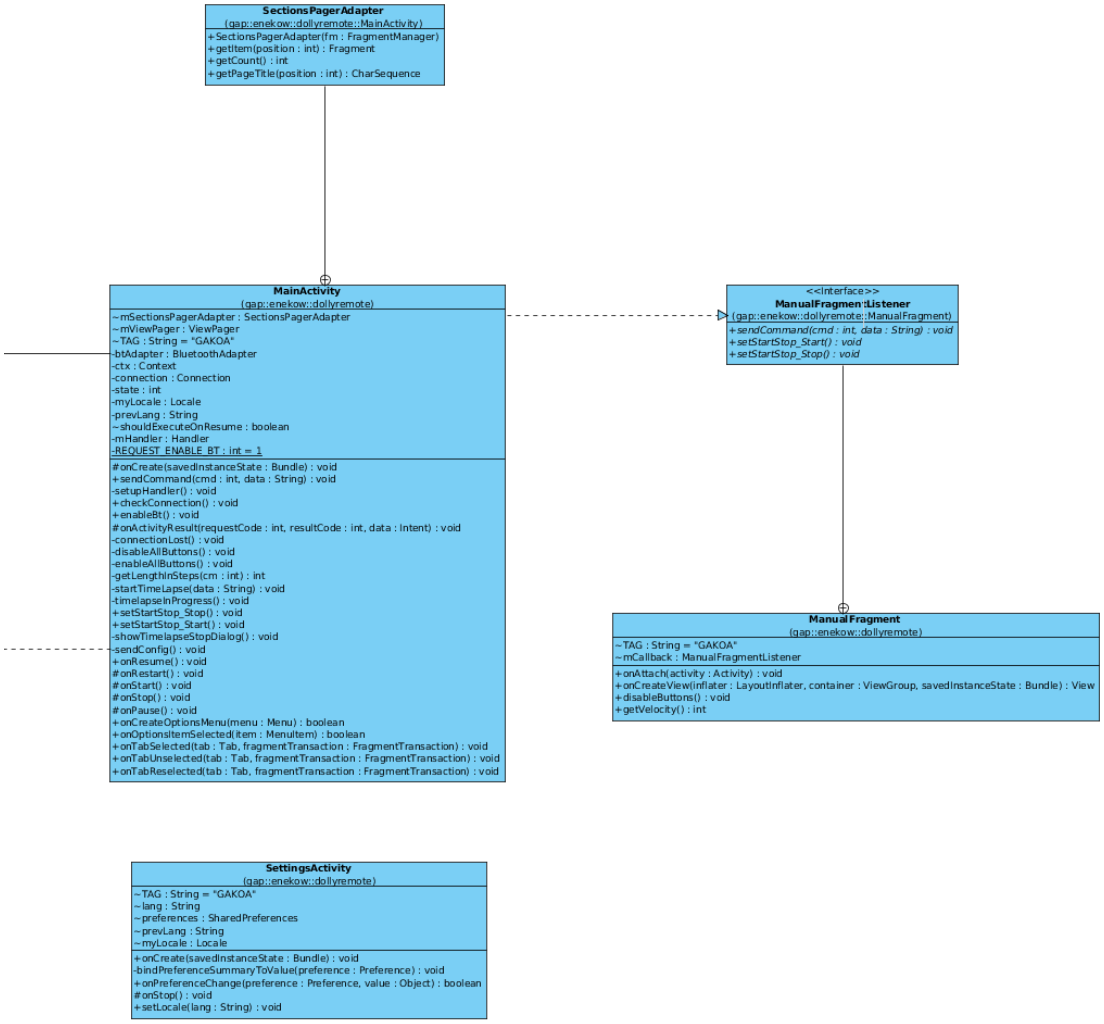
[1] *Crowdfunding* pertsona talde batek dirua edo bestelako baliabideak biltzeko egiten duen kooperazio kolektiboari deritzo.

Eranskinak

A. ERANSKINA

UML klase-diagrama


Connection
(gap::enekow::dollyremote)
~TAG : String = "GAKOA"
-mmSocket : BluetoothSocket
-mmInStream : InputStream
-mmOutStream : OutputStream
~mHandler : Handler
+Connection(socket : BluetoothSocket, handler : Handler)
+run() : void
+write(s : String) : void
+cancel() : void
+connection
1
ConnectionManager
(gap::enekow::dollyremote)
~TAG : String = "GAKOA"
-btAdapter : BluetoothAdapter
-dollyDevice : BluetoothDevice
-mmSocket : BluetoothSocket
-ctx : Context
-mHandler : Handler
-filter : IntentFilter
-dollyWasFound : boolean = false
+connection : Connection
+MY_UUID : UUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
+ConnectionManager(context : Context, handler : Handler)
-configureReceivers() : void
+isBluetoothReady() : boolean
-createSocket(device : BluetoothDevice) : void
-isOurDolly(device : BluetoothDevice) : boolean
-pairDevice(device : BluetoothDevice) : void
+runSocket() : void
-manageConnectedSocket(socket : BluetoothSocket) : void
+cancel() : void
<<Interface>>
Constants
(gap::enekow::dollyremote)
+SUCCESS_CONNECT : int = 0
+CONNECTION_LOST : int = 1
+MESSAGE_READ : int = 2
+ENABLE_BT : int = 3
+STOP : int = 0
+GO_LEFT : int = 1
+GO_RIGHT : int = 2
+TAKE_A_SHOT : int = 3
+START_STOP_TIMELAPSE : int = 4
+SEND_CONFIG : int = 5
+STATUS_NOT_READY : int = 0
+STATUS_READY : int = 1
+STATUS_GOING_LEFT : int = 2
+STATUS_GOING_RIGHT : int = 3
+STATUS_TIMELAPSE_IN_PROGRESS : int = 4
<<Interface>>
TimelapseFragmentListener
(gap::enekow::dollyremote::TimelapseFragment)
+sendCommand(cmd : int, data : String) : void
+setStartStop_Start() : void
+setStartStop_Stop() : void
TimelapseFragment
(gap::enekow::dollyremote)
~TAG : String = "GAKOA"
~mCallback : TimelapseFragmentListener
+onAttach(activity : Activity) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+onResume() : void
-startStopTimelapse() : void
-getDirection() : String
-isMoveChecked() : boolean
-getFrameIntervalString(frmInt : double) : String
+onValueChange(picker : NumberPicker, oldVal : int, newVal : int) : void
-refreshSummary() : void
-getMinSecResultDuration() : String
-getDurationInSeconds() : int
-getNumberOfPhotos() : int
-getFrameInterval() : double
-getAceleration() : int
-acelerationSelected() : void
-frameIntervalSelected() : void
-isRdioBttnFrameSelected() : boolean


SectionsPagerAdapter
(gap::enekow::dollyremote::MainActivity)
+SectionsPagerAdapter(fm : FragmentManager)
+getItem(position : int) : Fragment
+getCount() : int
+getPageTitle(position : int) : CharSequence
MainActivity
(gap::enekow::dollyremote)
~mSectionsPagerAdapter : SectionsPagerAdapter
~mViewPager : ViewPager
~TAG : String = "GAKOA"
-btAdapter : BluetoothAdapter
-ctx : Context
-connection : Connection
-state : int
-myLocale : Locale
-prevLang : String
~shouldExecuteOnResume : boolean
-mHandler : Handler
-REQUEST_ENABLE_BT : int = 1
#onCreate(savedInstanceState : Bundle) : void
+sendCommand(cmd : int, data : String) : void
-setupHandler() : void
+checkConnection() : void
+enableBt() : void
#onActivityResult(requestCode : int, resultCode : int, data : Intent) : void
-connectionLost() : void
-disableAllButtons() : void
-enableAllButtons() : void
-getLengthInSteps(cm : int) : int
-startTimeLapse(data : String) : void
-timelapseInProgress() : void
+setStartStop_Stop() : void
+setStartStop_Start() : void
-showTimelapseStopDialog() : void
-sendConfig() : void
+onResume() : void
#onRestart() : void
#onStart() : void
#onStop() : void
#onPause() : void
+onCreateOptionsMenu(menu : Menu) : boolean
+onOptionsItemSelected(item : MenuItem) : boolean
+onTabSelected(tab : Tab, fragmentTransaction : FragmentTransaction) : void
+onTabUnselected(tab : Tab, fragmentTransaction : FragmentTransaction) : void
+onTabReselected(tab : Tab, fragmentTransaction : FragmentTransaction) : void
<<Interface>>
ManualFragment::Listener
(gap::enekow::dollyremote::ManualFragment)
+sendCommand(cmd : int, data : String) : void
+setStartStop_Start() : void
+setStartStop_Stop() : void
ManualFragment
(gap::enekow::dollyremote)
~TAG : String = "GAKOA"
~mCallback : ManualFragmentListener
+onAttach(activity : Activity) : void
+onCreateView(inflater : LayoutInflater, container : ViewGroup, savedInstanceState : Bundle) : View
+disableButtons() : void
+getVelocity() : int
SettingsActivity
(gap::enekow::dollyremote)
~TAG : String = "GAKOA"
~lang : String
~preferences : SharedPreferences
~prevLang : String
~myLocale : Locale
+onCreate(savedInstanceState : Bundle) : void
-bindPreferenceSummaryToValue(preference : Preference) : void
+onPreferenceChange(preference : Preference, value : Object) : boolean
#onStop() : void
+setLocale(lang : String) : void

B. ERANSKINA

Komunikazio-protokoloa

Dokumentu honetan dollyaren eta D-Lappse aplikazioaren arteko Bluetooth konexioan zehar bidaltzen diren aginduak azalduko ditugu; honekin, D-Lappse aplikazioarekin bateragarria den hardwarea eraikitzea ahalbidetuz.

B.1 D-Lappse aplikazioak bidalitako aginduak

Konfigurazioa ezarri Konfigurazioa ezarri agindua konexioa ezartzen den momentua bidaltzen da eta baita D-Lappse aplikazioaren ezaugarriak aldatzen diren bakoitzean ere.

- Agindua: **CONF**
- 1. parametroa: kameraren modeloa adierazten du.
  - Luzera: 2 byte.
  - Formatua: NI (Nikon), CA (Canon), OL (Olympus), MI (Minolta), PE (Pentax), SO (Sony).
- 2. parametroa: bidearen luzeera adierazten du (pausotan[1]).
  - Luzera: 4 byte.
  - Formatua: [0000 - 9999] tarteko zenbakia.

[1] Pauso bakoitza urratsez urratseko motorrak egiten duen mugimendu bakoitza da.

Time-lapsea hasi Time-lapsea hasteko agindua erabiltzaileak *Hasi* botoia sakatzen duenean bidaltzen da.

- Agindua: **STRT**
- 1. parametroa: argazki kopurua adierazten du.
  - Luzera: 5 byte.
  - Formatua: [00000-99999] tarteko zenbakia.
- 2. parametroa: frame-tartea adierazten du.
  - Luzera: 5 byte.
  - Formatua: [00.00 - 99.99] tarteko zenbakia (lehenengo bi zifrek segundoak adierazten dituzte).
- 3. parametroa: mugimendua adierazten du.
  - Luzera: 2 byte.
  - Formatua: LR (ezkerretik eskuinera), RL (eskuinetik ezkerrera), NO (mugimendurik ez).

Ezkerrera mugitu Ezkerrera mugitzeko agindua erabiltzaileak *dena ezkerrera* [«] edo *ezkerrera* [<] botoia sakatzen duenean bidaltzen da.

- Agindua: **LEFT**
- 1. parametroa: mugimenduaren abiadura adierazten du.
  - Luzera: byte 1.
  - Formatua: [0-9] tarteko zenbakia.

Eskuinera mugitu Eskuinera mugitzeko agindua erabiltzaileak *dena eskuinera* [»] edo *eskuinera* [>] botoia sakatzen duenean bidaltzen da.

- Agindua: **RIGT**
- 1. parametroa: mugimenduaren abiadura adierazten du.
  - Luzera: byte 1.
  - Formatua: [0-9] tarteko zenbakia.

Gelditu Gelditu agindua erabiltzaileak *gelditu* botoia sakatzean edota *ezkerrera* [<] edo *eskuinera* [>] botoia askatzean bidaltzen da.

- Agindua: **STOP**

Argazkia atera Argazkia atera agindua erabiltzaileak *argazkia* botoia sakatzen duenean bidaltzen da.

- Agindua: **SHOT**

B.1.1 Dollyaren driverrak bidaltzen dituen aginduak

Bidearen amaierara iritsi da Dollya bidearen ertz batetara iristean bidaltzen da agindua.

- Agindua: **DEND**
- 1. parametroa: bidearen zein ertzetan dagoen adierazten du.
  - Luzera: byte 1.
  - Formatua: L (ezkerreko ertzera iritsi bada), R (eskuineko ertzera iritsi bada).

Konfigurazioa ezarri da D-Lappse aplikazioak konfigurazioa bidaltzen duenean, konfigurazioa ezartzen saiatu ondoren bidaltzen da.

- Agindua: **CONF**
- 1. parametroa: konfigurazioa ondo ezarri den ala ez adierazten du.
  - Luzera: 2 byte.
  - Formatua: OK (konfigurazioa ondo ezarri bada), KO (konfigurazioa ezin izan bada ezarri).

Time-lapseak huts egin du D-Lappse aplikazioak time-lapsea egiteko agindua bidali ondoren, time-lapsearen ezaugarriak bete ezin direnean bidaltzen da.

- Agindua: **FAIL**

C. ERANSKINA

Kalitatearen egiaztapen-zerrenda

Proiektuaren kalitatearen egiaztapen zerrenda	Bai/Ez
Plangintzatutako datak bete daitezke?	
Proiektuaren irismena bat dator egindako lanarekin?	
Proiektuaren garapena plangintzatutakoarekin bat dator?	
Egindako estimazioak zuzenak edo onargarriak dira?	
Proiektuaren arriskuak kontrolatuak daude?	
Proiektuaren interesatuak proiektuaren egoeraren jakitun dira?	
Proiektuaren exekuzioaren gora-berak behar bezala dokumentatu dira?	
Egin da proiektuaren segurtasun kopiarik azken aldian?	

Produktuaren kalitatearen egiaztapen zerrenda	Bai/Ez
D-Lappse aplikazioa gutxienez 3 mugikor ezberdinetan erabilgarria da?	
D-Lappse aplikazioan erakusten den informazioa bat dator sistemaren egoerarekin?	
D-Lappse aplikazioak bidaltzen dituen aginduak behar bezala burutzen dira?	
Sistemaren integritate fisikoa bermatuta dago?	
D-lappse aplikazioak ez du ustekabeko itxierarik jasaten?	
Sistemak azkar erantzuten du bidaltzen diren aginduen aurrean?	

D. ERANSKINA

Erabilpen-gida

Ondorengo orrietan sistema martxan jarri eta erabiltzeko beharrezkoak diren argibideak daude. Egitura hau jarraitzen du gida honek:

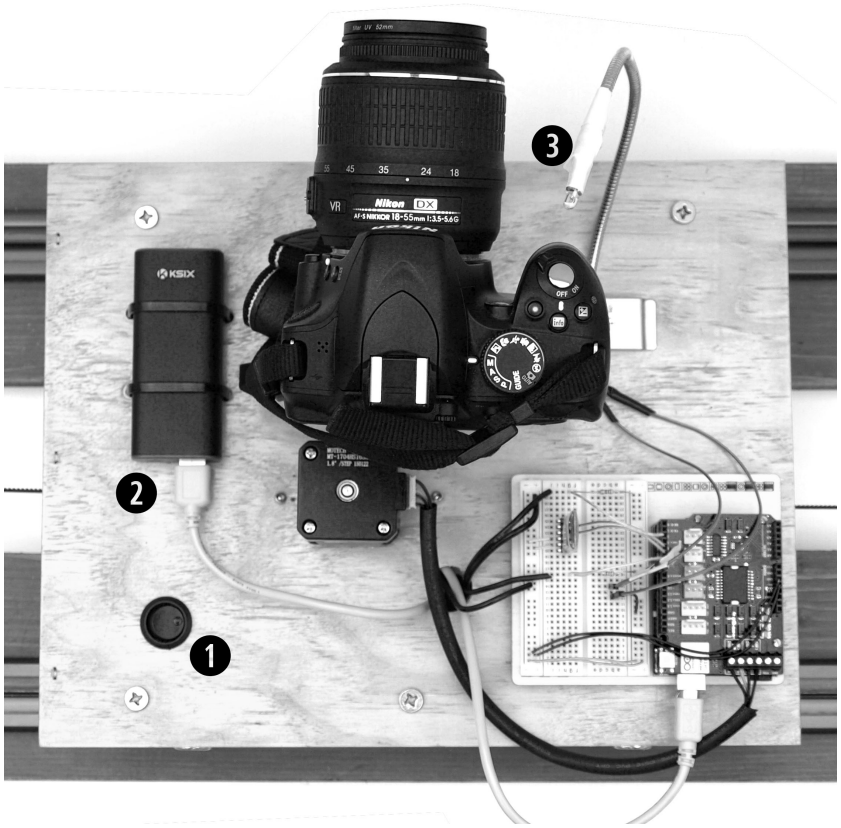
- Dollya prestatu
- D-Lappse app-a instalatu
- D-Lappse app-aren funtzionamendua

D.1 Dollya prestatu

- Dollya piztu: dollya piztu eta itzaltzeko etengailua (1) aktibatu behar da. Etengailua aktibatu ondoren, elikadura iturria (2) martxan jarriko da eta kolore berdeko argitxoak piztuko zaizkio. Argitxo berdeak elikadura iturriaren karga maila adierazten dute: geroz eta argitxo gehiago aktibatu, orduan eta karga gehiago duenaren seinale da.

- IR diodoa kokatu: diodoa (3) kameraren infragorri hartzailera begira jarri behar da. Kamera guztiek ez dute toki berean infragorri hartzailea, informazio hori kameraren erabilpen gidan aurkitu daiteke.

- Kamera prestatu: kamera piztu eta infragorri hartzailea aktibatu. Urrats hau egiteko informazioa ere kameraren erabilpen gidan aurkitu daiteke.



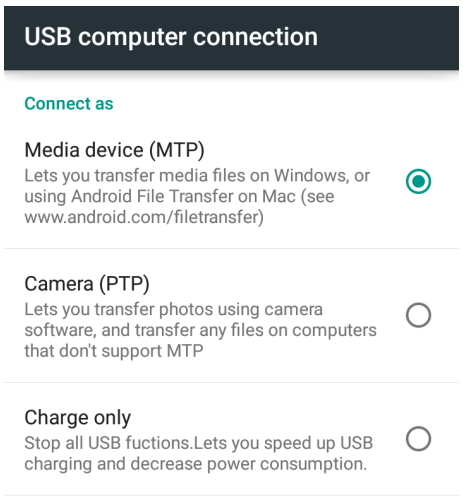
D.1 Irudia: Dollya goitik ikusita.

D.2 D-Lappse app-a instalatu

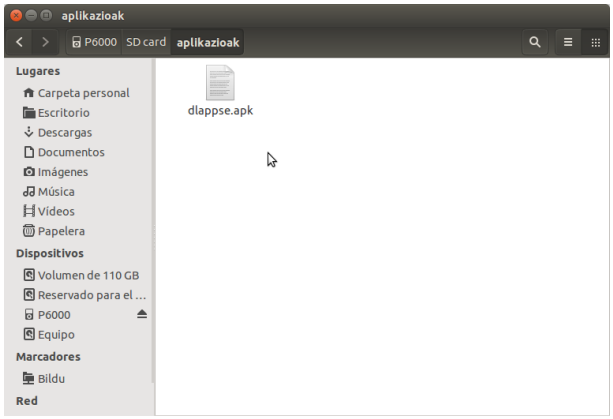
D.2.1 Aplikazioaren instalatzailea kopiatu

Lehenik eta behin, D-Lappse aplikazioa instalatzeko *dlappse.apk* fitxategia gure mugikorrera[1] kopiatu behar dugu. Horretarako hurrengo urrats hauek jarraitu behar dira.

- Mugikorra ordenagailura konektatu USB bidez.
- Mugikorra biltegiratze moduan jarri (ikus D.2 irudia).
- Ordenagailutik mugikorraren biltegiratze memoriara sartu.
- *dlappse.apk* mugikorraren biltegiratze memorian kopiatu (ikus D.3 irudia).



D.2 Irudia: Biltegiratze moduan konfiguratu.



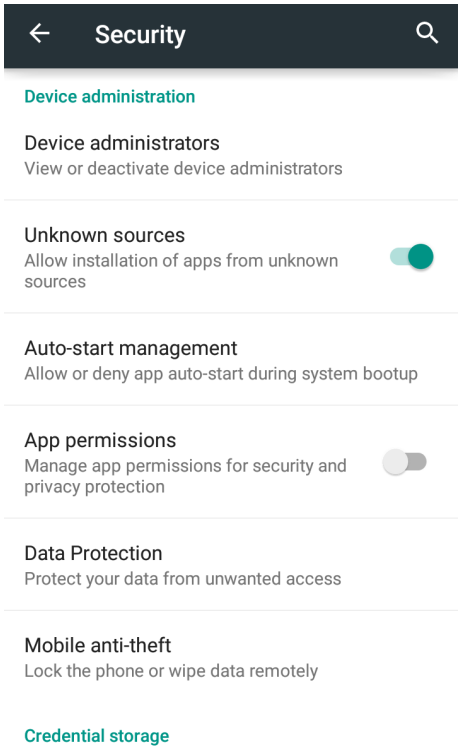
D.3 Irudia: Mugikorraren biltegiratze memorian kopiatu.

D.2.2 Aplikazioa instalatu

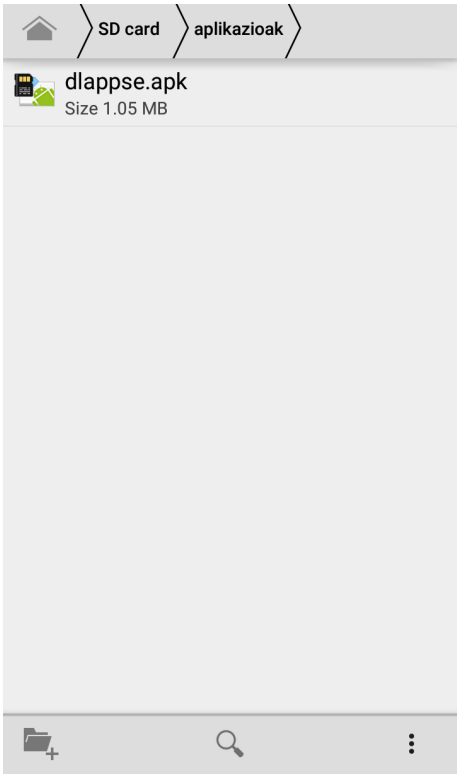
Aplikazioa instalatu ahal izateko, lehenengo Android sistema eragilearen segurtasun aukera batzuk aktibatu behar dira. Horretarako Android mugikorra hartu eta *Ezarpenak* > *Segurtasuna* atalera joan behar da. Ondoren,*Jatorri ezezagunak* aukera aktibatu behar dugu, D.4 irudian ikusten den modura.

[1] Aplikazioak Android sistema eragilearen 4.1 bertsioa edo bertsio berriagoa behar du.

Hurrengo eginbeharra, kopiatu dugun *dlappse.apk* instalatzailea bilatzea da, horretarako *Fitxategi kudeatzailea* aplikazioa (edota fitxategiak kudeatzeko edozein aplikazio) erabiliko dugu. Instalatzailea aurkitzen dugunean (ikus D.5 irudia), abiarazi egingo dugu, ondoren, Android sistema eragileak gidatuko gaitu aplikazioa instalatu arte.



D.4 Irudia: Jatorri ezezagunak aktibatua.



D.5 Irudia: *dlappse.apk* gure Android mugikorrean.

Aplikazioa instalatu ondoren, *aplikazioen kutxan* aurkituko dugu D-Lappse aplikazioa.

D.3 D-Lappse app-aren funtzionamendua

D.6 eta D.7 irudietan ikusi daitezke azalpen hauei dagozkien erreferentziak.

D.3.1 Dollyarekin konektatu

Konexioa egin ahal izateko, mugikorraren Bluetooth-a aktibatu beharra dago: Android sistemaren ezarpenetan aktibatu daiteke, baina baita D-Lappse aplikaziotik bertatik. Nahikoa da menua (3) sakatu eta *Konektatu* aukeratzea, automatikoki Bluetooth-a aktibatzeko eskaera zabalduko da.

D-Lappse aplikazioa exekutatzen denean (edota Bluetooth-a aktibatzen dugunean) automatikoki egiten da dollyaren bilaketa. Dollya topatutakoan lehenengo eginbeharra dollyarekin parekatzea da. Prozesua guztiz gidatua da: aplikazioak berak erakusten du dollyarekin parekatzeko eskaera. Bertan pasahitza sartu behar da (1234) eta onartu. Parekatzea ondo burutu bada, aurrerantzean automatikoki konektatuko da dollyarekin.

Dollyarekin konexioa ezartzean mezu bat azalduko zaigu konexioa ezarri dela abisatzen eta aginduak bidaltzeko botoiak aktibatuko dira.

Momenturen batean konexioa galdu egingo balitz, D-Lappse aplikazioak abisu bat erakutsiko luke eta botoiak desaktibatu egingo lirateke. Berriro ere dollyarekin konektatu ahal izateko menuko (3) *Konektatu* aukera sakatzea besterik ez dugu.

D.3.2 Time-lapsea egin

Time-lapsea egiteko *TIME-LAPSE* (1) pantailan egon beharra daukagu. Pantaila honetan, time-lapseak izango dituen ezaugarriak konfiguratu behar ditugu: lehenengo time-lapsea egiteko beharrezko denbora aukeratuko dugu (4); zenbakiak gora eta behera mugituz edota aukeratu daiteke denbora edota zenbakien gainean sakatu eta eskuz sartu dezakegu denbora. Ondoren, time-lapseak izango duen abiadura aukeratu behar dugu (5): *Frame-tartea* aukeratu dezakegu, argazki batetik bestera egongo den tartea definitzeko, edota *azelerazioa* aukeratu dezakegu, abiadura erreala zenbat bider azkartuko den aukeratuz. Konfigurazioarekin amaitzeko time-lapsearen mugimendua aukeratuko dugu (6). Mugimendua kutxa aktibatu edo desaktibatu dezakegu mugimendua egin edo ez erabakitzeko eta mugimendua egitea erabakitzen badugu, norabidea aukeratu.

Time-lapsea hasi aurretik, laburpena ikusi dezakegu (7), bertan, emaitzaren iraupena eta aterako den argazki kopurua ikusi daiteke. Azkenik, *Hasi* botoia (8) sakatu dezakegu time-lapsea abiarazteko. Time-lapsea martxan dagoen bitartean, (8) botoia *Gelditu* testuarekin azalduko zaigu eta time-lapsea gelditzeko balioko digu.

D.3.3 Dollya eskuz kontrolatu

Dollya eskuz kontrolatzeko, *ESKUZKO KONTROLA* pantailan (2) egon behar gara. *Slider*-a (9) alde batera eta bestera mugituz mugimenduaren abiadura aukeratu dezakegu eta botoiak sakatuz berriz, zenbait ekintza egin ditzakegu:

(11) Argazkia aterako du kamerak.

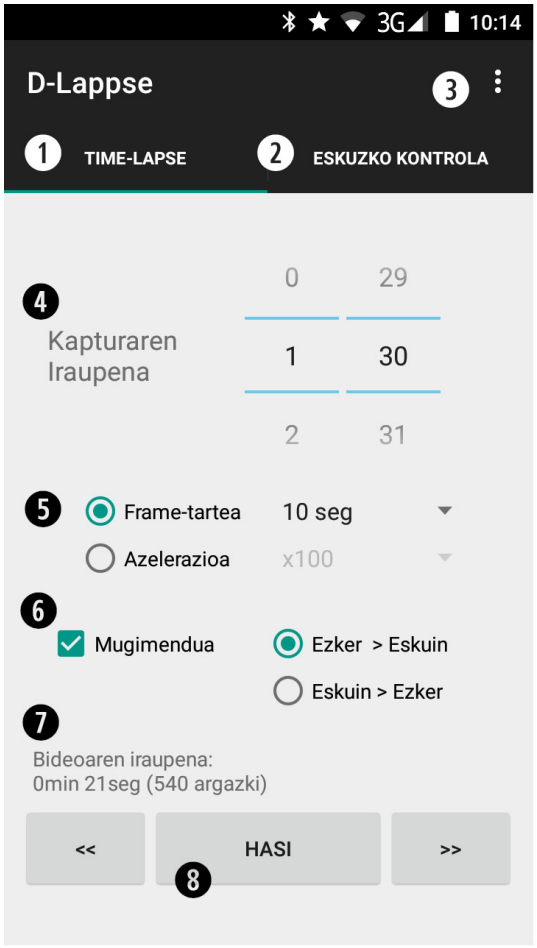
(10) Dollya ezkerrerantz mugituko da botoia sakatua mantentzen dugun bitartean, askatzean gelditu egingo da.

(13) Dollya ezkerrerantz mugituko da, ezkerreko ertzera iritsi arte.

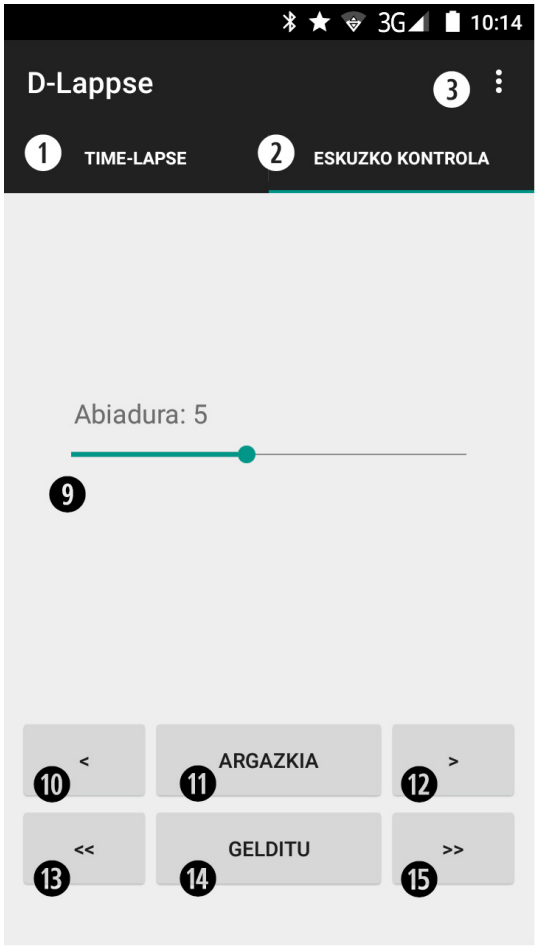
(12) Dollya eskuinerantz mugituko da botoia sakatua mantentzen dugun bitartean, askatzean gelditu egingo da.

(15) Dollya eskuinerantz mugituko da, eskuineko ertzera iritsi arte.

(14) Dollya gelditu egingo da.



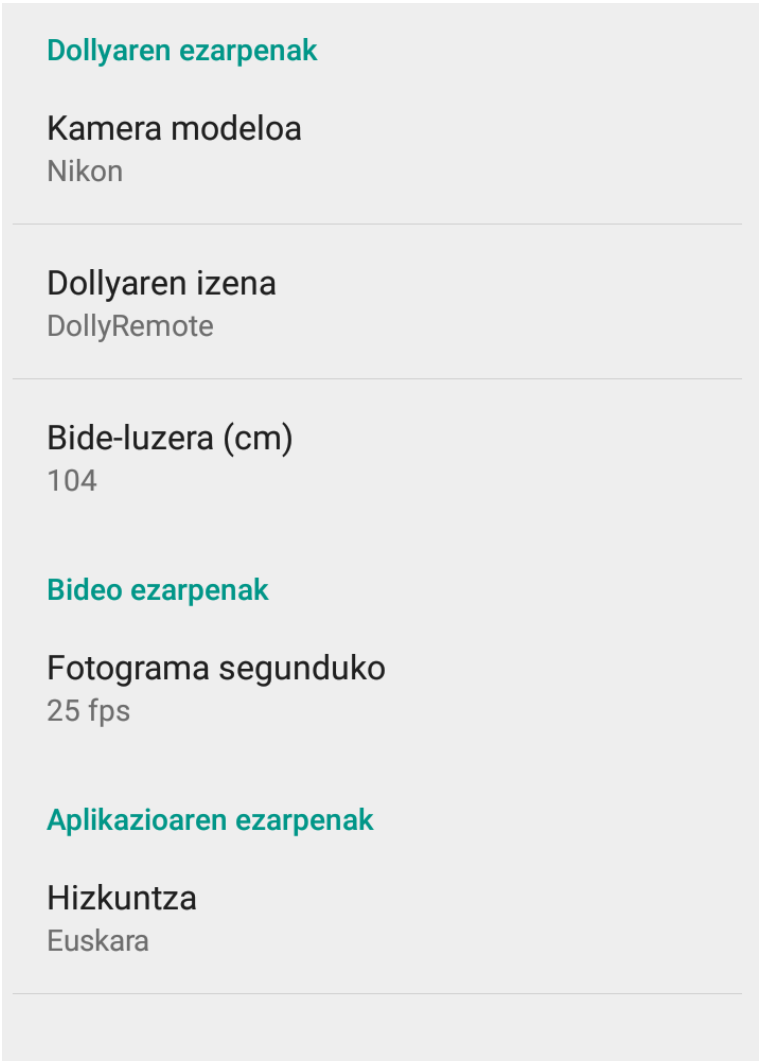
D.6 Irudia: *TIME-LAPSE* pantaila.



D.7 Irudia: *ESKUZKO KONTROLA* pantaila.

D.3.4 Ezarpenak aldatu

Ezarpenak aldatzeko, ezarpenen pantailara heldu behar gara lehenengo. Horretarako, menuko (3) *Ezarpenak* aukera sakatuko dugu.



D.8 Irudia: Ezarpenen pantaila.

Ezarpenen pantailan (ikusi D.8 irudia) zenbait ezarpen ditugu aldatzeko. Ezarpen bakoitzaren gainean sakatuta, hori aldatzeko aukera izango dugu.

- Kameraren modeloa: zenbait kamera modelo izango ditugu aukeran; Canon, Minolta, Nikon, Olympus, Pentax eta Sony. Horien artean bat aukeratuko dugu.

- Dollyaren izena: dollyaren Bluetooth moduluak konfiguratuta duen izena jarri behar da; defektuz *DollyRemote* da.
- Bide-luzera: bidearen luzera sartu behar da zentimetrotan.
- Fotograma segundoko: time-lapse sekuentzia sortzerakoan erabiliko dugun konfigurazioa; sortuko dugun bideoak segundoko izango dituen fotograma kopurua.
- Hizkuntza: hizkuntza aldatzeko aukera ematen digu honek; euskara, gaztelania edo ingelesa aukeratu daiteke.

Bibliografia

[Developers, 2015a] Developers, A. (2015a). Android studio. https://developer.android.com/sdk/index.html. [Atzipena: 2015-08-25].

[Developers, 2015b] Developers, A. (2015b). Best practices. http://developer.android.com/guide/practices/index.html. [Atzipena: 2015-07-05].

[EHU, 2015] EHU, I. F. (2015). Gradu amaierako proiektuari buruzko arautegia. http://www.ehu.eus/documents/340468/2334257/GAP_araudia.pdf. [Atzipena: 2015-08-15].

[IDC, 2015] IDC (2015). Smartphone os market share. http://www.idc.com/prodserv/smartphone-os-market-share.jsp. [Atzipena: 2015-07-05].

[Perception, 2015] Perception, D. (2015). Multi-axis timelapse and video motion. https://www.dynamicperception.com/. [Atzipena: 2015-07-03].

[Pucci, 2014] Pucci, M. (2014). Arduino time lapse. https://www.youtube.com/watch?v=IrYYqB02508. [Atzipena: 2015-07-02].

[Setz, 2011] Setz, S. (2011). Multi camera ir control. http://sebastian.setz.name/arduino/my-libraries/multi-Camera-IR-Control. [Atzipena: 2015-07-30].

[TIMELAPSE, 2010] TIMELAPSE (2010). Motion controlled dollyes and timelapse photography. http://timelapsesa.co.za/. [Atzipena: 2015-07-02].

[ymkangyt, 2012] ymkangyt (2012). Diy motorized dolly for timelapse. https://www.youtube.com/watch?v=X-AA7CItCOM. [Atzipena: 2015-07-02].