Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Informatika Fakultatea

Informatika Ingeniaritzako Gradua

▪ Gradu Amaierako Lana ▪

Konputagailuen Ingeniaritza

Berotech: berotegi automatizatua.

Arkaitz Legarra Ibarguren

2019 - ekaina

Zuzendariak

Izaskun Etxeberria eta Jose Ignacio Martín

Esker onak

Bereziki eskertu nahi diet proiektu hau posible egin duten Izaskun Etxeberria eta Jose Ignacio Martín zuzendariei, eskaini didaten dedikazio eta laguntza benetan aipagatzekoa baita.

Orokorrean, urte hauetan nire irakasle izan direnei ere eskerrak eman nahi dizkiet izan duten laguntzeko prestutasunagatik eta kontzeptu teorikoak modu praktiko batean lantzeko proiektuak proposatu izanagatik.

Azkenik, hainbat proiektu eta ordu partekatu ditugun ikaskideak ere aipatu nahi ditut, ibilbide akademikoa atsegingarriagoa egin izanagatik eta beste hainbat arlotan partekatu ditugun esperientziengatik.

Laburpena

Proiektu honetan, modu automatikoan lan egiten duen berotegi bat eraiki da. Modu automatikoaz funtzinatzeaz gain, erabiltzaileak edozein unetan berotegia osatzen duten elementuen egoera aldatzeko aukera izango du, garatu den interfazearen bidez. Kontrol hori edozein lekutatik egin ahal izateko gainera, interfazearen eta berotegiaren arteko komunikazioa UDP protokoloa erabiliz inplementatu da, beraz, sarearen bidez bidaltzen eta jasotzen dira agindu eta datuak.

Berotegiaren funtzionamendua ahalbidetzeko, etengabe ikuskatu behar da elementuen egoera, hala nola, lurraren hezetasuna, kanpo tenperatura, etab. Horretarako, hainbat sentsore erabili dira eta horien kudeaketa Arduino Mega 2560 plakaren bidez gauzatu da.

Arduinok komunitate zabala du atzean, eta jadanik liburutegi ugari existitzen badira ere, guztiz baztertu dira, eta programazioa erregistro mailan egin da. Hau da, mikrokontrolagailuko moduluak eta hortik kanpoko beste gailuak behe-mailan landu dira.

Aurkibidea

Irudi eta Taulen zerrenda

IRUDIAK

TAULAK

1

Proiektuaren deskribapena

Kapitulu honetan proiektuaren helburuak eta landu nahi diren kontzeptuak azalduko dira. Horrekin batera, kontzeptuak lantzeko erabiliko den metodologia eta produktua osatzeko garatu beharreko elementuak azalduko dira. Proiektuaren deskribapen zehatza ere egingo da proiektuak jaso behar dituen atalak eta hauen ezaugarrien inguruan.

1.1. Sarrera

Etengabe entzuten dugu gizarteak guztiz aldatu behar dituela elikadura eta kontsumo ohiturak. Alde batetik, elikagaiek propietate ugari galtzen dituzte gaur egun laborantza masiboetan erabiltzen diren tekniken ondorioz. Bestetik, baliabide ugari xahutzen da larrean, neurketa fidagarririk egiten ez denez, proaktiboki egiten da lan, behar zehatzei erantzun ordez.

Arazo hauei aurre egiteko soluzioetako bat laborantza prozesua modu automatikoan egitea da. Etengabeko zaintzari esker, erabakiak modu erreaktibo batean hartuko dira, behar zehatzei soilik erantzunez, eta ondorioz, behar diren baliabideak soilik kontsumituko dira. Metodo honi esker gainera, landareentzat ingurune optimoa izango da uneoro eta uzta hobea lortuko da. Bestalde, zaintza-lanerako behar diren baliabideak murrizteko aukera izango da, bai erreminta aldetik, eta baita eskulan aldetik ere. Azken honek, landa-produktuen prezioak murrizten lagun dezake, gainera.

Hori guztia kontuan hartuta, berotegi automatizatua pertsona ezberdinei bideratuta dago: lehenik, beren irabaziak handitu eta baliabideak optimizatu nahi dituzten enpresei. Bigarrenik, landa lanaz ezagutzarik edo denbora faltaren ondorioz baratzik ez duen orori etxeko produktuak kontsumitzeko aukera zabaltzen dio.

1.2. Helburuak

Proiektuaren helburu nagusia, modu automatikoan zein eskuzkoan funtzionatzen duen berotegi bat eraikitzea da. Horretarako, beharrezkoak diren gailuak identifikatu, aztertu eta programatzen ikasi beharko da.

Proiektuaren bidez, atal ezberdinetan eta modu independentean landu diren teknologien arteko bateragarritasuna zenbaterainokoa den aztertu nahi da. Bai eta bateragarritasun horri esker bizitza errealeko egoera edo arazo bati aurre egiteko aukera dagoen edo ez ondorioztatu.

Berotegia osatzen duten gailu ezberdinen kontrolerako, mikrokontrolagailu bat erabiliko da. Mikrokontrolagailuaren barne-funtzionamendua ulertzeko asmoarekin, programazioa guztiz hutsetik egingo da; hau da, ez da jadanik existitzen den funtzio edo liburutegirik erabiliko. Azken horrek, datuak prozesatu, eskuratu eta erantzun bat emateko erabiliko diren gailuen kontrolerako, mikrokontrolagailuaren barne-moduluen konfigurazioa erregistro-mailan egitea behartzen du.

Behe-mailan lan egitearen helburua, *sistema txertatuen diseinua* ikasgaian landutako hainbat kontzeptu guztiz ezezaguna den arkitektura batera moldatzeak dituen zailtasunak ezagutzea da; erregistro-mailan lan egiterakoan jarraitu behar den metodologia gogora ekartzearekin batera.

Berotegiko gailuen egoera azterketa eta urruneko kontrolaren bidez, *sare zerbitzu eta aplikazioak* zein *sare teknologiak eta azpiegiturak* ikasgaietan landu diren komunikazio-protokolo, tresna eta baliabideak egoera erreal batean nola aplikatu daitezkeen aztertzea da beste helburu bat.

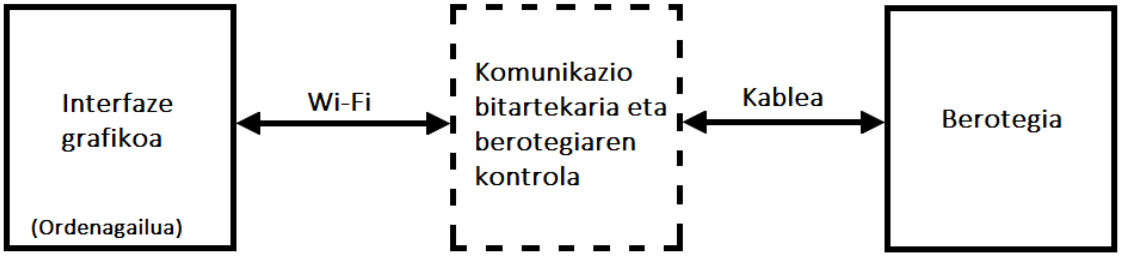
Azkenik, eta sinplea izango bada ere, datu-baseen inguruan ikasitako kontzeptuak lantzeko, berotegiaren parametro aldaketarako sarrera gisa erabiliko diren datuak kargatzeko, datu berriak sortzeko eta datu historikoa pantailaratzeko erabiliko da datu-basea.

1.3. Proiektuaren deskribapena

Jarraian, aipatutako helburuak lortzeko inplementatuko den sistemaren atal ezberdinak zein diren aipatuko da.

Ikuspuntu globaletik, lortu nahi dena, erabiltzaileak bisualki berotegi automatizatu baten parametroak ezagutu eta aldatzeko aukera izatea da. Kontrol hori edozein lekutatik egiteko aukera izango da, Wi-Fi bidezko komunikazioari esker.

Hori lortzeko, 3 atal nagusi bereizten dira: berotegia, interfazea, eta bien artean datu trukea gauzatzeko komunikazio-sistema. Atal ezberdinen arteko konexioak eta erabilitako komunikazio-teknologia zein den, 1.1 irudian ikus daiteke. Batetik, erabiltzaileak berotegiaren inguruko informazioa atzitu eta honen konfigurazioa aldatzeko interfazea dago. Beste muturrean, berotegia bera, landa-zaintza modu automatikoan egiteko behar den informazioa eskuratzeko gailuz eta berotegiaren elementuen egoera aldatzeko eragingailuz osatua. Bien artean, komunikazioa ahalbidetzeko beharrezko diren baliabideak eskaintzen dituen modulua eta berotegiko informazioa prozesatu eta horren araberako erantzun bat emateaz arduratzen den sistema.



1.1 irudia: Proiektuko atalen ikuspegi orokorra

1.4. Funtzionalitateak

Atal honetan, proiektua osatzen duten elementuek eskaini beharreko funtzionalitateak zehaztuko dira, bai eta beharrezko diren teknologiak ere.

- **Berotegia:** programatu diren gailuen eragina egoera erreal batean ikusteko, egitura fisikoki eraikiko da. Berotegiak elementu hauek izango ditu: atea, leihoa, ureztatze-sistema, haizagailua eta hezetasun eta tenperatura sentsoreak.
- **Berotegiaren kontrola:** berotegiko elementuen egoera kontrolatzeko sentsoreetatik eskuratutako balioen bilketa eta prozesamenduaz arduratuko den atala da. Datu hauen arabera, berotegian kokatutako eragingailuak aktibatu eta desaktibatu behar ditu. Berotegian bi magnitude kontrolatzen dira: tenperatura eta hezetasuna. Bi magnitude horien balioek ureztatze-sistemaren, leihoaren, atearen eta haizagailuaren egoeran eragingo dute jarraian azalduko den moduan:
    - Tenperatura: erabiltzaileak aukeratutako landare motarako optimoa den tenperatura mantentzen dela bermatu beharra dago. Horretarako, berotegian dagoen tenperatura etengabe kontrolatzen da, eta definitu den tenperatura maximoa gainditzean, hozte prozesua martxan jartzen da.

3 dira tenperaturak baldintzatzen dituen ekintzak. Lehenik, berotegiko tenperatura egokia bada, haizagailua, atea eta leihoa guztiz itxita mantenduko dira. Bigarrenik, tenperaturaren goi-muga gainditzean, leihoa zabalduko da tenperatura murrizteko. Azkenik, landareentzako arriskutsua den tenperaturara iristean (tenperatura maximoa 10°C-z gainditzen bada), haizagailua martxan jarri eta atea guztiz zabalduko da, ahalik eta azkarren tenperatura balio egokietara itzularazteko.

- Hezetasuna: lurraren hezetasuna minimoa ere, erabiltzaileak hautatutako landare motaren araberakoa izango da. Hezetasunak minimotik behera egiten badu, ureztatze-sistema martxan jarriko da n segundoz. Ureztaketen artean m segundoko tartea utziko da, lurrak ura xurga dezan. Kasu honetan, ureztatze-sistema bakarra da berotegi guztiarentzat. Hori dela eta, bi sentsore erabili dira, bakoitza berotegiaren alde banatan kokatuta. Hezetasunaren kalkulua bi sentsoreetan irakurritako balioen arteko batezbestekoa da; honela bi muturretan dauden landarek hezetasun maila desberdintasun handiak izatea ekidingo da. Proba kasuak egiterakoan, ureztatze-sistema 4 segundoz mantentzen da aktibatuta, eta ureztaketen arteko denbora minimoa 30 segundokoa da.

- **Interfaze grafikoa:** erabiltzaileak berotegiko elementuen egoeraren berri izateko eta parametroak kontrolatzeko erabiliko da eta funtzionalitate hauek jaso behar ditu:
  - Berotegiaren elementuen egoera pantailaratu: berotegian kokatu diren sentsore eta elementuen inguruko informazioa pantailaratuko da.
  - Berotegiaren elementuen egoera aldatu: uneko hezetasun eta tenperatura arbuiatuz, erabiltzaileak ureztatze-sistema, leihoa, atea eta haizagailua eskuz kontrolatzeko aukera izango du.
  - Datuen historikoa: denboran zehar tenperaturak eta hezetasunak izan dituzten gorabeherak aztertzeko, 30 segundoan behin bi magnitude hauek datu-basean gordeko dira.
  - Landare hautaketa: berotegiaren portaera unean landatuta dagoen landarera egokitzea ahalbidetuko du.
  - Erabiltzaile kontuak kudeatu: berotegi automatizatuko tenperatura eta hezetasun mugak aldatzeko baimena pertsona jakin batzuek soilik izan dezaten, erabiltzaile kontuak erabiliko dira.

- **Komunikazioa:** interfazeak aipatu berri ditugun funtzionalitateak eskaintzeko, beharrezkoa da interfazearen eta berotegia kontrolatzeko erabilitako txartelaren artean komunikazioa izatea. Batetik, informazioa pantailaratzeko datuak txartelean bertan eskuratzen eta prozesatzen dira, beraz, berotegiko elementuen egoeraren berri izateko datu horiek interfazera bideratu behar dira. Bestalde, erabiltzaileak berotegiko elementuen egoera aldatzeko aginduak ematen dituenean, interfazeak identifikatu eta horren araberako eskaera bat egingo dio txartelari.

  Agindu eta datu trukea zehatz-mehatz sistema honetarako soilik erabiliko denez, komunikazio-protokolo pertsonalizatu bat inplementatuko da, eta ekintza bakoitza modu unibokoan identifikatzen duen agindu-multzoa definituko da. Aginduak zein diren eta bakoitzerako espero den erantzuna 8. Kapituluan ikus daiteke.

  Interfazea eta berotegia bananduta daudenez, agindu eta datuak Wi-Fi bidez bideratuko dira batetik bestera.

2

Plangintza

Plangintzan proiektua garatzeko beharrezkoa den informazioa zehaztuko da. Proiektuaren fase bakoitza desberdindu eta bakoitzerako denbora estimazioa egingo da. Bestalde, proiektua amaitutzat emateko bete behar dituen kalitate-parametroak definituko dira. Amaitutakoan, plangintza honekin alderatuta, denbora kudeaketa benetan espero dena izan den edo ez baieztatzeko erabiliko da eta hala ez bada, zergatia aztertzeko aukera zabaltzen du.

2.1. Sarrera

Proiektuan zehar erreferentzia gisa erabiliko den atala izango da hau. Egindako aurreikuspenen arabera, proiektua entrega datarako prest egotea espero den edo ez jakiteko balioko du. Uneren batean desbiderapen nabarmenak sumatuz gero, lan egiteko moduan edota antolakuntzan aldaketak egingo dira.

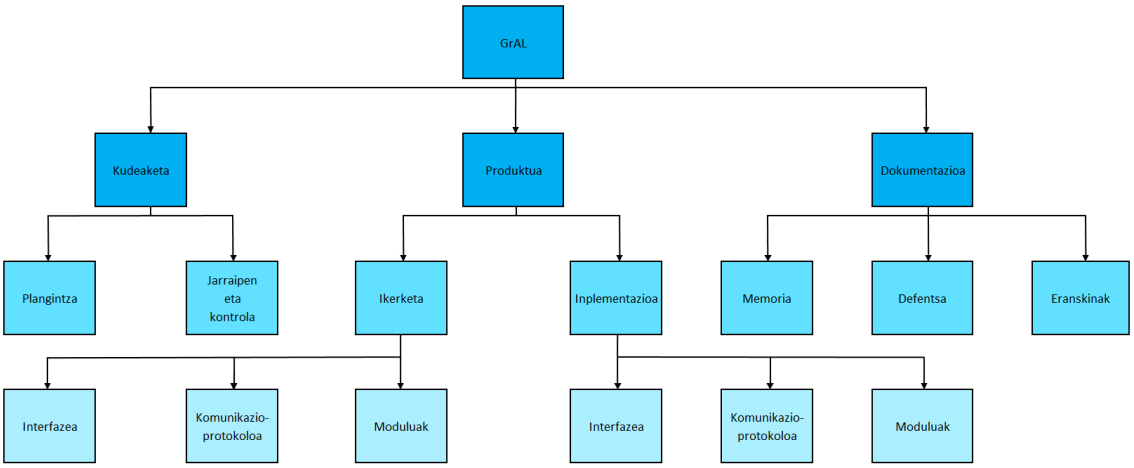
2.2. Emangarriak

Proiektua amaitzean entregatu beharreko emangarriak hauek dira:

- Eraiki den egitura, hau da, berotegia.
- Berotegiaren kontrolerako kodea.
- Interfaze grafikoaren kodea, komunikaziorako beharrezkoa den guztiarekin.
- Memoria

2.3. Lan-paketeak eta denbora estimazioak

2.1 irudiko LDE diagraman, proiektua guztiz osatzeko landu beharreko atalak ikus daitezke. Lan-pakete bakoitza helburu zehatz bati lotuta dago.



2.1 irudia: LDE diagrama

2.3.1. Kudeaketa

Proiektuaren ezaugarriak finkatzeko eta bete beharreko denbora tarteen xehetasunak definitzeko behar diren ataza guztiak biltzen ditu, kalitate minimoa bermatzeko baldintzak bete diren edo ez erabakitzeko atazez gain.

- Plangintza: eskakizunen identifikazioa, hasierako erabakiak eta proiektuaren nondik norakoarekin zerikusirik duten zalantzak argitzea.
- Jarraipen eta kontrola: plangintzan definitu diren funtzionalitate eta betebeharrak betetzen ari diren edo desbiderapen nabarmenik gertatu den edo ez identifikatzeko.

2.3.2. Produktua

Berotegia, interfaze-grafikoa eta komunikazio-protokoloa definitu eta inplementatzeko beharrezko diren teknologiak aukeratzeaz gain, garapen prozesuarekin zerikusia duten beste edozein ataza jasotzen dira.

- Ikerketa: proiektuaren atal nagusiak nola inplementatu behar diren aztertu, eta horretarako existitzen diren teknologiak ezagutu eta erabiltzen ikasi.
- Inplementazioa: ikertutako teknologia hauek erabiliz sentsore eta eragile ezberdinak garatzeko beharrezko moduluak programatu, komunikazioa gauzatu eta interfazea garatu.

2.3.3. Dokumentazioa

Gradu Amaierako Lanean egindako lana justifikatzeko, azaltzeko eta aurkezteko behar den dokumentazioaren prestakuntzarako atazak.

- Memoria: proiektuan burututako lana, erabilitako baliabideak eta jasotako informazioa biltzen duen dokumentua.
- Defentsa: proiektua tribunalaren aurrean aurkezteko erabiliko diren baliabideak prestatu.

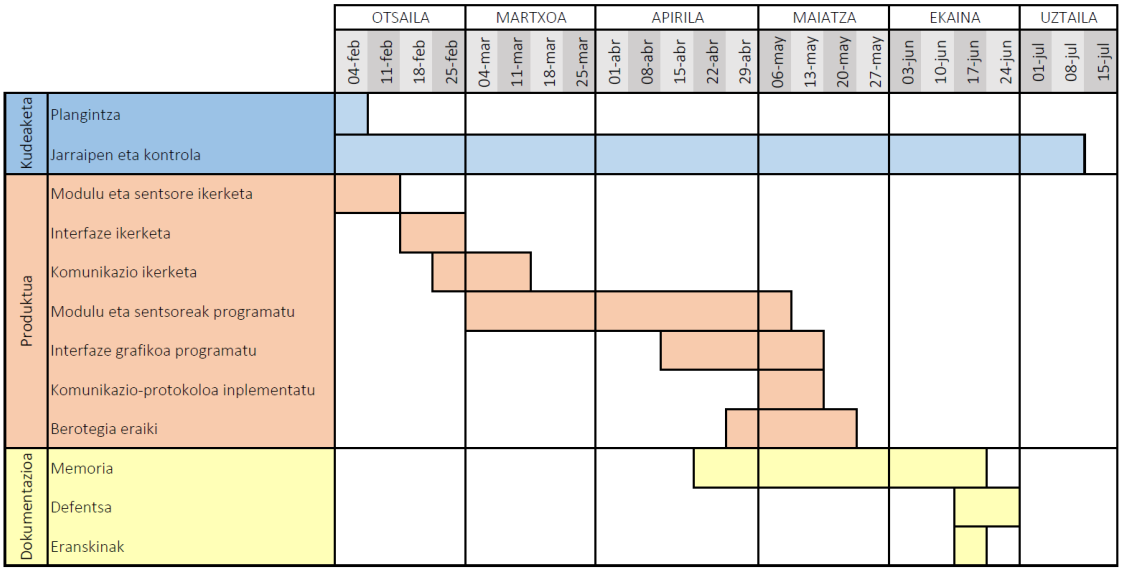
2.3.4. Atazen denbora estimazioak

2.1 taulan proiektuaren ataza ezberdinei dedikatu beharreko denbora estimazioak jaso dira.

2.2 irudiak, proiektuaren atal bakoitza zein data egunerako amaituta egotea espero den adierazten du. Planifikazio horri esker, uneoro jakingo da proiektuak martxa bera jarraituz gero denborarik izango den edo ez. Denbora falta sumatzen den kasuei erantzun bat eman behar zaie, proiektuari ordu gehiago dedikatuz eguneko, berriz ere aurreikuspenak beteko diren puntu batetara iritsi arte.

Kategoria	Lan-paketea	Denbora estimazioa
Kudeaketa	Plangintza	5o
	Jarraipen eta kontrola	10o
	Guztira	**15o**
Produktua	Interfaze ikerketa	6o
	Komunikazio ikerketa	12o
	Modulu eta sentsore ikerketa	8o
	Berotegia eraiki	15o
	Berotegia programatu	110o
	Interfaze grafikoa programatu	20o
	Komunikazio-protokoloa inplementatu	12o
	Guztira	**183o**
Dokumentazioa	Memoria	85o
	Defentsa	15o
	Eranskinak	2o
	Guztira	102o
Proiektuan guztira:		**300o**

2.1 taula: denbora estimazioak



2.2 irudia: gantt diagrama

2.4. Informazio-sistema

Bakarkako lana izanik, informazio-sistemaren kudeaketa eta antolakuntzak aldaketak jasan ditzakeen arren, lagungarria da hasieratik definitutako egitura zehaztea. Kasu honetan, bilduko diren elementuak ez dira horrenbeste; ondorioz, egunen batean egituran aldaketak egin nahiko balira ez litzateke hain kaltegarria izango. Hala ere, ahal den neurrian behintzat, egitura-aldaketak ekidingo dira.

Informazio galerak dituen kalteak ekidin nahian, proiektua hainbat lekutan egongo da kopiatuta. Dokumentaziorako kopia lokalen gainean lan egingo bada ere, eguneko lanekin edo aldaketekin amaitzen denean, *Google Drive* plataformara igoko da.

Proiektuaren kodeari dagokionez, *Github* biltegira igoko da modulu berriren bat programatzen denean edo dagoeneko existitzen den kodean hobekuntzak gehitzen direnean.

2.5. Kalitate-plana

Jarraian, proiektuak kalitate-minimoa izan dezan bermatzeko bete behar dituen ezaugarriak zehaztuko dira. Honi esker, proiektuaren amaieran, definitutako eskakizunak guztiz asetu diren edo ez aztertuko da, eta lortutako produktuaren inguruan hausnarketa egingo da.

Hauek dira proiektuak gutxienez bete beharreko xehetasunak:

- Kodea: kodeak modu argi batean egon behar du antolatuta, eta txukuntasuna zaindu beharko da. Horretarako, funtzioak ondo dokumentatuko dira, iruzkinak erabiliz. Funtzio-izenak adierazgarriak izango dira eta funtzio barruan exekutatutako kode-blokeak nahasgarriak direla ikusten bada, lagungarriak diren azalpenak emango dira. Hau baliagarria da bai kodea garatu ez duen pertsona batek ulertu eta alda dezan, eta baita garatzaileak funtzionalitateak hobetu nahi izanez aldaketak non egin behar dituen identifkatzeko.
- Komunikazio-protokoloa: ezerezetik sortuko da eta protokoloaren definizioa guztiz pertsonalizatua izango denez, ekintza bakoitza indibidualki identifikatuko duen kode-sistema erabiliko da. Kode bakoitzak exekutatuko duen ekintzaren ahalik eta adierazgarrien izan behar du.
- Berotegia: berotegiaren funtzionamendua egokia izan dadin gutxienez gailu hauek izan beharko ditu: serbomotorrak, lurraren hezetasun neurgailua, tenperatura sentsorea, atea ireki eta ixteko motorra eta ur-bonba.
- Dokumentazioa: proiektuan zehar egin den lanaren adierazle da. Proiektuaren xehetasunak ulertzera emateko, modulu bakoitza programatzeko jarraitu den metodologia eta honekin erlazionatutako edozein kontzeptu azaltzea ezinbestekoa da.

2.6. Denbora desbiderapen kontrola

2.2 taulan, proiektuan planifikazioan estimatutakoaren eta benetan dedikatu denaren konparazioa egingo da.

Ikus daitekeenez, interfazeak aldaketa nabariak jasan ditu, berotegiaren kontrolerako gailu aniztasun faltagatik intefazea gehiago lantzea erabaki baita.

Azkenik eta dokumentazioari dagokionean, memoriari garrantzi handia eman bazitzaion ere eta ordu kopurua kalkulua kasurik txarrenetarikoan egin bazen ere, ordu kopurua haratago joan da. Honen arrazoi nagusia, dokumentazioa idazten hasitakoan proiektua oraindik nahikoa garatu gabe egon izana da. Izan ere, aipatu beharreko kontzeptuak zein ziren ez zegoen argi eta ondorioz, dokumentuaren egitura desegokia zen. Bestalde, mikrokontrolagailuan konfiguratu behar izan diren moduluen barne-funtzionamenduaren zehaztasunak non mugatu eta zenbaterainoko zehaztasunak aipatu definitzea ere kostatu da.

Proiektu osoa kontuan izanda, ez da desbiderapen nabarmenik izan, desbiderapenak elkar orekatu baitira. Lan-paketeetan aldiz, desbiderapen nabarmenak izan dira proiektuaren fase nagusienetan.

Ikerketa fasean %50-eko desbiderapena izan da gutxi gorabehera. Batetik, interfazearen garapenerako erreminta hautaketa azkarra izan da, Qt oso ezaguna den eta asko erabiltzen den softwarea baita. Komunikazioarekin erlazionatutako ikerketak ere desbiderapen nabarmena izan du, *socket*-en erabileraren inguruan dokumentazio asko baitago eta honek asko erraztu du bilaketa prozesua. Azkenik, Arduino plataformetan erabiltzeko asko estandarizatu dira sentsore modeloak, eta proiektuan, sentsore ezagunenak erabiltzea erabaki da.

Inplementazioari dagokionean, interfaze grafikoak jasan du desbiderapen nagusiena. Honen arrazoia, hasieran planteatu ez ziren zenbait funtzionalitate gehitzearena izan da:

- Erabiltzaile kudeaketa
- Datuen historikoa
- Landare hautaketa

Kategoria	Lan-paketea	Denbora estimazioa	Dedikatutako denbora	Desbiderapena (%)
Kudeaketa	Plangintza	5o	3o	-40
	Jarraipen eta kontrola	10o	10o	0
	Guztira	**15o**	**13o**	**-13**
Produktua	Interfaze ikerketa	6o	3o	-50
	Komunikazio ikerketa	12o	5o	-58’33
	Modulu eta sentsore ikerketa	8o	4o	-50
	Berotegia eraiki	15o	16o	+6’66
	Berotegia programatu	110o	95o	-13’64
	Interfaze grafikoa programatu	20o	35o	+75
	Komunikazio-protokoloa inplementatu	12o	10o	-17
	Guztira	**183o**	**168o**	**-8**
Dokumentazioa	Memoria	85o	100o	+18
	Defentsa	15o	10o	-33’33
	Eranskinak	2o	2o	0
	Guztira	102o	112o	+9’8
Proiektuan guztira:		**300o**	**293o**	**-2’33**

2.2 taula: Denbora desbiderapenak

3

Berotegia

Kapitulu honetan, berotegiaren egitura nolakoa den ikusiko da, eta irakurketak egiteko gailuak eta eragingailuak non kokatu diren argituko da.

3.1. Sarrera

Berotegiaren egitura nagusia eraikitzeko PVC xaflak erabili dira. Konexio guztiak eta Arduino txartela berotegiaren barnean sartzea erabaki denez, tamaina nahiko handiko berotegia (3.1 irudia, eraikitzea erabaki da (50x35x35 cm)).

Konexioak eta Arduino Mega 2560 txartela ur-sistematik babesteko ere, beste egitura txikiago bat eraiki da. Egitura hau ordea, ez da finkoa, konexioak aldatu behar izanez gero arazorik gabe mugitzeko. Landareei dagokionez, 3 loreontzi ipini badira ere, benetan landareak berotegi guztian sakabanatuta egongo lirateke.

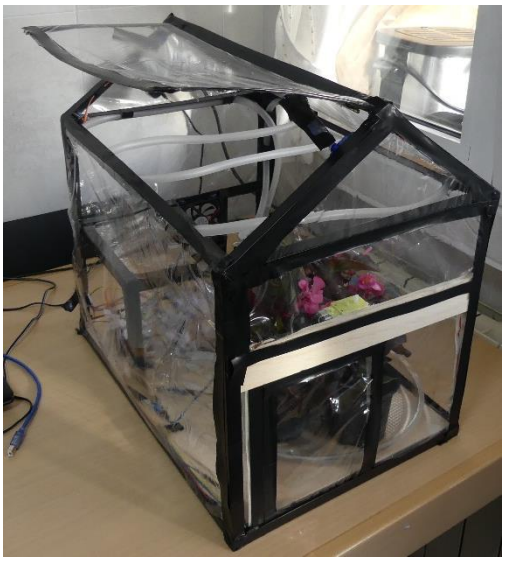
3.1 irudia: berotegia - albotik

3.2 irudia: berotegia - orokorra

3.2. Funtzionamendua

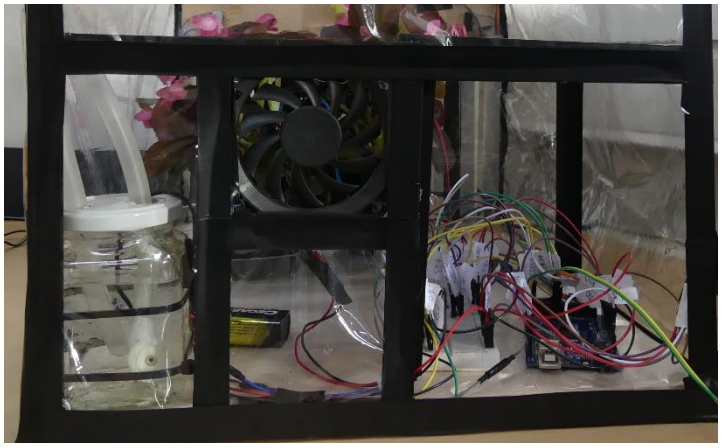
Berotegiaren funtzionalitate ezberdinak errepasatuko dira jarraian, eta bakoitza erabilitako posible egiteko erabilitako gailuak zerrendatuko dira:

- **Tenperatura:** hautatutako landarearen tenperatura maximoa gainditzen denean, leihoa zabalduko da (3.3 irudia). Leihoa zabalik egonda, tenperaturak gora egiten jarraituz gero eta tenperatura maximoa 10°C-z gainditzen bada, leihoaz gain, haizagailua (3.5 irudia) martxan jarri eta atea zabalduko dira. Leihoa zabaltzeko bi serbomotor (3.4 irudia) erabili dira, leihoaren alde banatan kokatuta.



3.3 irudia: leihoa zabalik

3.4 irudia: serbomotorra leihoan



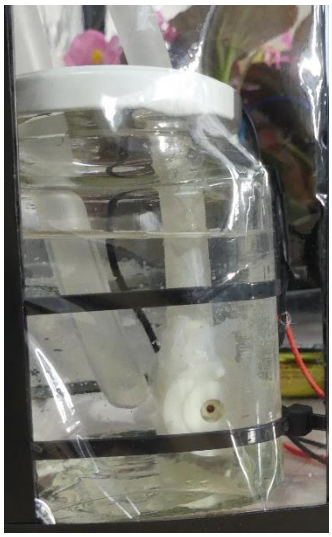
3.5 irudia: haizagailua

Tenperatura murrizten denean, pausoak kontrako norantzan egingo dira; hau da, tenperatura maximoa gainditu bada, baina 10°C baino gutxiagoz, atea itxi eta haizagailua gelditu egingo da. Tenperatura maximoa baino gutxiago izanez gero, leihoa itxiko da. Zenbait kasutan, hala ere, ez du zertan horrela gertatu, elementu bakoitza erabiltzaileak blokea baitezake. Adibidez, erabiltzaileak eskuz leihoa itxi du, beraz, tenperaturak maximoa gainditzean, ez da leihorik zabalduko. Horrez gain, ureztatze-sistema martxan jartzen denero, segurtasuna dela medio, leihoa ixtea erabaki da.

Atea mugitzeko DC motor bat erabili da (ikus 3.6 irudia), eta abiadura murrizteko motorrari erreduktore bat gehitu zaio. DC motorrak ate mugitzeko, engranajea eta kremailera elkartu dira. Atearen posizioa zein den ezagutzeko eta mugara iristen denean gelditu dadin, bi ibilbide amaiera kokatu beoregiaren alde banatan.

3.6 irudia: atea

- **Hezetasuna:** tenperaturaren antzera, hezetasuna maila landatutako landarera optimiza dezake erabiltzailea. Zehaztutako hezetasun minimora iristean, ureztatze-sistema martxan jarriko da 4 segundoan zehar eta ureztaketen artean 30 segundoko tartea utziko da gutxienez. Ureztatze-sistema, ur-bonba, ur-tankea eta berotegiaren goikaldean kokatutako hobi sortaz dago osatuta.



3.7 irudia: ur-tankea eta ur-bonba

3.8 irudia: ur hobiak

Hezetasunaren irakurketa egiteko 2 sentsore (3.10 irudia) erabili dira, berotegiaren aldebanatan kokatuta. Hezetasunaren kalkulua bi sentsoreen batezbestekoa da,

ureztatze-sistema bakarra denez, landare guztiak hartu behar dira kontuan, posible baita mutur ezberdinetan dauden landareak gutiz kontrako hezetasun maila izatea.

3.9 irudia: hezetasun sentsoreak

4

Gailuak eta teknologiak

Berotegiak eskaini beharreko funtzionalitateak asetzeko erabiliko den hardwarea nola kontrolatu behar den azalduko da. Berotegiaz gain, interfazea garatzeko softwarearen azalpena emango da. Azkenik, berotegiaren eta interfazearen artean bitartekari moduan lan egingo duen txartelaren ezaugarri batzuk aipatuko dira.

4.1. Sarrera

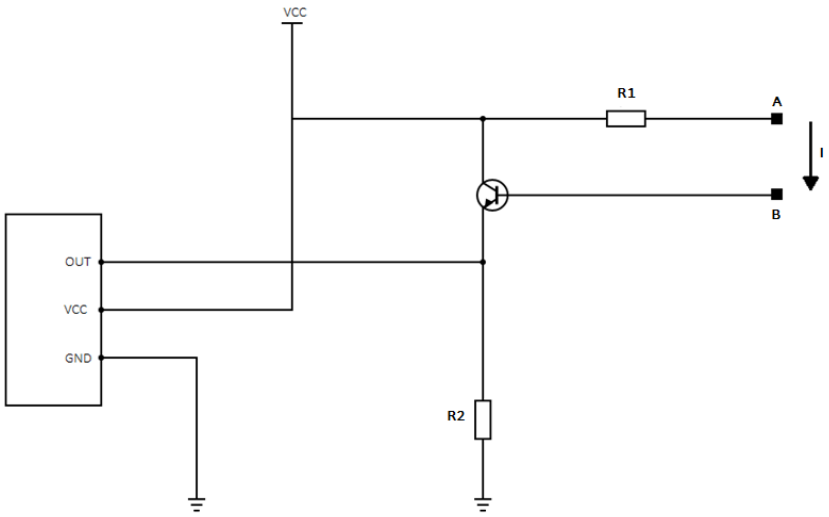
Bi atal nagusi bereizi dira kapitulu honetan. Batetik, berotegiaren uneko egoera aldatzeko instalatu den hardwarea, eta bestetik, interfazea sortzeko eta erabiltzaileari aukera ezberdinak eskaintzeko beharrezko diren teknologiak.

4.2. Gailuak

Berotegiaren elementu ezberdinak kontrolatzeko beharrezko diren gailuak zein diren eta hauen kontrolerako beharrezko diren teknologiak jasoko dira jarraian.

4.2.1. YL-69 hezetasun sentsorea

Lurraren hezetasuna neurtzeko sentsorea analogikoa da. Neurketa egiteko, 4.1 irudian, A eta B terminalen artean tentsioa aplikatzen da (bi terminal hauek, YL-69 sentsorearen hankatxoak dira). Bi muturren arteko tentsioa aldakorra izango da lurraren hezetasunaren arabera, lurra gero eta hezeago, orduan eta tentsio altuagoa lortzen da, lurraren erresistentzia murrizten baita. Bi terminoen arteko tentsioak, transistoretik igarotzen den korrontea regulatzen du, eta korronte hori da, hain zuzen, hezetasunaren balioa kalkulatzeko erabiltzen dena.

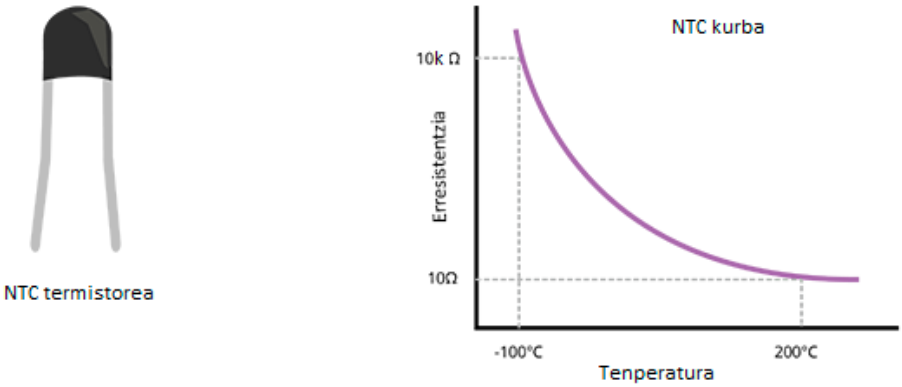


4.1 irudia: YL-69 sentsorearen zirkuitu elektrikoa

4.2.2. KY-013 tenperatura sentsorea

Sentsore honek NTC (*Negative Temperature Coefficient*) termistore bat baliatzen du inguruko tenperatura neurtzeko. Termistore bat tenperaturarekiko aldakorra den erresistentzia bat da.

Tenperaturak gora egin ahala, orduan eta erresistentzia txikiagoa aplikatzen da. Erlazio hori logaritmikoa da, 4.2 irudian ikus daitekeen moduan, tenperaturak termistorearen mugara egiten duen heinean, erresistentziak aldaketa txikiagoa sufritzen du graduko.



4.2 irudia: termistore baten NTC kurba

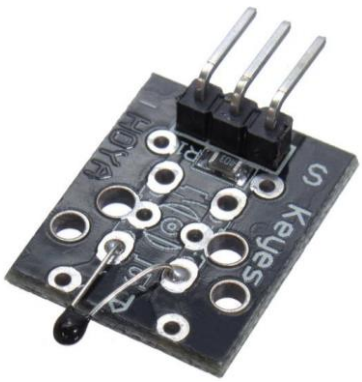
Portaera hau egokien islatzen duen ekuazioa 4.3 irudian agertzen den *Steinhart-Hart* ekuazioa da. Ekuazioa aplikatu ahal izateko, beharrezkoa da lehenik gutxienez 3 koefiziente ezagun eskuratzea, tenperatura eta erresistentzien arteko erlazioa zein den ondorioztatzeko. Koefiziente horiek, jadanik fabrikatzaileek sentsorearen datu-orrian aipatzen dituzte, normalean.

$$\frac{1}{T} = A + B\ln R + C(\ln R)^3$$

4.3 irudia: Steinhart-Hart ekuzioa

KY-013 tenperatura sentsorearen (4.4 irudia) *Steinhart* koefizienteak ondorengoak dira: A = 0.001129148, B = 0.000234125 eta C = 0.0000000876741.

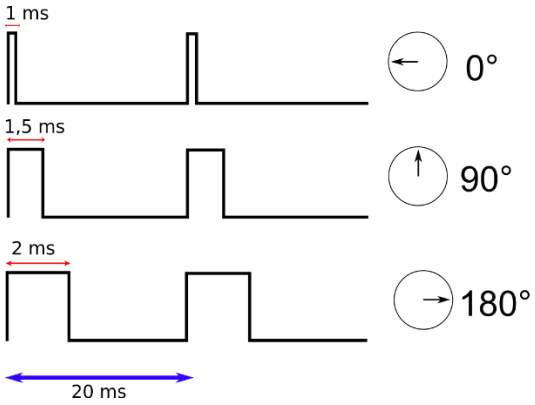
Ekuazioa eta aipatu berri diren koefizienteak, tenperatura *kelvin*-etan eskuratzeko dago pentsatuta, beraz, beharrezkoa da kalkulatutako tenperatura *celsius* sistemara bihurtzea.



4.4 irudia: KY-013 tenperatura sentsorea

4.2.3. Serbomotorrak

Leihoa zabaldu eta ixteko, leihoaren mutur banatan kokatu diren bi serbomotor erabili dira. Serbomotorrak PWM seinaleen bidez kontrolatzen dira. Serbomotorraren periodoa finkoa den arren, elikatzen duen seinalea aktibatuta mantentzen den denbora aldakorra izan daiteke (*duty* zikloa). Serbomotorra konektatu den hankatxora luzera ezberdineko pultsu elektrikoak bidaliz, serbomotorraren posizioa zehazten da, 4.5 irudian ikus daitekeen moduan.



4.5 irudia: PWM seinaleen duty eta periodoa

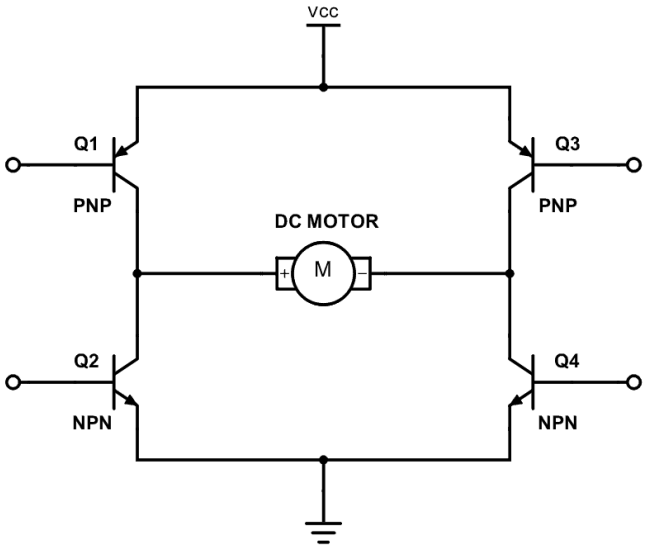
Hautatutako serbomotor modeloa SG90 (4.6 irudia) izan da. Oso oinarrizkoak eta potentzia gutxikoak diren arren, leihoa mugitzeko nahikoa dira.

4.5 irudian, SG90ren kontrola gauzatzeko denbora-tarteak agertzen dira. SG90ren periodoa 20ms-koa da; hau da, *duty* seinalea 20ms-ro bidali behar da. *Duty* seinalearen behe-muga 1ms-koa da. Seinalea 1ms-z aktibatuta mantenduz gero, serbomotorra 0°-tan egongo da. Goi-muga aldiz 2ms-koa da, eta hartuko duen posizioa 180°.

4.6 irudia: SG90 serbomotorra

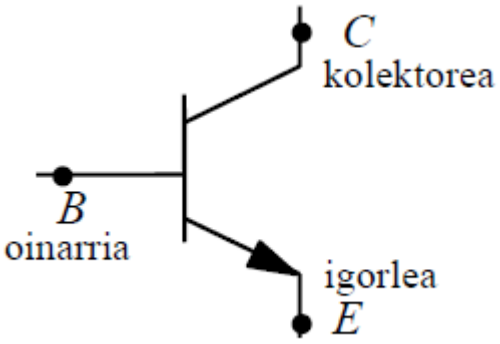
4.2.4. DC motorra

DC motorrak, aterako, ureztatze-sistemarako eta haizagailuan erabiliko dira. DC motorren biraketa sortzeko, motorraren borneen artean tentsioa aplikatu behar da, eta aplikatutako korrontearen noranzkoak, motorraren biraketa noranzkoa baldintzatzen du. Atearen kasuan, motor bakarrarekin itxi eta ireki nahi da eta motorrean aplikatutako korrontearen noranzkoa aldatzeko, *h-brigde*[2] zirkuitua eraiki da (ikus 4.7 irudia).



4.7 irudia: *h-bridge* zirkuitua

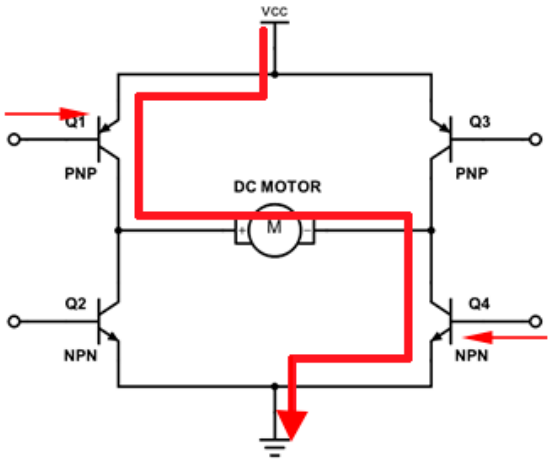
Zirkuitua 2 transistore parez dago osatuta. Transistoreak 3 borne (ikus 4.8 irudia) dituzten gailu elektrikoak dira: oinarria, kolektorea eta igorlea. Oinarrian korrontea aplikatzean, kolektorea eta igorlearen artean korrontea igarotzen uzten du.



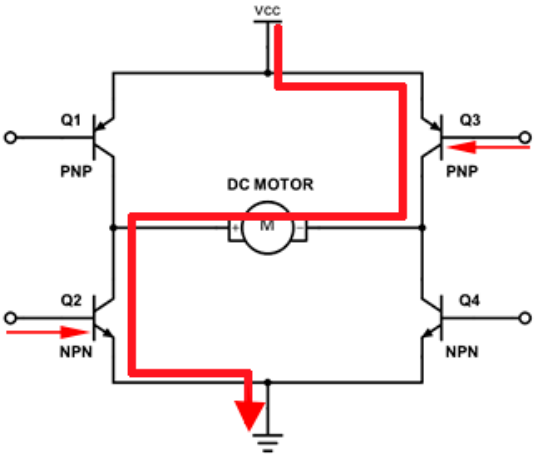
4.8 irudia: transistorea

Transistoreen portaera ikusita, zirkuituan alde banatan dauden 2 transistore aldi berean aktibatuz, DC motorrean korrontea aplikatuko da. 4.7 irudiko Q1 eta Q4 transistoreak aktibatzean (4.9 irudia), korrontea motorraren borne positibotik negatibora doa. Q3 eta Q2 aktibatzean (4.10 irudia), aldiz, negatibotik positibora. Motorrak espero duen portaera izan dezan, Q1 eta Q4 transistoreen oinarria txarteleko hankatxo berera konektatu dira, eta Q3 eta Q2 beste hankatxo

batera. Honela, hankatxoetako bat aktibatzen denean, dagokion transistore parea aktibatuko da, eta motorrak noranzko horretan biratuko du.



4.9 irudia: Q1 eta Q4 aktibatuta



4.10 irudia: Q2 eta Q3 aktibatuta

Ateak eta ur-bonbak (4.12 irudia) DC motor bera erabiltzen dute, 5V-tan 6000rpm inguruko abiadura duena. Ateko motorraren (4.11 irudia) kasuan, bira abiadura murrizteko 1:48 erlazioko erreduzitzaile bat gehitu zaio. Haizagailuak ere 5V-eko DC motorra du.

4.11 irudia: ateko motorra

4.12 irudia: ur-bonba

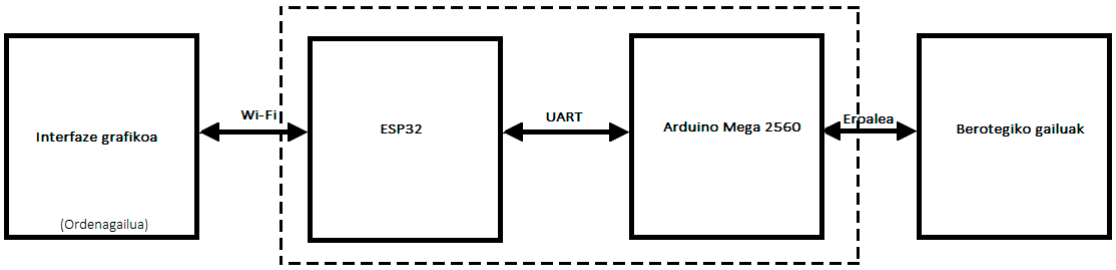
4.2.5. ESP32

Interfazearen eta berotegia kontrolatzeko erabili den Arduino Mega 2560 (5. Kapituluan sakondu da) txartelaren arteko komunikazioa burutzeko hautatu den txartela da.

ESP32[3] txartela (4.14 irudia) interfazean informazioa eguneratuta mantentzeko eta botoiak sakatzean burutu beharreko ekintzak identifikatzen dituen aginduak jaso eta Arduino Mega 2560 txartelera bideratzeaz arduratzen da. Honetarako beharrezkoa da ESP32 txartela, berotegia kontrolatzeko erabili den Arduino Mega 2560 txartelak ez baitu Wi-Fi komunikazioak gauzatzeko modulurik. ESP32 eta Arduino Mega 2560ren arteko komunikazioa kablezkoa da, zehazki, UART

moduluaren bidez gauzatu da (ikus 4.13 irudia). Beraz, ESP32 txartelak egin beharrekoa, Wi-Fi bidez aginduak jaso eta Arduino txartelerara UART moduluaren bidez igortzea da.

Komunikazioa bi noranzkotakoa da, deskribatu berri den komunikazioaz gain, ESP32ri Arduino Mega 2560tik datozkion aginduak eta erantzunak, interfazera bidali behar ditu Wi-Fi bidez.



4.13 irudia: ikuspegi globala

ESP32 txartela oso ezaguna bihurtu da IoT munduan, duen kontsumo baxuagatik eta jadanik integratuta dituen Wi-Fi eta Bluetooth moduluengatik, nagusiki.

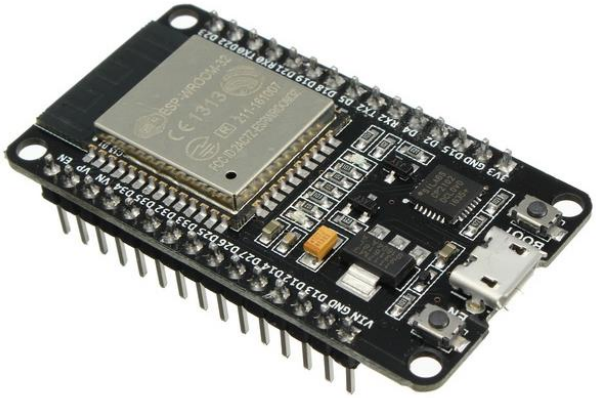
Hardwareari dagokionez, 240MHz-eko Xtensa Dual-Core 32-bit LX6 du instalatuta. Aginduak 16 bitekoak dira eta agindu-leihoak 16 biteko 7 agindu kargatzeko ahalmena du.

Bi nukleoen artean zuzeneko loturarik ez badago ere exekuzioan, memoria eta erregistro-multzoa partekatzen dituzte. Nukleoetako bakoitzak helburu zehatz bat du:

- **Protocol CPU (PRO_CPU):** sistemako moduluen kontrolaz eta Wi-Fi zein Bluetooth konexioen kudeaketaz arduratzen da.
- **Application CPU (APP_CPU):** aplikazio kodea exekutatzen du.

Memoria aldetik, 448KB-eko ROMa, 520KB SRAMa eta 16MB-eko kanpo flash memoria izateko aukera eskaintzen du.

Beste ezaugarri garrantzitsuenen artean, bere aurrekariekiko egin diren segurtasun hobekuntzak aipagarriak dira. Batetik, flash memorian iturburu-kodea enkriptatzera pasa da. Bestetik, komunikazio seguruak gauzatzeko enkriptazioa software bidez egin ordez, hardwarearen bidez egiten da, eta ondorioz, CPU karga murriztu da.



4.14 irudia: ESP32 txartela

4.3. Teknologiak

Atal honetan, proiektua osatzen duten elementu ezberdinak programatzeko erabili diren teknologiak aipatuko dira.

4.3.1. Atmel Studio 7

Atmel Studio 7[4] Visual Studio 10-en oinarrituta dagoen AVR eta SAM mikrokontrolagailuentzako programak garatzeko eta *debugg*eatzeko garapen plataforma da. Bertan sortutako proiektuak, mikrokontrolagailuan kargatzeko beharrezko konfigurazioa dute eta avrdude programaren bidez kargatzen dira.

4.3.2. Qt eta PyQt5

Qt[5] plataforma ezberdinetan erabili ahal izateko interfazeak garatzeko lan-ingurunea da. Interfazeak modu grafiko batean inplementatzeko aukera eskaintzen du, Qt Designer programaren bidez. Proiektua 1990 urtean jaio zen, Trolltech enpresaren eskutik, datu-baseak modu grafiko batean erakusteko beharrari soluzio bat emateko. 1992 urtean Nokia enpresak Trolltech erosi ondoren, teknologiaren garapenarekin jarraitu zuten, eta hasierako liburutegi arruntetik, lan-ingurune oso bater izatera pasa da.

Kode irekia izateari esker, komunitateak proiektu ezberdinak bultzatu ditu funtzionalitateak gehitzeko. C++ programazio-lengoaian kodetuta dago, eta hasieran programazio-lengoaia hau bakarrik onartzen bazuen ere, beste lengoaietarako euskarria ere izatea lortu da. Lengoaien artean: Python, Ruby eta Java.

Proiektuan, berotegia kontrolatzeko interfaze grafikoa Python programazio-lengoaian inplementatu da, PyQt5[6] erabilita. Azken hau, Python eta Qt-k eskaintzen dituen funtzio eta klaseen arteko lotura egiteaz arduratzen da. Horretarako, eta esan bezala, Qt C++ programazio-lengoaian kodetuta dagoenez, SIP erreminta da PyQt-ren oinarria, zeinak Python funtzio eta klaseak C++-en egiturara moldatzen dituen.

Fase ezberdinetan sortutako fitxategien arteko itzulpenak egiteko hiru programa bereizten dira:

- **pyuic5:** Qt Designer programaren bidez sortutako interfazeak Python kodera itzultzeko.
- **pyrcc5:** gehitu nahi diren irudi, ikono eta itzulpen fitxategien kudeaketaz arduratzen da.
- **pylupdate5:** Python kodearen itzulpen fitxategiak sortzen dituena.

4.3.3. MicroPython

MicroPython[7] ESP32 txartela programatzeko erabiliko den programazio lengoaia da. 2013an hasi eta oraindik garapenean dagoen Python 3-ren berrinplementazio bat da, baliabide gutxiko sistemetara bideratua dagoena. Python 3-n existitzen diren liburutegietan oinarritzen da, baina optimizazioa helburu izanik moldatu egin dira. Hasieran PyBoard mikrokontrolagailurako sortu bazen ere, izandako arrakastaren ondorioz ARM arkitekturan oinarritutako hainbat mikrokontrolagailutan erabiltzeko aukera zabaldu da; zehazki: ESP32, Arduino eta ESP8266.

Hau guztia kontuan hartuta eta jadanik inplementatutako liburutegien oinarria erabili bada ere zenbait desberdintasun daude Python 3 eta MicroPythonen artean:

- **Drastikoki murriztu diren liburutegiak:** liburutegi originaleko funtzioen itzulpen minimalista egin da, eta liburutegiaren funtzionamendu nagusia soilik eskaintzen dute. Kentzen diren funtzioak sistema txertatuetan erabiltzeak logika edo zentzurik ez duten funtzioak dira. Liburutegi mota hauek izendatzeko u aurrizkia gehitu diote liburutegi originalari.
- **Hutsetik inplementatutako liburutegiak:** Python-en sorreran sistema txertatuentzat erabilgarriak diren baina kontuan hartu ez ziren funtzionalitate berriak gehitzeko.
- **Mikrokontrolagailu zehatzetarako inplementatutako liburutegiak:** mikrokontrolagailu bakoitzak eskaintzen dituen baliabide berezi eta zehatzak aprobetxatzera orientatuak.

4.3.4. MySQL

Erabiltzaileen sarrera baimenak eta berotegiko datuen historikoa gordetzeko erabili den datu-base kudeatzailea. MySQL[8] munduan erabiltzen diren SGDB-etan gehien erabiltzen den artean dago. Bertako datu-basak erlazionalak dira, hau da, datuak egituratzeko taulak sortzen dira eta taula horien artean loturak sortzen dira. Oracle Corporation konpainia da teknologia honen jabea, baina kode irekikoa da.

5

Txartela

Aurreko kapituluan jorratutako hardwarearen kontrola burutzeko hautatu den txartelaren ezaugarriak eta hautaketaren arrazoiak zein diren aipatuko da. Horrez gain, barne funtzionamendua ezagutzeko, berotegia osatzen duten gailuren kontrolerako beharrezko diren hainbat moduluren konfigurazioa zer nolako den erakutsiko da.

5.1. Arduino

Erabili den modelo zehatzaren ezaugarriak aipatzen hasi aurretik, lehenik Arduino plataformaren[9] filosofia azalduko da.

2003an *Interaction Design Insitute Ivrea* unibertsitate italiarreko ikasle batzuk, garai horretan existitzen ziren plataformen alternatiba bat eskaintzeko sortutako proiektua da Arduino. Hasieran, proiektuaren helburu nagusia ikaskideentzako plataforma merkea eskaintzea zen, eta 2005 urtean, lehen Arduino txartela merkaturatu zuten. Zenbait urte igaro ondoren eta txartelak izandako arrakastaren ondorioz, 2012 urtetik aurrera modelo berriak merkaturatu zituzten, ahalmen handiagoko Cortex M3 eta 32 biteko ARM prozesadoreak zituztenak.

Arduino guztiz irekia da, bai kodearen ikuspuntutik eta baita hardware aldetik ere. Hori dela eta, enpresa ugari dira txartel hauek eskaintzen dituztenak. Enpresen arteko konpetentziak, jadanik merkea zen plataformaren prezioa nabarmen murriztu du.

Komunitate oso zabala lortu du urte gutxian, eta plataformarik erabilienetarikoa da proiektu txikietarako. Arrazoi honegatik, enpresa ugari dira txartel hauei funtzionalitateak gehitzeko gailuak (pultsometroak, tenperatura sentsoreak, komunikaziorako gailuak, etab.) sortzen dituztenak. Proiektu honetan erabiliko ez badira ere, gailu ezberdinen horien kontrolerako jadanik liburutegi ugari daude inplementatuta.

Aipatu berri diren arrazoien artean, komunitate zabala izatea da plataforma hau erabiltzeko erabakia hartu izanaren eragile nagusia, hain erabilia den plataforma baten behe-mailako funtzionamendua zer nolakoa den ulertu nahi izan baita. Gainera, komunitate handia denez, hautatutako gailu ezagunen eta plataformaren inguruan informazio zabala aurkitzea espero da, bai eta barne-funtzionamenduaren inguruko informazioa.

Behin plataforma zehaztuta, Arduinok eskaintzen dituen plaken artean hautaketa egingo da jarraian. 5.1 taulan erabiltzaile mailan ezagunenak eta gehien erabiltzen diren modeloak alderatu dira.

5.1 taulan jaso diren modelo erabilienen artetik hautatutakoa Mega 2560 izan da. Aukeraketa fasean, hainbat arrazoirengatik baztertu dira gainerakoak:

- Zero eta Due: gehiegizko prozesamendu ahalmena dute. Due gainera berezia da, AVR ordez, ARM arkitekturan oinarritzen delako. Zero-ren kasuan, 14 portu bakarrik izateak arazoak sor ditzake proiektuan funtzionalitate berriak gehitzea erabakitzen bada.
- Uno eta Nano: ez dituzte behar diren adina tenporizadore.

Proiektuaren beharretara gehien hurbiltzen dena, beraz, Arduino Mega 2560 da. Prozesamendu-ahalmen nahikoa du eta ez da portu faltarik izango.

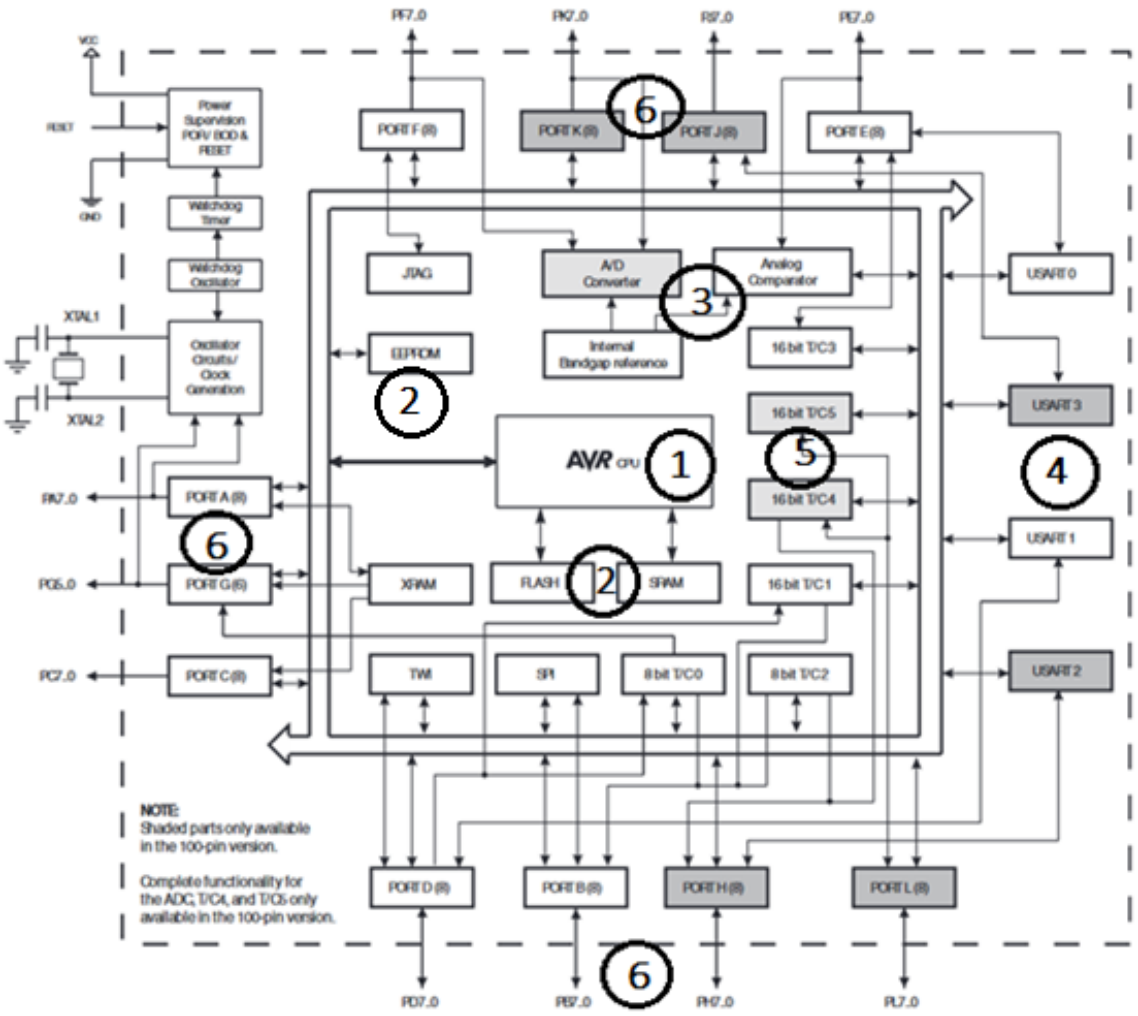
Izena	Mega 2560	Uno	Zero	Due	Nano
Mikrokontr.	ATmega2560 (8-bit)	ATmega38P (8-bit)	ATSAMD21G18 (32-bit)	ATSAM3X8E (32-bit)	ATmega168 (8-bit)
Tentsioa [txartela] / [kanpo elikadura maximoa]	5 V / 12V	5 V / 12V	3.3 V / 12 V	3.3 V / 12 V	5 V / 9 V
CPU abiadura	16 MHz	16 MHz	48 MHz	84 MHz	16MHz
Sarrera analogikoak	16	6	6	12	8
Irteera analogikoak	0	0	1	2	0
Sarrera/Irteera digitalak	54	14	14	54	14
SRAM[kB]	8	2	32	96	1
Flash[kB]	256	32	256	512	16
U(S)ART	4	1	2	4	1
Tenporizadoreak	6	3	6	9	3

5.1 taula: Arduino familiako txartelen konparaketa

5.1.1. ATmega2560 mikrokontrolagailua

Jarraian, Arduino Mega 2560 txartelak duen mikrokontrolagailuaren barne-egitura azalduko da; horretarako, 5.1 irudian agertzen den bloke-diagraman zehaztutako zenbaketari erreferentzia egingo da.

1. AVR prozesadorea: datu eta programaren exekuzioa guztiz bereizten duen Harvard arkitektura du, datu eta exekuzioarentzat bus eta memoria ezberdinak erabiltzen dituena. Agindu bateko agindu-leihoari esker, eta blokeaketarik gertatzen ez bada, exekuzioa jarraitua da, agindu bat exekutatzen ari den bitartean, hurrengoa jadanik kargatuko baita. Ziklo bakarrean atzigarri diren 8 biteko 32 erregistro ditu eta UAL-ean exekutatzen diren eragiketa aritmetikoak ere ziklo bakarrekoak dira.
2. Memoriak: 3 memoria bereizten dira. Batetik, flash memorian hasieraketa programa eta erabiltzailearen programa gordetzen dira. Bestetik, SRAM memorian datuak gordetzen dira. Azkenik, EEPROM memorian, txartelak korrontea galtzen badu ere, galduko ez diren datuak gordetzen dira.
3. ADC bihurgailua: balio analogikoak digital bihurtzeaz arduratzen den modulua da. PORTF eta PORTK hankatxoetan konektatuta dauden ADC kanaletako datua irakurtzen ditu.
4. USART: kanpo-sistemekin komunikatzeko modulua. Guztira, 4 modulu daude, eta bakoitzari dagokion PORT erregistroetan definituta dauden sarrera/irteeren bidez bidali eta jasotzen dituzte datuak.
5. Tenporizadoreak: 16-biteko 4 eta 8-biteko 2 tenporizadore ditu integratuta. Hauen konfiguraziorako ere, PORT erregistroetako hainbat bit eta datu erabiltzen dira.
6. PORT erregistroak: zenbait moduluren konfiguraziorako beharrezko diren parametroak zehazteko eta txartelaren hankatxoetatik seinaleak jaso zein bidaltzeko erabiltzen dira.



5.1 irudia: Atmega2560ren bloke-diagrama

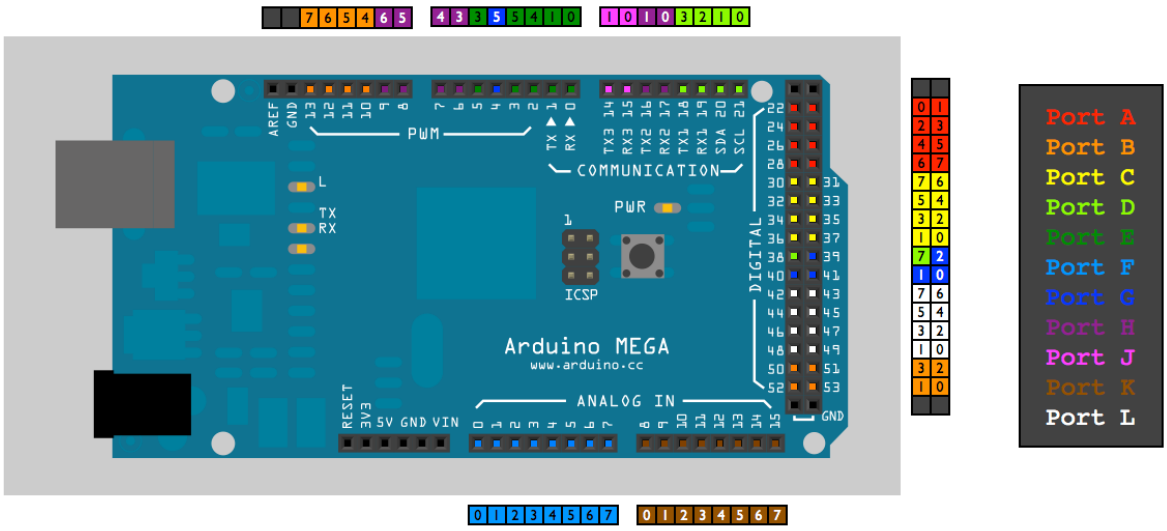
5.1.2. Sarrera/irteera portuak

Sarrera/irteera portuak[10] kanpo gailuak kontrolatzeko eta horietatik informazioa jasotzeko erabiliko dira.

Txarteleko hankatxoak 8ko taldetan antolatzen dira erregistro mailan (erregistro horiei, portu deritze). Talde bakoitzari erregistro zehatz bat dagokie eta txarteleko hankatxo bakoitza, erregistro horietako bit zehatz bat da. 5.2 irudian ikus daiteke hankatxo bakoitzari zein portuko (eskuineko zerrenda) bit zehatza dagokion (txartelaren irudia inguratzen duten balioak).

Hankatxo-talde bakoitzeko, 3 erregistro bereizten dira (x balioarekin 5.2 irudian zerrendatuta dauden portuak adierazi nahi izan dira):

- **DDRx:** hankatxoa sarrera edo irteera bezala erabiliko den zehazteko.
- **PORTx:** hankatxoaren uneko balioa adierazten du: 0 edo 1. Irteera batekin lan egiten denean, hankatxoari dagokion bit-ean 1 balioa ezarriz gero, hankatxotik korrontea bidaliko da; 0 esleituz gero, ez da korronterik bidaliko. Hankatxoa sarrera moduan konfiguratu bada, kanpo gailuetatik informazioa jasoko da, korronte moduan. Kanpo gailuak tentsioa igortzen duenean, hankatxoari dagokion bit-ean 1ekoa esleitzen da, eta korronterik jasotzen ez denean, 0koa.
- **PINx:** PORTxren antzera, hankatxoaren egoera adierazten du. Irakurketak soilik onartzen ditu erregistro honek, beraz, hankatxoak sarrera moduan daudenean hankatxoaren balioa eskuratzeko erabiltzen da.



5.2 irudia: Arduino Mega 2560 txartelaren portuak

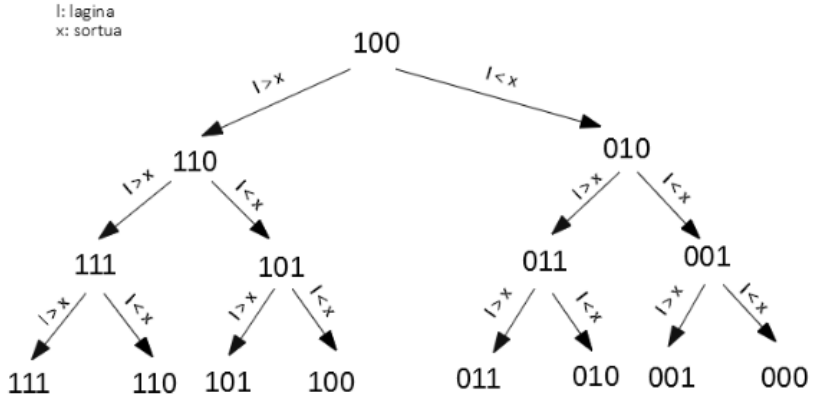
5.2. ADC bihurgailua

Tenperatura eta hezetasuna magnitude analogikoak dira. Arduino Mega 2560, ordea, sistema digitala da, eta ondorioz, seinaleak prozesatu aurretik, beharrezkoa da ADC bihurgailuaren bidez digitalizatzea.

Modulu honek, sarrera analogikoaren baliokidea den 10 biteko balioa kalkulatzen du, ondoz ondoko hurbilketa teknika aplikatuta.

Bihurketa prozesuan bi fase bereizten dira: laginketa eta digitalizazioa. Laginketa prozesuan, periodikoki, seinale analogikoaren balioa aztertu eta *sample and hold* zirkuitu elektrikoan kargatzen da (kargatu den balioari lagina deritzo). Zirkuitu honek, balioa konstante mantenduko du bihurketa prozesuan zehar. Digitalizazio fasean, laginan analogikoaren baliokidea den balio digitala kalkulatzen da.

Balio digitalaren kalkulua, ondoz ondoko hurbilketa metodoaren bidez egiten da. Balio digitala lortzeko, pisu handieneko bita aktibatzen da, eta balio horren parekoa den balio analogiko bat (tentsioa) sortzen da. Tentsio hori, laginketa fasean *sample and hold* zirkuituan kargatu den tentsioarekin konparatzen da. *Sample and hold* zirkuituko tentsioa, balio digitaletik lortu dena baina handiagoa bada, kode digitalean aktibatu den bita mantendu egiten da. *Sample and hold* zirkuituko tentsioa txikiagoa bada, aldiz, bita desaktibatu egiten da. Ondoren, hurrengo pisu handieneko bita aktibatu eta prozesua errepikatzen da, balio digital egokia topatzen den arte (5.3 irudiko adibidean 3 *bit*-eko doitasuna du moduluak).



5.3 irudia: ondoz ondoko bihurketa

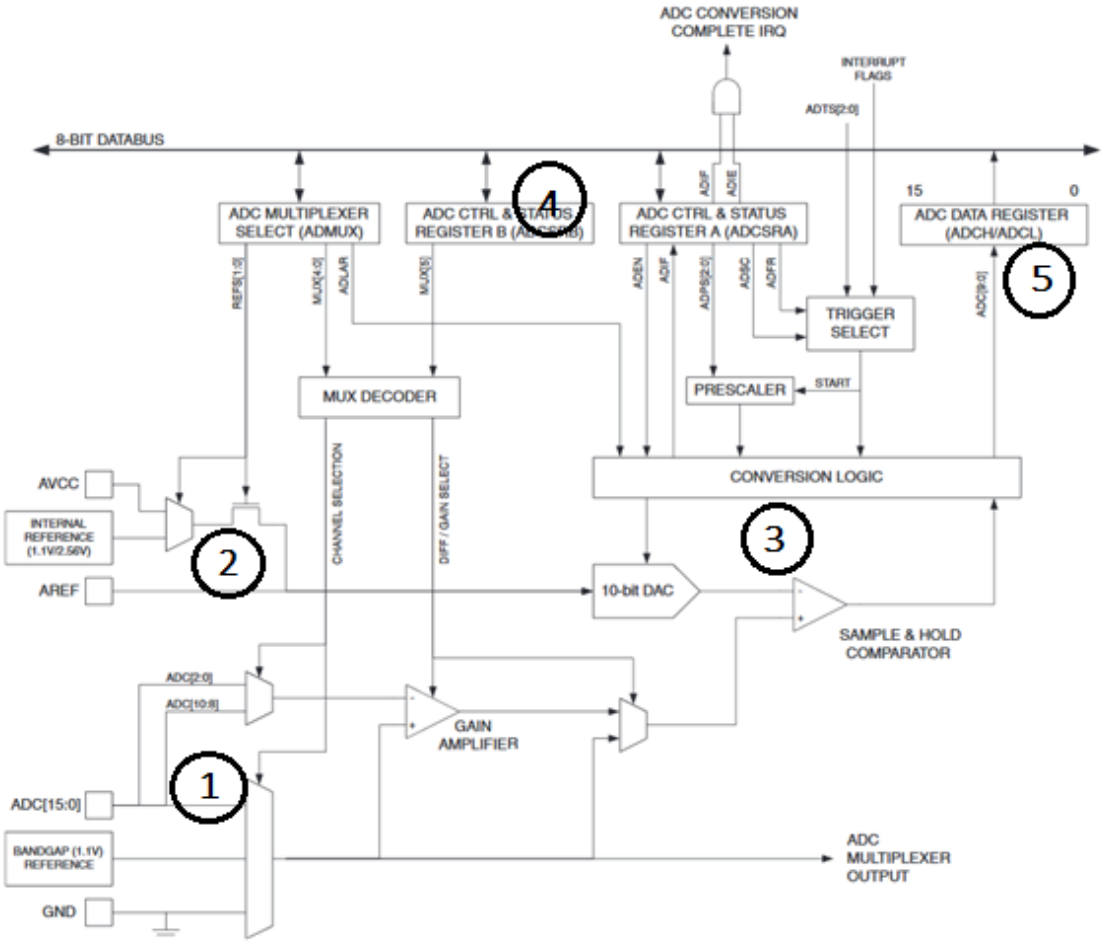
Sarrera gisa guztira 16 kanal ditu erabilgarri ADC bihurgailuak eta bi dira bereizten diren sarrera motak:

- **Single ended channel:** sarrera tentsioa kalkulatzeko, tentsio minimoa GND da.
- **Differential channel:** kanal hauetan tentsioa kalkulatzeko GND erabili ordez, definitu den beste sarrera analogiko batean antzeman den tentsioa hartzen da behe-erreferentzia gisa.

Differential moduan aritzeko aukera, kanal guztiek ez izan arren (6k soilik dute), proiektuan 16 kanalak erabiltzeko aukera dago, *single ended* moduan lan egingo baita.

ADC moduluaren barne-egitura 5.4 irudian ikus daiteke eta bihurgailua osatzen duten atalak ondorengoak dira:

1. Hautatutako kanaleko tentsioa eskuratzeko atala.
2. ADC bihurgailuak, balio digitalak kalkulatzeko, ADMUX erregistroan erabaki diren eta erreferentzia gisa erabiliko diren goi- eta behe-tentsioak jasotzeko sistema.
3. Deskribatu berri den ondoz ondoko bihurketak egiteko beharrezkoak diren elementuak dira hauek. 3 elementu nagusi ditu: *10-bit DAC*-ak, hautatutako balio digitalari dagokion balio analogikoa kalkulatzen du. *Sample and hold* konparatzaileak sarrera eta erreferentzia balioak konparatzen ditu eta *conversion logic*-en, konparaketaren emaitzaren arabera, hurrengo balio digitala definitzen da.
4. ADC bihurgailuaren konfigurazioa definitzeko erregistroak (5.2.1 atalean sakonki aztertuko dira).
5. Bihurketaren emaitza (digitala) gordetzeko erregistroa.



5.4 irudia: ADC bihurgailuaren bloke-diagrama

5.2.1. ADC erregistro-multzoa

ADC moduluaren konfigurazioa 4 erregistro nagusitan egiten da. Erregistro bakoitzak, ADC bihurgailuaren kontrolean helburu nagusi bat du eta erregistro bakoitzean hainbat eremu bereizten dira, ADC moduluaren konfigurazio zehatza egiteko erabiliko direnak:

- ADMUX eta ADCSRB: erreferentzia tentsioak, irakurketa egiteko kanal analogikoa eta balio digitalaren doitasuna hautatzeko erabiltzen dira.
- ADCSRA: bihurketak noiz eta zein baldintzatan egingo den zehazteko erregistroa.
- ADC datu-erregistroa: bihurketaren emaitza gordetzeko erregistroa.

Jarraian, erregistroetako eremu zehatzak zein diren, zertarako balio duten eta proiektuan zein konfigurazio erabili den aipatuko da. (Erabilitako konfigurazioa, eremu bakoitzaren izenaren ondoren datorren bit segida da, adib.: REFS[1:0] – 01, REFS eremuan, 01 balioak).

5.2.1.1. ADMUX – ADC Multiplexer Selection Register

Bit	7	6	5	4	3	2	1	0	
(0x7C)	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	ADMUX
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- REFS[1:0] - 01: erreferentzia-tentsioa hautatu. KY-013 eta YL-69 sentsoreek 0-5V arteko irakurketak egiten dituzte. Hori dela eta, ADC moduluan ere digitalizazio prozesurako bi muga horiek behar dira erreferentzia gisa. *REFS[1:0]* eremuan 01 bitak kargatu dira, honela, AREF pin-era konektatu den tentsioa hartzen da goi-muga gisa. Txartelean 5V-eko elikadura konektatu da adierazitako pin-ean.
- ADLAR - 0: datu-erregistroan, balioaren biten ordena zein den adierazi. Proiektuan emaitza eskuinetik doitua izatea erabaki da, gure ohiko erabilerara gehien gerturatzen den modua baita.
  Adib.: ADCko sarrerako irakurri eta bihurketa egin denean, balio digitala 64 eskuratu da. Eskuin doiketarekin 10 biteko datu erregistroan 0001000000 gordeko litzateke. Ezker doiketarekin, aldiz, biten ordena aldatzen da, pisu handieneko bita eskuinekoa izanik: 0000001000.
- MUX[4:0] - 00000: irakurketarako erabiliko den ADC kanala hautatu (hauek ez dira bit guztiak, *ADCSRB*-ko *MUX5* bita ere behar da). Lehen irakurketa 0 kanalean egiteko konfiguratu da. Hala ere, bihurketak burutzen direnean, kanalez aldatzen da.

5.2.1.2. ADCSRB – ADC Control and Status Register B

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	MUX5	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- MUX5 - 0: ADC kanalaren hautaketa osatu.
- ADTS[2:0] - 000: bihurketa bakoitzari hasiera emango dion ekintza aukeratu.
  Hautatutako modua *Free running mode* da; *ADCSRA* erregistroko *ADSC* eremuan 1-ekoa idatzitakoan hasiko da bihurketa prozesua.

5.2.1.3. ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **ADEN - 1:** ADC modulua gaitu.
- **ADSC - x:** bihurketa bat martxan jartzeko bita. Kasu honetan, bitaren balioa aldakorra da, eta bihurketa bakoitza amaitzerakoan, 1 balioa kargatuko da hurrengoarekin has dadin.
- **ADATE - 0:** *ADCSRB* erregistroko *ADTS* eremuan bihurketa hasieratuko duen gertaera-eragile bat konfiguratu bada, bit honen bidez baldintza gaitu edo desgaitzen da. *ADTS* eremuan *Free running mode* hautatu denez, bihurketaren hasiera ez da baldintzapekoa, beraz, ez da *ADATE* bita gaitu behar.
- **ADIF - 1:** ADC balio erregistroan datua prest dagoen edo ez adierazten du. Konfigurazio fasean desgaitzea erabaki da.
- **ADIE - 1:** ADC moduluak etenak sortzea baimendu (beharrezkoa da sistema osoko etenak baimenduta izatea ere).
- **ADPS[2:0] - 111:** ADC bihurgailuaren maiztasuna hautatzeko eremua. Sarrera gisa txartelaren maiztasuna hartuko da (16MHz) eta hori moldatuta, ADC bihurgailuaren maiztasuna zehazten da. Bihurgailuak irakurketa egokiak egin ditzan, ezin du 200KHz-ko maiztasuna baina handiagoa izan. Beraz, erabili beharreko *prescaler* minimoa 128koa da (16MHz/128 = 125KHz).

5.2.1.4. ADCL and ADCH (ADC Data Register)

Moduluak bihurketa bakoitza bukatzen duenean, 10 biteko doitasuna duen balioa erregistro honetan kargatzen da.

5.3. Tenporizadoreak

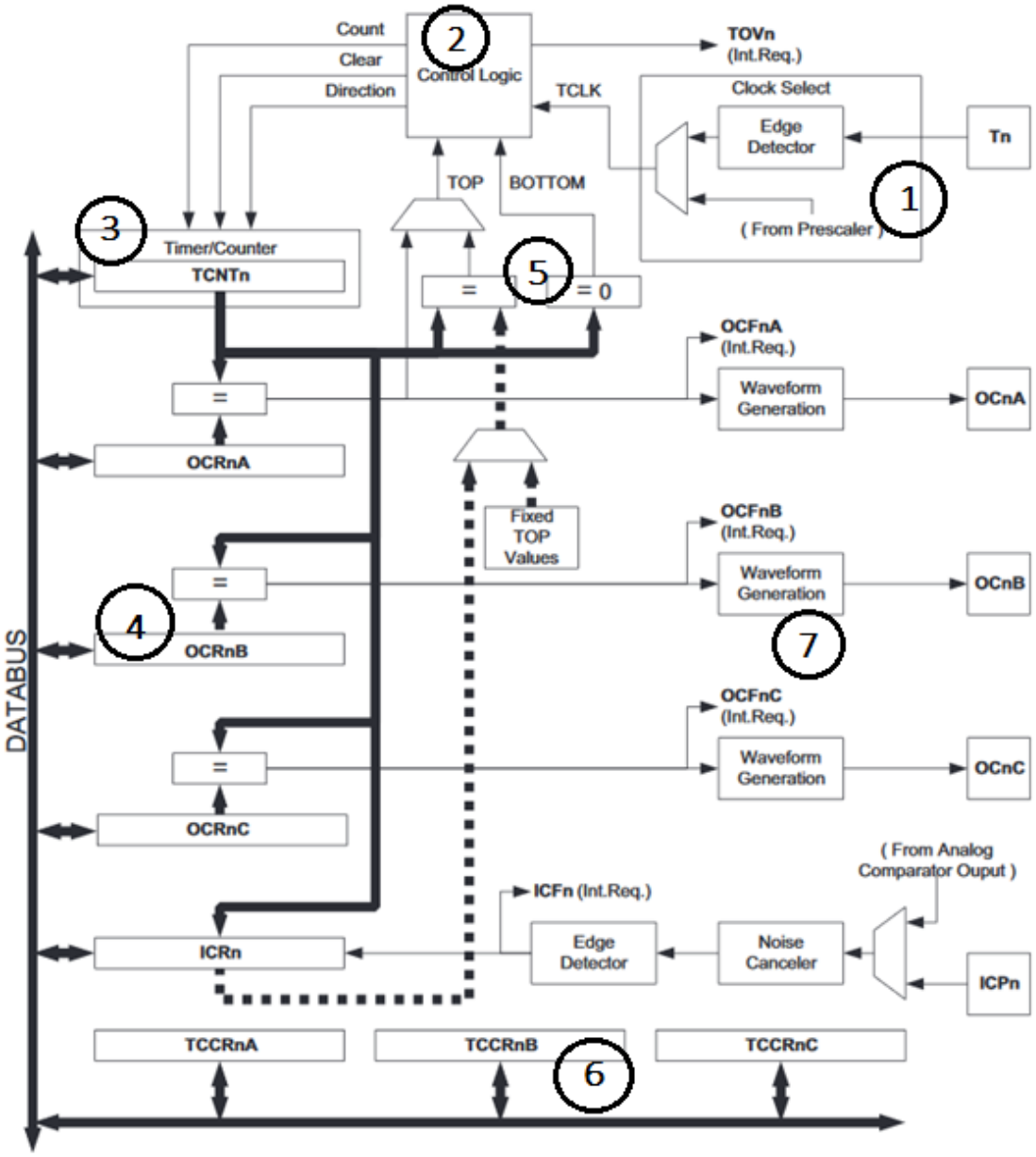
Denborarekiko menpekoak diren funtzioak kontrolatzeko aukera eskaintzen dute. Proiektuan bi dira tenporizadore bidez kontrolatuko diren ekintzak:

- **PWM seinaleen sorrera**: 4. kapituluan ikusi bezala, serbomotorren kontrolerako PWM seinaleak erabili dira. Serbomotorraren posizioa, seinalea aktibatuta mantentzen den denboraren menpekoa da eta tenporizadoreen bidez, seinalea behar adina denboran aktibatu eta desaktibatuko da.
- **Ureztatze-sistema:** lurraren hezetasuna maila baxua denean ureztaketa 4 segundoan zehar egingo da.

5.3.1. Funtzionamendua

Tenporizadoreen funtzionamenduaren xehetasunak aztertzeko, 16 biteko timerra erabiliko da. Horretarako, 5.5 irudiko bloke diagraman atal ezberdinak zenbatu dira, eta jarraian bakoitza zertarako den azalduko da.

1. *Clock Select* zatia erloju-seinalea sortzeaz arduratzen da. Horretarako, sarrera gisa mikrokontrolagailuaren erloju-seinalea hartu eta *prescaler*-aren bidez abiadura aldatzen zaio.
2. *Control Logic*-a denboragailuaren zikloen kontaketa egiteaz arduratzen da. *Clock Select* eremutik datorkion *TCLK* seinalearen goi hertza somatzean, *TCNTn* erregistroaren balioa garbitu, handitu (+1) edo murrizten (-1) du, *TOP* eta *BOTTOM* seinaleetan detektatzen duenaren arabera.
3. *TCNTn* erregistroa, tenporizadorearen zikloen kontaketa gordetzeko erabiltzen da.
4. *OCRnA, OCRnB eta OCRnC* erregistroetan,tenporizadorearen goi-muga zehazten da, *TCNTn* erregistroko datua balio horietako batera iristean, etena sortzen da (gaituta badaude); *TCNTn* erregistroa berrabiarazteko edo PWM seinaleak hardware bidez sortzeko erabil daitekeena. *OCRn* erregistroen etenak bakarka gaitu edo desgaitu daitezke. Proiektuan, tenporizadorearen denboraren kontrola egiteko *OCRnA* erregistroa soilik erabili da.
5. *Control Logic*-era sarrera gisa konektatu diren seinaleak sortzeko atala da. *TCNTn* erregistroa aztertuta, tenporizadorea behe-muga edo goi-mugara iritsi den adierazteko erabiltzen dira. Seinale hauek aztertuta, *Control Logic*-ek tenporizadorea berrabiarazi, balioa handitu edo balioa txikituko du.
6. *TCCRnA*, *TCCRnB* eta *TCCRnC* erregistroak tenporizadorea konfiguratzeko erabiltzen dira (4.4.3 atalean aztertuko dira sakonki).
7. Tenporizadoreek PWM seinaleak hardware zein software bidez sortzeko aukera eskaintzen dute.



5.5 irudia: 16 bit-eko tenporizadorearen bloke-diagrama

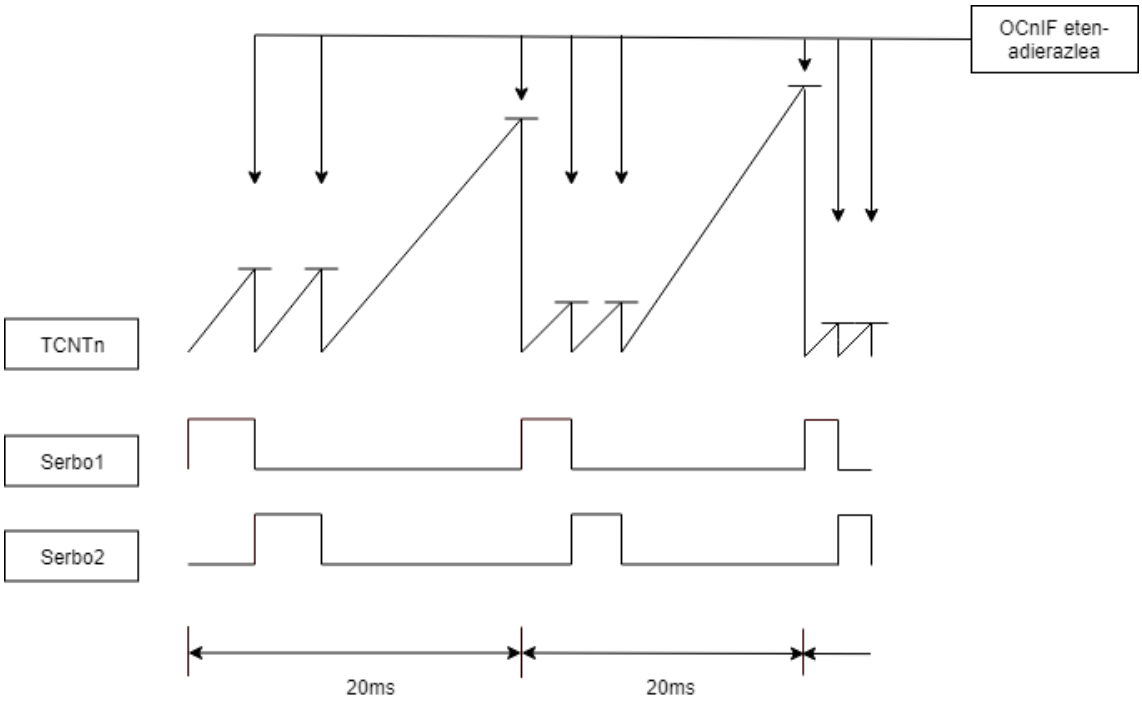
5.3.2. PWM seinalea

Tenporizadoreak PWM seinaleak sortzeko metodologia ezberdinak eskaintzen baditu ere, proiektuan *CTC (Clear Timer on Compare Match)*[1] erabili da. Modu honetan, tenporizadorearen kontagailua *OCRnA* erregistroarekin bat datorrenean, etena sortzen da.

5.6 irudian ikus daitekeen moduan, hainbat serbomotorren kontrola sekuentzialki egin behar da. Lehenik, lortu nahi den posizioari dagokion denboran zehar (*duty zikloa),* PWM seinalea serbomotorra konektatuta dagoen hankatxoan aktibatzen da. Horretarako, *OCRnA* erregistroan, *duty* zikloaren kontaketa balioa kargatzen da eta *TCNTn* erregistroak balio hau hartzen duenean, etena sortzeko. Etena sortzen denean, PWM seinalea nahiko denboran aktibatuta egon dela adierazten du eta ondorioz, hankatxotik serbomotorrera bidaltzen den seinalea desaktibatzen da.

Etenaren zerbitzu errutinan, bigarren serbomotorrarentzako prozesu bera jarraitzen da, lortu nahi den posizioari dagokion balioa *OCRnA* erregistroan kargatzen da, PWM seinalea bigarren serbomotorraren hankatxoan denbora-tarte horretan zehar aktibatzeko.

Bigarren serbomotorraren *duty* zikloa amaitutakoan, PWM seinalea desaktibatzen da eta serbomotorren periodoa (20ms) errespetatzeko falta den denbora tartea itxarongo da.



5.6 irudia: CTC modua

5.3.3. Tenporizadoreen erregistro-multzoa

Tenporizadoreak konfiguratzeko kudeatu beharreko erregistroak zein diren azalduko da. ADC moduluaren antzera, ez da gehiegi sakonduko aukera guztiak azalduz. Xehetasun gehiago ezagutzeko dokumentazioa[1] kontsultatzea egokiagoa da.

Erregistro-multzoa aztertzeko erabiliko den tenporizadorea 1.a da, beraz, konfiguratu nahi den tenporizadorearen identifikadorearengatik ordezkatu behar da 1 balioa.

5.3.3.1. TCCR1A – Timer/Counter 1 Control Register A

Bit	7	6	5	4	3	2	1	0
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	COM1C1	COM1C0	WGM11	WGM10
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- COM1A[1:0] – 0000: *OC1A* hankatxoaren portaera definitu. Serbomotorren kontrola softwarez egingo denez, hankatxo horiek desaktibatuta mantenduko dira, sarrera/irteera digital arrunt moduan lan egiteko.
- COM1B[1:0] – 0000: *OC1B* hankatxoaren portaera definitu. Aurreko kasuan aipatu bezala, deskonektatuta mantenduko da.
- COM1C[1:0] – 0000: *OC1C* hankatxoaren portaera definitu. Aurreko kasuan aipatu bezala, deskonektatuta mantenduko da.
- WGM1[1:0] – 00: Tenporizadorearen kontaketa sekuentzia aldatzeko erabiltzen bada ere, kasu honetan, kontaketa normala izango da (banan banan kontaketa gorakorra), eta beraz, bitak desaktibatuta mantendu dira.

5.3.3.2. TCCR1B – Timer/Counter 1 Control Register B

Bit	7	6	5	4	3	2	1	0
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	0	0	0

- ICNC1 – 0: sarrera seinalea leuntzeko eta interferentziak deuseztatzeko modulua aktibatzen da. Interferentziarik espero ez denez, desgaituta mantenduko da.
- ICES1 - 0: ICPn erregistroa kontagailuaren goi edo behe muga izateko gaitu. Proiektuan, *OCR1A* erabiliko da etenak sortzeko eta beraz, ez da zertan goi mugarik definitu behar.
- WGM1[3:2] - 00: tenporizadorearen kontaketa sekuentzia hautatu. Kontaketa normala eignge da proiektuan.
- CS1[2:0] - 010: erloju-seinalearen maiztasuna moldatzeko *prescalerra* konfiguratzeko eremua. PWM pultsuak sortzeko denboragailuan, adibidez, 8ko *prescalerra* erabili da. Mikrokontrolagailuaren maiztasuna 16MHz-ekoa da. Serbomotorren periodo guztia kontuan hartuta, 20ms-ko denbora-tarteak onartu behar dira, beraz:

$$zikloko = \frac{1\ \text{ziklo}}{16 * 10^6 Hz} = 62.5ns/ziklo$$

$$serbomotor\ periodioa\ (20ms) = \frac{20 * 10^6 ns}{62.5\ ns} = 320000\ ziklo$$

1 tenporizadorea 16 bitekoa da, eta ondorioz onartzen duen balio maximoa $2^{16} - 1 = 65535$ da. Hori dela eta, beharrezkoa da tenporizadorearen maiztasuna lehenik moldatzea, ziklo kopurua kontatu ahal izateko. 20ms kontatzeko erabili daitekeen prescaler txikiena, 8koa da:

$$prescalerra\ erabilita\ kontatu\ beharreko\ zikloak = \frac{320000}{8} = 40000\ ziklo$$

5.3.3.3. TIMSK1 – Timer/counter 1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	OCIE1C	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- ICIE1 – 0: hankatxoren batean aldaketa egotearen ondorioz sortzen diren etenak baimentzeko. Proiektuan desaktibatuta mantendu da.
- OCIE1C- 0: *TCNT1* eta *OC1C* erregistroa bat datozenean etena sortu.
- OCIE1B - 0: *TCNT1* eta *OC1B* erregistroa bat datozenean etena sortu.
- OCIE1A - 1: *TCNT1* eta *OC1A* erregistroa bat datozenean etena sortu. Denboragailuetan erabilitako erregistroa *OCIE1A* soilik izan da.
- TOIE1 - 0: kontagailua goi-mugara iristean etena sortu.

5.4. USART

Berotegiaren kontrola urrunekoa izan dadin, interfazearen eta berotegia kontrolatzeko erabilitako Arduino Mega 2560ren arteko komunikazioa Wi-Fi bidez burutu da. Arduino Mega 2560k, ordea, ez du sarera konektatzeko modulurik, eta ondorioz, bien artean bitartekari lanak egingo dituen kanpo gailu bat beharrezkoa da. Zehazki, 3. Kapituluan aipatu den ESP32 txartela da erabili dena.

ESP32 txartelak, interfazetik datozkion aginduak Arduino Mega 2560 txartelera bidaltzen ditu eta agindu hauei erantzunez txartelak bidalitako datuak interfazera bideratuko ditu (kasu batzuetan interfazetik datuak ere bidaltzen dira txartelera).

Muturren arteko komunikaziorako bi komunikazio modu desberdin erabili dira. Batetik, interfazearekin eta ESP32 txartelaren artean Wi-Fi konexioa gauzatzen da, UDP socketen bidez. Bestetik, ESP32 eta Arduino Mega 2560 txartelaren arteko komunikazioa U(S)ART moduluaren bidez gauzatu da.

Atal honetan, ESP32 eta Arduino Mega 2560 txartelaren arteko komunikazioa nola burutzen den azalduko da.

U(S)ART moduluen artean informazioa seriean bidaltzen da eta helburu txartelean, jasotako datuak prozesatzen hasi aurretik, byte osoa jaso arte itxaroten da.

U(S)ART moduluak, komunikazioak modu sinkrono zein asinkronoan egitea ahalbidetzen du:

- Komunikazio asinkronoa (UART): komunikatuko diren bi moduluek beren barne-periodoa erabiltzen dute datuen transmisio abiadura zehazteko, beste muturrean dagoen gailua kontuan hartu gabe. Periodo sinkrono faltagatik, beharrezkoa da bidaltzen den byte edo datu bakoitzeko, hasiera eta amaiera bitak bidaltzea. Byte bakoitzeko beraz, gutxienez 10 bit bidaltzen dira eta horrek, eraginkortasuna murrizten du. Sinkronizazio seinalerik ez dagoenez, komunikazioa hasi aurretik ezinbestekoa da bi moduluek aurrez definituta izatea zein transmisio-abiadurarekin komunikatuko diren.
- Komunikazio sinkronoa (USART): bi kable berezitu erabiltzen dira transmisioan; datuak alde batetik, eta sinkronizazio seinalea bestetik. Sinkronizazio seinalearen bidez, igorleak datua osatzen duten bit bakoitza kablean noiz dagoen irakurtzeko prest adierazten du. Hartzaileak sinkronizazio-seinalean gorako hertza antzematean, datu-kablean dagoen bita gordetzen du. Metodo hau asinkronoa baino eraginkorragoa da, ez baita hasiera eta amaiera bitik behar bidaltzen den byte bakoitzeko.

Aipatu berri den bezala, komunikazio sinkronoan datu-transmisioa eraginkorragoa bada ere, ESP32 txartelak komunikazio asinkronoa erabiltzera behartzen du, *UART* modulua soilik baitu instalatuta.

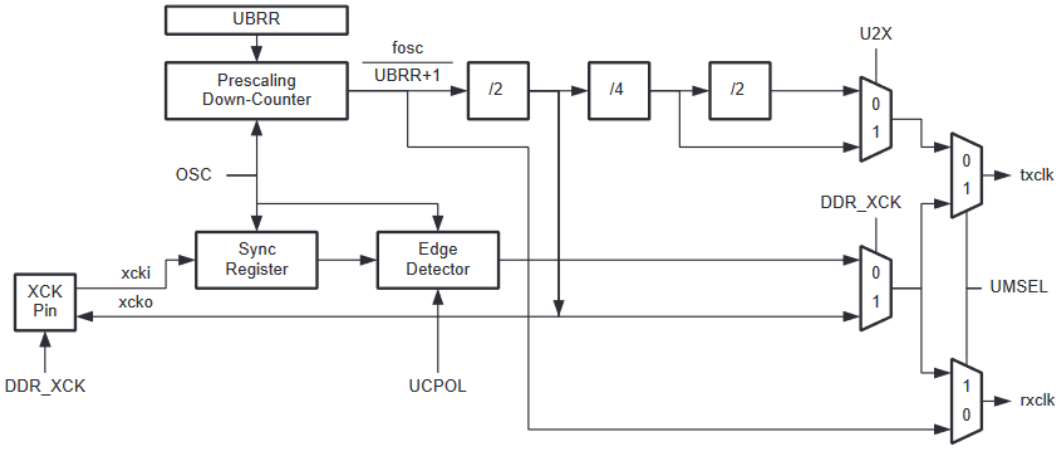
UART moduluan 2 bloke nagusi bereizten dira:

- Hartzailea: datuak jaso zain dagoen atala da eta etengabe hasiera seinalea iritsi den edo ez aztertzen du. Hasiera seinalearen jarraiki datozen bitak, desplazamendu-erregistroan gordetzen ditu byteak osatu arte. Behin erregistroan byte osoa kargatuta, etena sortuko du, eta hala konfiguratuta badago, etenari dagokion zerbitzu-errutina exekutatuko da.
- Igorlea: hargailuaren kontrakoa egiten du, bidalketa desplazamendu-erregistroan sistemak datuak idatzi bezain pronto *start bit*a bidaltzen du, ondoren datua bitez bit igortzen ditu eta azkenik, s*top bit*a.

5.4.1. Barne-abiaduraren konfigurazioa

5.7 irudian, USART moduluaren barne-abiadura zehazteko sistema ikus daiteke. Abiadura *UBRR* erregistroan kargatutako balioaren eta *UBRR* erregistroa sarrera modura konektatzen zaion kontagailu beherakor baten bidez zehazten da. Kontagailua 0 baliora iristen den unean, ziklo bat sortzen du, UART moduluak erloju-seinale gisa erabiliko duena. Kontagailuaren erloju-seinalea, Arduino Mega 2560 txartelaren maiztasunak zehazten du (f_{osc}), eta beraz, UBRR erregistroan kargatu beharreko balioa, txartelaren maiztasunak baldintzatzen du.

Barne-erloju seinale hori zuzenean konektatzen da hartzailera; igorlearen kasuan ordea, seinalearen abiadura 2, 8 edo 16 aldiz murrizten da lehenik, *U2X* erregistroan hautatu den konfigurazioaren arabera.



5.7 irudia: barne erloju-seinalea sortzeko bloke-diagrama

5.8 taulan, UART modulua konfiguratu den arabera lortzen den transmisio-abiadura eta *UBRR* balioaren arteko erlazioa agertzen dira. *U2X* erregistroan 0 kargatuz gero, modu normala erabiltzen da, zeinean kontagailuak sortutako barne-erlojua 16 aldiz murrizten den. *U2X* erregistroan 1 kargatuz gero, aldiz, 8 aldiz murrizten da.

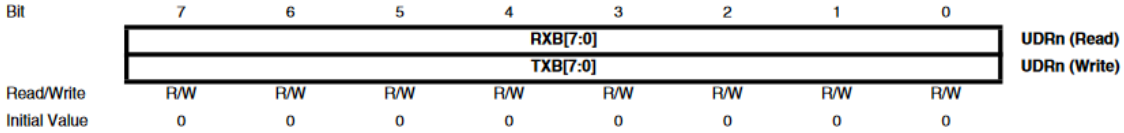
Lan-modua	Baud rate-a kalkulatzeko ekuazioa	UBRR balioa kalkulatzeko ekuazioa
Modu normala	$BAUD = \frac{f_{osc}}{16(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{16BAUD} - 1$
Abiadura bikoitza	$BAUD = \frac{f_{osc}}{8(UBRR + 1)}$	$UBRR = \frac{f_{osc}}{8BAUD} - 1$

5.2 taula: UBRR kalkulua

5.4.2. USART moduluaren erregistro-multzoa

Atmega2560 mikrokontrolagailuko USART modulua programatzeko hainbat dira konfiguratu beharreko erregistroak. Kasu honetan, aurreko zenbaitetan adierazi den bezala, *n* balioak hainbat modulu daudela adierazi nahi du eta programatu nahi den moduluaren zenbakiarengatik ordezkatu behar da.

5.4.2.1. UDRn – USART I/O Data Register n



8 biteko bidalketa (*TXB*) eta hargailu (*RXB*) datu-erregistroak. Igorlearen kasuan, datu bat transmititzeko beharrezkoa da *TXB* erregistroa guztiz hutsa egotea.

5.4.2.2. UCSRnA – USART Control and Status Register n A

Bit	7	6	5	4	3	2	1	0
	RXCn	TXCn	UDREn	FEn	DORn	UPEn	U2Xn	MPCMn
Read/Write	R	R/W	R	R	R	R	R/W	R/W
Initial Value	0	0	1	0	0	0	0	0

- RXCn – 0: *RXB* erregistroan irakurri gabeko datuak daudela adierazten du. Moduluaren hasieraketa egiten denean, daturik oraindik ez dagoela adierazi da.
- TXCn – 0: bidalketa gauzatu denean aktibatzen den bita. Hartzailearen antzera, moduluaren hasieraketa egitean, transmisiorik egin ez dela adierazi da.
- UDREn – 0: bidalketa erregistroa hutsik dagoen edo ez adierazten du bitak. Hasieraketan hutsa ez dagoela adierazi bada ere, automatikoki aktibatuko da bita, baldin eta bidalketa erregistroa hutsik badago.
- FEn – 0: datu okerrak jaso direnean aktibatzen den seinalea.
- DORn– 0: hartzailearen erregistroa beteta badago eta *start* bita jasotzen bada, seinale hau aktibatzen da. Hasieran, arazorik ez dagoenez, 0 balioarekin hasieratu da.
- UPEn – 0.: transmisioan gertatutako interferentzien ondorioz jaso diren datuak okerrak direla adierazten du.
- U2Xn – 1: transmisio-abiadura bikoitza konfiguratzeko. Honek zuzenean eragiten dio, erloju-zikloak sortzeko erabili den *UBRR* erregistroan kargatu beharreko balioari.
- MPCMn – 0: prozesadore anitzen arteko komunikazio modua aktibatu eta desaktibatzeko. Proiektuko komunikazioan ez da horrelakorik behar.

5.4.2.3. UCSRnB – USART Control and Status Register n B

Bit	7	6	5	4	3	2	1	0
	RXCIEn	TXCIEn	UDRIEn	RXENn	TXENn	UCSZn2	RXB8n	TXB8n
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R	R/W
Initial Value	0	0	0	0	0	0	0	0

- RXCIEn – 1: datuak jasotzean etenak sortzea baimendu. Proiektuan gaituak egongo dira.
- TXCIEn – 1: datu bidalketa burutzean sortutako etenak baimendu. Proiektuan gaituak.
- UDRIEn – 0: *UDRE*n erregistroa husten denean etena sortzea ekidin.
- RXEN – 1: hartzailea gaitu.
- TXEN – 1: igorlea gaitu.
- UCSZn2 – 0: karaktere bakoitzaren/datu-blokeen tamaina definitzeko bita. 0 balioarekin, karaktereen tamaina byte-ekoa izango da.
- RXBn – 0: 9 biteko datuekin lan egin behar denean aktibatu beharreko bita (hartzailea). Desgaituta mantendu da.
- TXBn – 0: 9 biteko datuak lan egin behar denean aktibatu behar da (igorlea). Aurrekoaren antzera, desgaituta mantendu da.

5.4.2.4. UCSRnC – USART Control and Status Register n C

Bit	7	6	5	4	3	2	1	0
	UMSELn1	UMSELn0	UPMn1	UPMn0	USBSn	UCSZn1	UCSZn0	UCPOLn
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
Initial Value	0	0	0	0	0	1	1	0

- **UMSELn[1:0] – 00:** USART moduluaren aukerak hautatu. 00 balioarekin komunikazio asinkronoa gaitu da.
- **UPMn[1:0] – 00:** errore detekziorako paritate bitik ez da erabiliko.
- **USBSn – 0:** transmisio amaiera adierazten duten bit kopurua hautatu (stop bit). 0 balioarekin, stop bit bakarra hautatu da.
- **UCPOLn – 0:** modu sinkronorako soilik erabiltzen da.

5.4.2.5. UBRRnL and UBRRnH – USART Baud Rate Registers

Bit	15	14	13	12	11	10	9	8
	–	–	–	–	UBRR[11:8]			
	UBRR[7:0]							
	7	6	5	4	3	2	1	0
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W

- **UBRR[11:0]:** *UBRR* balioa gordetzeko erregistroa.

6

Interfaze grafikoa

Erabiltzaileak modu intuitibo batean berotegia kontrolatzeko interfazeak eskaintzen dituen aukera ezberdinak zein diren azalduko da. Gainera, interfazea nola egituratuko den aipatuko da.

6.1. Sarrera

Erabiltzaileak berotegia modu intuitibo batean kudeatu dezan, ordenagailuan exekutatuko den interfaze grafikoa garatu da.

Interfazeak jasotako funtzionalitateak ondorengoak dira:

- Berotegiaren egoeraren informazioa pantailaratu: lurraren hezetasuna, giro tenperatura eta atearen, leihoaren eta haizagailuaren egoeraren berri ematen du.
- Berotegiko elementuen eskuzko kontrola: berotegia modu automatikoan kontrolatzen bada ere, erabiltzaileak edozein momentutan eskuz kontrola ditzake atearen eta leihoaren posizioa. Ureztatze-sistema ere aktiba dezake eta haizagailua martxan jartzeko eta itzaltzeko aukera du.
- Datuen historikoa: tenperatura eta hezetasun balioak 30 segundoan behin gordetzen dira datu-basean. Erabiltzaileak interfazearen bidez datu horiek eskuragarri ditu, edozein unetan kontsultatzeko. Informazio hau interesgarria da uzta eta landareetan kalterik sumatzen bada, arazoaren jatorria non dagoen detektatzeko.
- Landare hautaketa: funtzionalitate honi esker, lurraren hezetasun minimoa eta giro tenperatura, berotegian landatu den landare motarentzat optimizatzen dira. Erabiltzaileak bisualki, landare berrientzako parametro optimoak gehitu ditzake datu-basean, hortik aurrera, landare mota hau landatzen den kasuetan erabili ahal izateko.
- Erabiltzaile kudeaketa: landare hautaketa egiteak ondorio larriak ekar ditzake, baldin eta hautatutako landarearen eta benetan landatuta dagoenak kontrako beharrak badituzte. Egoera hori ekiditeko, landare hautaketa bistara sartzen den pertsonak erabiltzaile kontu bat behar du.

Interfazearen egitura 4 atal nagusitan banatu da:

- Erabiltzaileak eskuz botoiak sakatzea: edozein momentutan erabiltzaileak berotegiko gailuen kontrola hartzeko, ekintza bakoitzeko botoi bat kokatu da interfazean. Horietako bat sakatzen den aldiro, agindu bat bideratuko du ESP32 eta interfazearen arteko komunikaziorako sortutako socket-aren bidez.
- Aginduak modu periodikoan bidaltzea: interfazeak berotegiaren hainbat parametroren informazioa modu antolatu batean erabiltzaileari aurkeztea du helburu. Informazio hau oso aldakorra da eta ondorioz, etengabe eguneratu behar da interfazean pantailaratutakoa. Horretarako, berotegiko elementuen egoeraren inguruko informazioa eskatzeko definitu diren aginduak periodikoki bidaliko dira ESP32 txartelera, eta honek, Arduino Mega 2560 txartelera bideratuko ditu.
- Aginduen erantzunak jasotzea: erabiltzaileak eskuz bidalitako eskaerak zein periodikoki bidaltzen diren aginduei erantzuten dien mezuak jasotzeaz arduratuko da.
- Datu-basearekin elkarrekintza: zenbait kasutan datu-basearekin elkarrekintza erabiltzaileak botoien bat sakatzean gertatuko bada ere, periodikoki (30s-ro) sentsoreetatik jaso den informazioa datu-basean gordetzen da.

6.2. Egitura

Interfazean gertatzen diren ekintza guztiak modu antolatuan kudeatzeko, logikoki guztiz banandu da interfazea, helburu bakoitzerako prozesu ezberdinak sortuta. Desberdindu diren prozesuak hauek dira:

- Prozesu nagusia: interfazea abiarazi eta erabiltzaileak botoiren bat sakatzean, interfazean aldaketa sortuko duena. Sakatutako botoia berotegia kontrolatzeko aginduren bat bada, ESP32 txartelera eskaera bideratuko du.
- Datu periodikoen eskaera prozesua: berotegiko elementuen egoera bistan datuak eguneratuta mantentzeaz arduratuko da. Horretarako, berotegiko gailuen egoera kontsultatzeko aginduak bidaltzen dizkio periodikoki ESP32 txartelari.
- Bidalitako aginduen erantzunak jasotzeko prozesua: Arduino Mega 2560 txartelak zehaztuta eta ESP32ren bidez bideratu diren datuak jaso eta prozesatzeaz arduratuko da.
- Datuen historikoaz arduratzen den prozesua: 30 segundoan behin tenperatura, hezetasuna eta hautatuta dagoen landarea zein den datu-basean gordetzen du.

Prozesu bakoitzak helburu finko eta independente bat izan arren, ezinbestekoa da elkarren artean komunikazioa izatea, elkarren datuak kontsumitu behar baitituzte. Adibidez: erabiltzaileak *Irten* botoia sakatzen duenean, 4 prozesuak eten behar dira. Erabiltzaileak sakatutako botoia, prozesu nagusiak soilik antzematen du eta ondorioz, bere betebeharra, gainerako prozesuei eten agindua bidaltzea da.

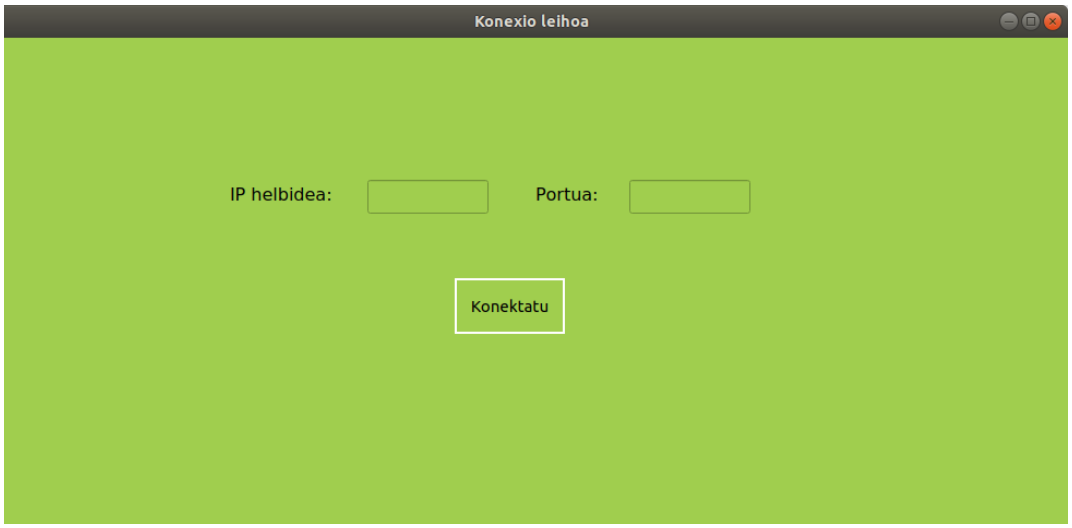
Prozesuen arteko komunikazioa gauzatzeko, bi baliabide erabili dira:

- Hobiak: bi prozesuren artean komunikazio bidea zabaltzen da. Mutur batean, irakurketa egingo da eta bestean, idazketa. Hobiak, adibidetzat jarri den kasua ebazteko erabili dira. Prozesu nagusiak, *Irten* botoia sakatu dela antzematean, gainerako prozesuekin komunikatzen duen hobietan, “exit” mezua idazten du eta gainerako prozesuek mezu hau irakurtzen dutenean, exekuzioa etengo dute.
- Memorian mapatutako fitxategia: interfazean pantailaratzen diren datuak eta datu-basean gordetzen direnak eguneratuta izateko, datuak jasotzeaz arduratzen den prozesuak, gainerako prozesuentzat eskuragarri jarri behar ditu. Hau, memorian mapatuta dagoen fitxategi baten bidez egin da. Memoria fitxategi hori, prozesu guztiek dute atzigarri. Fitxategian idatzitako datu bakoitza zein agindu edo magnituderi dagokion identifikatzeko, agindu bakoitzarentzat posizio jakin bat erreserbatu da. Honela, prozesuek daturen bat behar dutenean, nahikoa dute irakurri nahi duten datuari dagokien posiziotik irakurketa hastea.

6.3. Menuen deskribapena

Interfazea funtzionala izatea lortu nahi izan da, eta erabiltzailearentzat ulerterraza bada ere, itxura ez da gehiegi landu. Jarraian, interfazearen funtzionalitate bakoitzarentzat inplementatu den bista ezberdinak aurkeztuko dira.

- **Konexio leihoa (6.1 irudia):** harrera leiho honen bidez, ESP32 txartelarekin konexioa gauzatzen da. Horretarako, ESP32ri esleitu zaion IP helbidea eta socketa zein portutan atzigarria den zehaztu behar da.

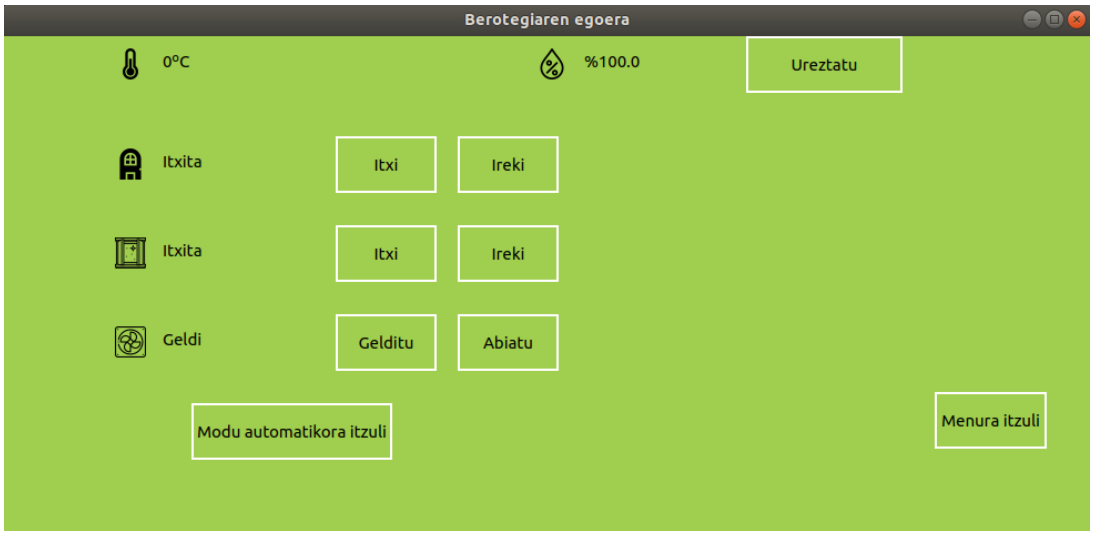


6.1 irudia: konexio panela

- **Menu nagusia (6.2 irudia):** Interfazearen bidez eskaintzen diren funtzionalitate guztiak jasotzen dituen bista da. Aukeran dauden bista eta funtzionalitateak honako hauek dira: berotegiaren egoera ikusi, datuen historikoa ikusi, berotegiaren konfigurazioa hautatutako landare motara moldatzeko bista eta azkenik, erabiltzaile kudeaketarako bista.

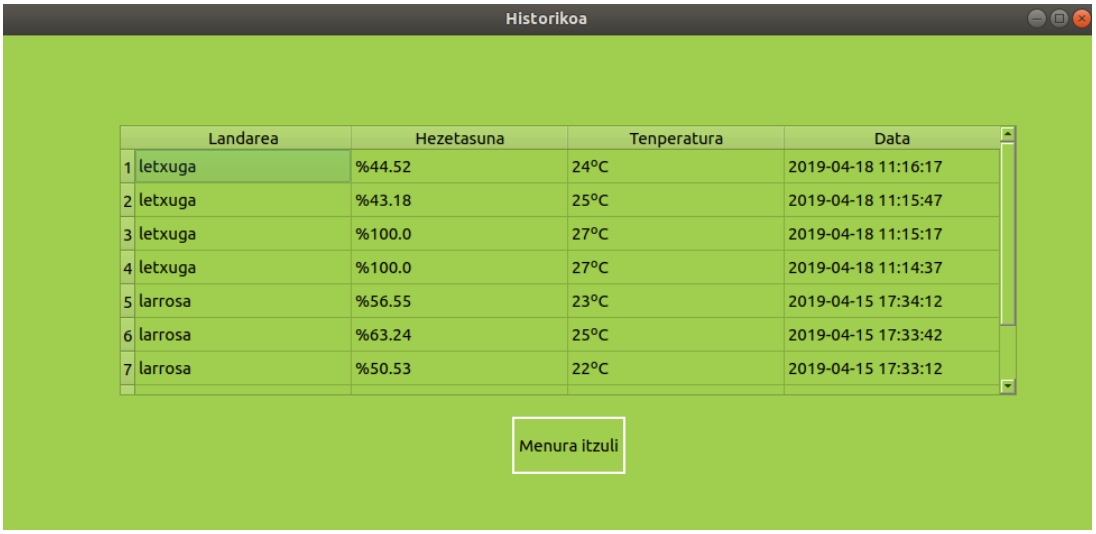
6.2 irudia: menu nagusia

- **Berotegiko gailuen egoera (6.3 irudia):** berotegia kontrolatzeko panel nagusia da, informazioa pantailaratzen da eta eskuzko kontrola egiteko botoiak ditu.



6.3 **irudia:** berotegiko gailuen egoera panela

- **Historikoa (6.4 irudia):** denboran zehar erregistratu diren lurraren hezetasun eta tenperatura balioak pantailaratzeko bista.



6.4 **irudia:** datuen historikoa

- **Landare mota hautatu (6.5 irudia):** berotegiaren parametroak landare zehatz baterako optimizatu daiteke. Bista honen bidez, landare berriak sortu eta berotegiko balioak zein landaretarako optimizatuko diren hautatu daiteke.

6.5 **irudia:** landare kudeaketa panela

- **Erabiltzaileak kudeatu (6.6 irudia):** landare aldaketa egiteko baimena izango duten pertsonentzat erabiltzaile berriak sortzeko eta jadanik existitzen direnak ezabatzeko.

6.6 irudia: erabiltzaile kudeaketa panela

7

Komunikazioa

Kapitulu honetan, berotegiaren urruneko kontrola eta datuen pantailaratzea ahalbidetzeko sortu den komunikazio-protokoloaren agindu definizio erregelak eta eskaintzen diren agindu guztiak sailkatuko dira. Agindu ezberdinen artean dagoen lotura ere aipatuko da (bidalketa-erantzuna).

7.1. Sarrera

Komunikazioa esan bezala, bi fasetan burutzen da. Batetik, interfazeak ESP32ra aginduak Wi-Fi bidez bidaltzen ditu; eta bestetik, ESP32k, agindu horiek berak UART moduluaren bidez Arduino Mega 2560 txartelera bideratzen ditu.

7.2. Sare-protokoloa

Wi-Fi bidezko komunikazioa gauzatzeko, socket-ak erabili dira. Socket-ak 2 programen artean datu-trukea ahalbidetzen duten baliabidea dira. Normalean programa horiek gainera, ordenagailu ezberdinetan exekutatzen dira. Socket-en datu-trukea TCP/IP protokoloaren gainean egiten da, eta beraz, beharrezkoa da socket bakoitzari IP helbide bat eta portu zenbaki bat esleitzea (ordenagailuko IP eta portu helbide bat). IP helbideak, ordenagailua identifikatzen du eta portuak, ordenagailuan exekutatzen ari den programa.

Komunikazioak bezero-zerbitzari egitura du; bezeroak datu eskaerak egiten ditu eta zerbitzariak, eskaera hauei erantzuten die. Proiektuan zerbitzari lana egingo duen elementua ESP32 txartela da, zeinak interfaze grafikoak (bezeroa) egindako eskaerak jaso eta erantzun bat emango dien, aurretik Arduino Mega 2560rekin komunikatu ondoren.

Sare mailan komunikazioak bideratzeko 2 protokolo existitzen dira: TCP eta UDP. 7.1 taulan, bien arteko konparaketa[11] egingo da.

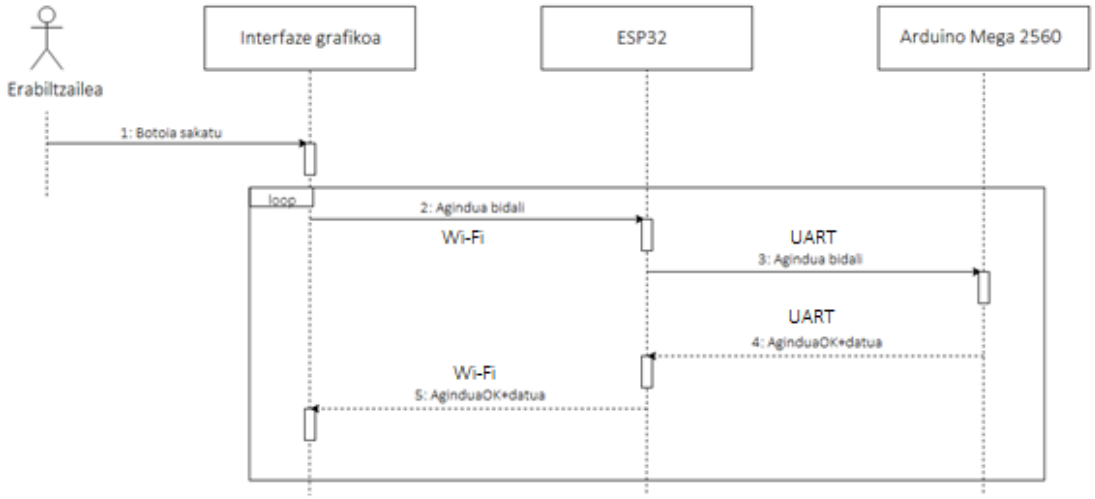
	TCP	UDP
Paketeen ordenaketa	Ordenatuta jasotzen dira	Ez da ordenarik errespetatzen.
Abiadura (pakete tamainak eraginda)	UDP baino mantsoagoa.	TCP baina azkarragoa.
Fidagarritasuna	Paketeak ordenan, osorik eta interferentziarik gabe jasoko direla bermatzen da.	Ez da paketeak iritsi izana bermatzen.
Burukoaren tamaina	20 bit.	8 bit.
Data jarioa	Data jarioa jarraitua da. Ez da tamaina zehatzetan zatitzen.	Datuak tamaina finkoko paketetan antolatzen dira.
Errore detekzioa	Bai.	Ez.
Zertarako erabilgarria	Datuen fidagarritasuna ezinbestekoa eta transmisio denbora kritikoa ez den kasuetarako.	Abiadura handiko konexioetara eta datuen fidagarritasuna kritikoa ez den kasuetarako.

7.1 taula: UDP eta TCP konparaketa

Proiektuan UDP erabiltzea erabaki da, komunikazioak dituen ezaugarriengatik. Komunikatuko diren 2 muturrak, bat bestearengandik gertu daude, beraz, transmisioan paketeak galtzeko arriskua txikia da. Gainera, paketeak desordenan iristeak ez du arazorik sortuko, datu bakoitza bestearekiko independentea baita. Gainera, paketeren bat galtzeak ez du eragozpenik sortuko, aplikazioak exekuzioan jarrai dezake eta aginduak periodikoki bidaltzen direnez, datua berriz ere eskuratuko da. Hau ikusita, datuen fidagarritasuna ez da ezinbestekoa eta komunikazioa arina

izatea komenigarriagoa da. Gainera, informazioa paketetan bidaliko denez, prozesaketa fasea erraztuko du.

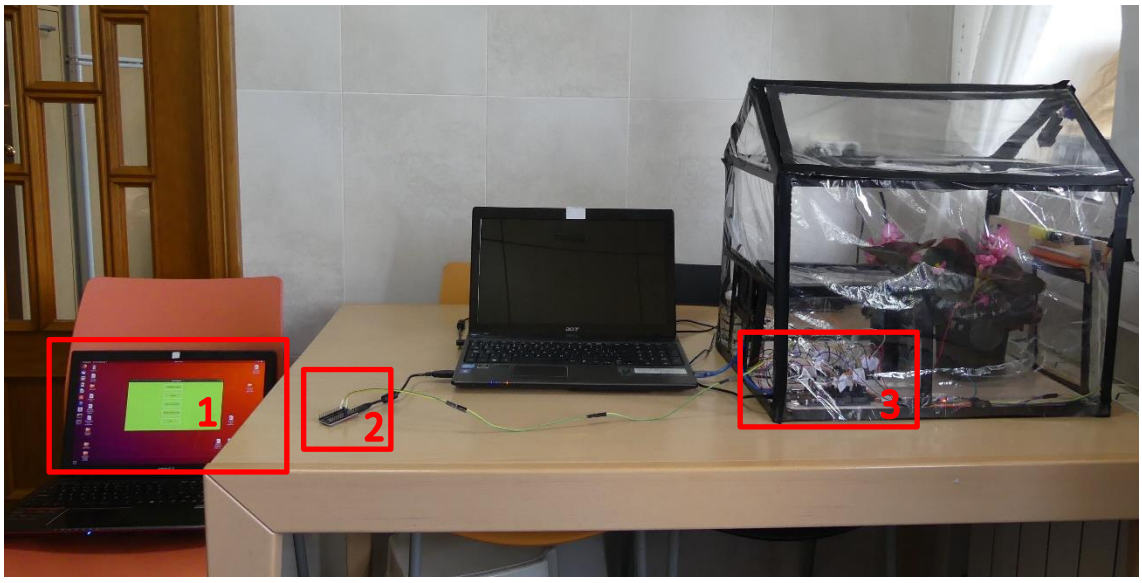
7.1 irudian komunikazioak dituen bi faseak bereiz daitezke.



7.1 irudia: sekuentzia-diagrama

Proiektu errealaren itxura, 7.2 irudian ikus daiteke argiago:

- 1 laukitxoan, interfazea exekutatzeko erabili den ordenagailua dago, Wi-Fi sarera soilik konektatu dena eta Arduino Mega 2560 txartelarekin komunikazio zuzenik ez duena.
- 2 laukitxoan ESP32 txartela dago, USB bidez bigarren ordenagailuari konektatuta. Ordenagailua ESP32k bideratzen dituen aginduak pantailaratzeko eta txartela elikatzeko soilik erabili da.
- 3 laukitxoan, Arduino Mega 2560 txartela dago, ESP32 txartelera kableekin konektatuta eta gailu ezberdinak kontrolatzeko beharrezko diren konexio guztiekin.



7.2 irudia: sistema erreala

Interfazearen bidez erabiltzaileari eskaintzen zaizkion aginduek berotegian eragin dezaten, beharrezkoa da lehenik agindu bakoitzari, modu unibokoan identifikatuko duen kode bat esleitzea. Kode horretako agindu bakoitza, komunikatuko diren muturreko bi sistemek interpretatu behar dute, behar bezalako erantzuna eman ahal izateko.

Agindu hauek helburu ezberdinak izan baditzakete ere, proiektu honetan 3 dira definitu agindu mota nagusiak:

- **Ekintza bat exekutatuko dutenak:** berotegiaren gailuak kontrolatzeko erabiliko dira.
- **Datuak eskatzeko aginduak:** berotegiko elementuen egoeraren berri izateko pentsatuak. Bidalitako agindu bakoitzak, eskatutako parametroaren datuekin osatutako erantzun bat espero du.
- **Berotegiaren parametroak aldatzekoak:** berotegiak modu automatikoan funtzionatzen duenean, tenperatura eta hezetasun mugak zehazteko aginduak.

7.3. Agindu-multzoa

Protokoloaren garbitasuna, ulergarritasuna eta homogeneotasunaren bila, oinarrizko baldintzak definitu dira aginduen izendapen, erantzun eta datuen formatuan:

- Agindu-izenak ahalik eta deskriptiboen izan behar dute.
- Agindu-izenak 3 letrako akronimoak dira.
- Agindu guztiek erantzun bat espero dute.
- Itzulera mezuek zein aginduri erantzuten dien erraz identifikatzeko, “aginduAkronimoaOK+” formatua errespetatuko dute.
- Datuak “+” ikurraren ondoren bidaliko dira, eta 5 digituko balioak izan behar dute.

7.2 taulan, komunikazio-protokoloa osatzen duten agindu-multzoa definituko da. Datuei dagokionez, xxxxx benetako balioarekin ordezkatzen da.

Agindua	Deskribapena	Erantzuna
WIO	Leihoa zabaldu.	WIOOK+
WIC	Leihoa itxi.	WICOK+
WIS	Leihoaren uneko egoera kontsultatu.	WISOK+xxxxx [00000]: Leihoa itxita dago. [00001]: Leihoa zabalik dago.
HUS	Uneko hezetasuna zein den kontsultatu.	HUSOK+xxxxx [0-1023]
HUM+xxxxx	Ureztatze automatikoa aktibatzeko hezetasun minimoa aldatu.	HUMOK+
TES	Uneko tenperaturaren balioa kontsultatu.	TESOK+xxxxx

TEM+xxxxx	Berotegian onartzen den tenperatura maximoa zehaztu.	TEMOK+
DOO	Atea zabaldu.	DOOOK+
DOC	Atea itxi.	DOCOK+
DOS	Atearen posizioa kontsultatu.	DOSOK+xxxxx [00000]: Atea itxita dago. [00001]: Atea zabalik dago. [00002]: Atea mugitzen ari da.
FAO	Haizagailua martxan jarri.	FAOOK+
FAC	Haizagailua gelditu.	FACOK+
FAS	Haizagailuaren egoera kontsultatu.	FASOK+xxxxx [00000]: Haizagailua itzalita dago. [00001]: Haizagailua martxan dago.
DAC	Gailu guztiak automatikoki kontrolatzera itzuli.	DACOK+
WAO	Landareak 4 segundoan zehar ureztatu.	WAOOK+
WPP+xxxxx	Hautatutako landarea zehaztu.	WPPOK+
WPS	Hautatutako landarea zein den kontsultatu.	WPSOK+xxxxx

7.2 taula: komunikazio-protokoloa

8

Inplementazioa

Azkenik, proiektua osatzen duten atal bakoitzerako inplementatutako kodearen egitura zein den azalduko da. Ez da xehetasun zehatzegirik tratatuko, kapituluaren helburua, garatu diren programen egitura zer nolakoa den argitzea da.

8.1. Arduino Mega 2560 kode egitura

5. kapituluan modulu bakoitza nola konfiguratu den azaldu da. Atal hau, fitxategien arteko sinkronizazioan eta fitxategi bakoitzean definitutako funtzioetan zentratuko da.

8.1 irudian ikusten den legez, fitxategi nagusia *main.c* da. Fitxategi honetan programaren sinkronizazioa gauzatzen dela esan daiteke, aldagai gehientsuen aldaketak, honen baitan baitaude.

Kontrolatu beharreko parametro guztien egoera zehazteko aldagai guztiak, *data.c* fitxategian mantendu dira.

main prozedura nagusian, modulu ezberdinek datuak prest noiz dituzten jasoko da eta hori gertatzean, datuak dagozkien aldagaian kargatuko ditu. *main* programa honek kargatzen dituen datuak, tenperatura, lurraren hezetasuna eta atearen egoera dira.

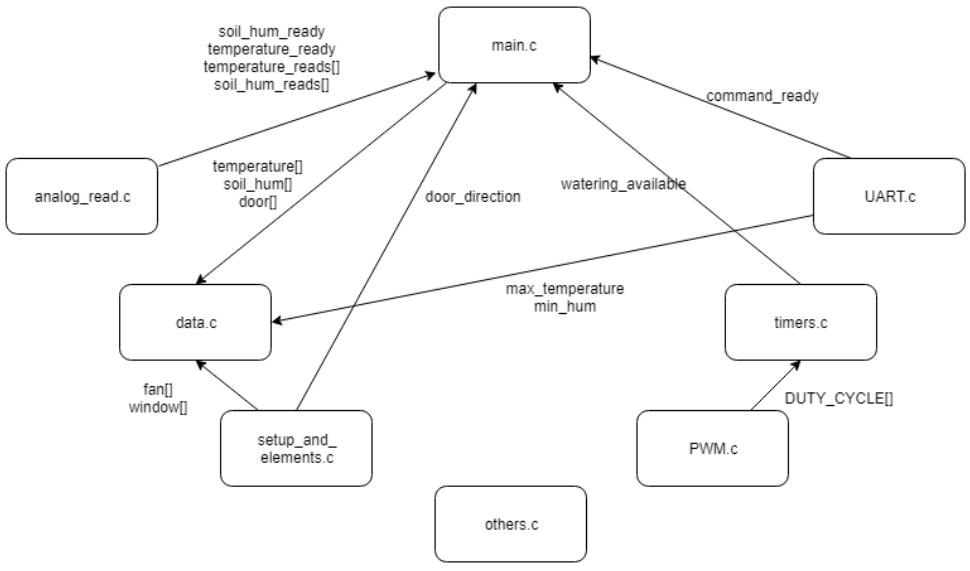
Sinkronizazio aldagaiak *main* programara bideratzen dira, modulu bakoitzak datuak prest noiz dutuen adierazten dutenak. Irakurketa analogikoak egiteaz ADC modulua arduratzen da. Bihurketak amaitzen dituenean, hezetasun eta tenperatura balioak prestatzen ditu eta *main* programari datuak prest dituela jakinarazteko *soil_hum_ready* eta *temperature_ready* aldagaiak aktibatzen ditu. UART moduluak ere, balizko agindua jasotzen duenean *main* programak prozesatu ditzan, *command_ready* aldagaia aktibatzen du.

watering_available aldagaiarekin bi ureztaketen artean gutxienez 30s pasa direla bermatzen da, eta *door_direction* aldagaiak, atearen zein noranzkotan mugitzen ari den adierazten du.

Datu aldagaiek bi fitxategitatik aldatzen dira.

Batetik, *setup_and_element.c* fitxategian leihoa eta atearen kontrolerako funtzioak daude, eta bertan aldatzen dira *fan* eta *window* aldagaiak.

Bestetik, *UART.c* fitxategian definitutako zerbitzu-errutina, interfazetik datozkion datu eta aginduak jasotzeaz arduratzen da. Datu horiek hautatutako landarea eta landarera egokitutako tenperatura maximoa eta hezetasun minimoa zehazten dute. Jasotako dautak, *main* programan baldintza gisara erabiliko dira (*max_temperature* eta *min_hum)* hainbat elementuren egoera zehazteko.



8.1 irudia: berotegiaren kontrol egitura

8.1.1. Funtzioen deskribapena

Jarraian fitxategi bakoitzean aurki daitezkeen funtzioen deskribapen labur bat emango da.

- PWM.c:
  - *serbo_middle():* leihoa erdiko posizioan jartzen du.
  - *open_window():* leihoa zabaltzeko.
  - *close_window():* leihoa ixteko.
  - *load_duty:* balio hamartarretan zehaztutako balioa serboa kontrolatzeko tenporizadorean kargatu behar den baliora bihurtzeko.

- Analog_read.c:
  - *measure_soil_hum_and_temp()*: hezetasun sentsoreak eta tenperatura sentsorea konektatuta dauden hankatxoak sarrera gisa definitzen ditu. ADC bihurgailua, 5. Kapituluan aipatu den konfigurazioarekin hasieratzen du eta lehen irakurketak 0 kanalean hasten ditu.
  - *ISR(ADC_vect)*: ADC moduluaren zerbitzu-errutina da hau. Irakurketa prozesuan, sentsore edo seinalean izan zitekeen erroreak baztertzeko asmotan, irakurketak 5 aldiz egiten dira kanalez aldatu aurretik.

    Lurraren hezetasuna kalkulatzeko bi YL-69 sentsoreen batezbestekoa erabiltzen da. Hezetasun balioak eskuratu direla adierazteko *soil_hum_ready* aldagaia ez da aktibatuko, guztira 10 irakurketa egin arte, sentsore bakoitzeko 5 irakurketa.

    Hezetasunaren antzera, tenperatura prest dagoela adierazteko aldagaia *temperature_ready* da.

    Aldagai horiek *main()* funtzioan inkesta bidez zelatatzen dira eta balio bakoitza dagokion aldagaian kargatzen da.

- Data.c: Berotegiko elementuen egoera gordetzeko aldagaien definizio fitxategia da.
- Greehouse_elements.c:
  - *setup_pins():* leihoa kontrolatzeko serbomotorren, atea ireki eta ixteko motorraren, haizagailuaren eta ur-bonbaren hankatxoak irteera moduan eta karrera amaierako sakagailuak sarreera moduan hasieratzen ditu.
  - *stop_door():* atea gelditzeko funtzioa, horretarako norabidea kontrolatzeko erabili diren irteera pinetan 0 balioa esleitzen da.
  - *close_door():* atea ixteko funtzioa.
  - *open_door():* atea irekitzeko funtzioa.
  - *activate_fan():* haizagailua martxan jartzen du.
  - *stop_fan():* haizagailua itzaltzen du.
  - *water_plants():* landareak ureztatzen ditu 4 segundoan zehar.
  - *stop_watering_plants():* ureztatze-sistema eteten du.

- *wait_between_watering():* ureztaketa bakoitzaren artean 30 segundoko itxarote tartea ezartzen du.

- Others.c: funtzio laguntzailez osatua dago.
  - *int_to_char(int value, char * result):* integer motako balioak, char motako bektore batera bihurtzen ditu. Funtzio hau, sentsoreetatik eskuratu diren balioak, UART modulutik bidaltzeko datu motara itzultzeko erabiliko da.
  - *uarray_to_int(unsigned char * index, int total_digits):* aurrekoaren kontrako funtzioa, karaktere katetik, int motako balioa lortzen da.
  - *calculate_median(int values[], int value_count):* balio analogikoak kalkulatzeko sentsoreetatik 5 balio hartzen dira, interferentzia edo irakurketa akatsak ekiditeko. Calculate_median funtzioaren bidez, 5 balio horien batezbestekoa kalkulatzen da.
  - *global_interrupts_enable():* sistema mailako etenak gaitzeko.
  - *global_interrupts_diable():* sistema mailako etenak desgaitzeko.
  - *calculate_temperature(int analog_value):* tenperatura sentsorearen irakurketen batezbestekoari Steinhart funtzioa aplikatzeko.

- UART.c: funtzio laguntzailez osatua dago.
  - *setup_UART():* UART modulua konfiguratzeko.
  - *send_UART(unsigned char * message, int letter_count): message* aldagaiko testua UART moduluaren bidez bidaltzen du.
  - *ISR(USART0_RX_vect):* jasotako informazioa aztertzeko zerbitzu-errutina. Jasotako agindua balizkoa bada, *command_ready* aktibatzen du, *main* programak prozesatu dezan. Aginduarekin lotuta dauden datuak jasotzeaz ere arduratzen da.

- Timers.c: ureztatze-sistema eta serbomotorren kontrolaz arduratzen da.
  - *timer_1_PWM() + ISR(TIMER1_COMPA_vect):* serboen posizioaren kontrolerako tenporizadorea.
  - *timer_3_wait_between_watering(int s) + ISR(TIMER3_COMPA_vect):* ureztatze jarraien artean pasa beharreko denbora minimoa kontrolatzen du.
  - *timer_4_move_PWM() + ISR(TIMER1_COMPA_vect):* serboaren posizioa mantsoa izateko behar diren elementuak kontrolatzen ditu.
  - *timer_5_wait_water_plants(int s) + ISR(TIMER5_COMPA_vect):* ureztatzeak 4 segundo iraun ditzan inplementatua.
- Main.c: datuen eta berotegiaren egoera aztertzeko funtzio nagusia du.

8.2. Interfaze grafikoaren kode egitura

Interfazearen egituran ere, goi mailako deskribapena egingo da, atal bakoitzak beste zein atalekin elkarrekintza duen argi ikusi ahal izateko. Deskribapenaren ulermenari laguntzeko 8.2 irudian eskema agertzen da.

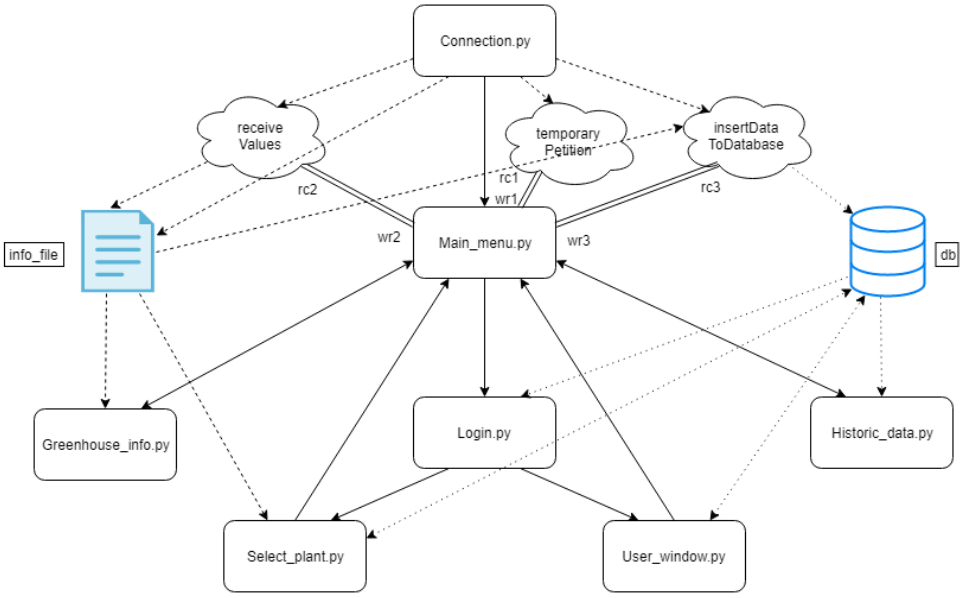
Interfazearen exekuzioa Connection.py-n hasten da, zeinak erabiltzaileak ESP32ra konektatzeko parametroak zehaztu behar dituen bista abiarazten duen. Hasi botoia sakatzen den unean, 3 prozesu ume sortzen dira: datu-basean datuak periodikoki gordetzen dituena, berotegiaren berri izateko aginduak etengabe bidaltzen dituena, eta bidalitako aginduei eman zaien erantzunak jasotzeaz arduratzen dena. Gainera, prozesu horietako bakoitzak konexio zuzena izango du hobien bidez programaren menu nagusiarekin. Hori, programa amaitzean prozesu umeak ere exekuzioa amai dezaten egin da (ikus 6.2 atala).

Jasotako datuak prozesu guztiek eskuragarri izan dezaten, memorian mapatutako *info_file* fitxategia sortu da. Fitxategi hau, balioak jasotzen ari den prozesuak idatziko du, eta pantailaratze prozesuak zein datu-basean berotegiaren egoera gordetzen duen prozesuak irakurriko dute.

Datu-basean gordetzen den informazioa 3 motakoa izan daiteke. Berotegiaren egoera deskribatzen duten *info_file* fitxategitik irakurritako datuak, landare-mota berriak sortzeko informazioa eta erabiltzaileek saioa has dezaten behar den informazioa.

Informazio hau baliatuko duten bistak 4 dira:

- **Login:** sartu den erabiltzaile eta pasahitza egokiak direla ziurtatzeko.
- **Select_plant:** landare berrien parametroak modu egonkorrean datu-basean gordetzeko zein datu-basetik ezabatzeko, eta datu-basean daudenen artean, landatu dena hautatzeko bista.
- **User_window:** erabiltzaile berriak datu-basean gehitzeko eta zaharkituak geratu direnak ezabatzeko.
- **insertDataToDatabase:** bigarren planoan exekutatzen ari den prozesuak periodikoki *info_file* fitxategitik irakurritako datuak datu-basean gordetzeko.



8.2 irudia: interfazearen egitura

8.2.1. Funtzioen deskribapena

Interfazearen fitxategi eta erabili diren azalpen laburra emango da jarraian. Deskribapen zehatzago bat nahi izanez gero, ikus B eranskina.

- Connection.py: ESP32rekin konexioa hasteko eta prozesuak exekuzioan jartzeko.
    - *initializeGreenhouseInfo():* datuen bidalketaz, balioak jasotzeaz eta datu-basean datuak txertatzeaz arduratuko diren prozesuak martxan jartzen ditu, menu nagusia zabaltzeaz gain.
    - *temporaryPetition(self, s, rc1):* berotegiko elementuen egoera kontsultatzeko aginduak ziklikoki bidaltzeko funtzioa.
    - *receiveValues(self, info_file, s, rc2):* Arduino Mega 2560k sortutako erantzunak jasotzeko prozesua. Datuak espero dituen agindu bakoitzaren erantzuna, *info_file* fitxategian gordetzen du.
    - *insertDataToDatabase(self, db, info_file, rc3):* hautatuta dagoen landarearen identifikadorea, uneko tenperatura eta lurraren hezetasuna datu-basean 30 segundoz behin gordetzeko.
- Greenhouse_info.py: berotegiaren elementuen egoeraren berri emateko interfazea pantailaratzeko fitxategia.
    - *sendCommand(self, command, s, info_file):* erabiltzaileak sakatutako botoiari dagokion agindua bidaltzen du.
    - *updateGUI(self, info_file):* interfazea eguneratu.
    - *goToMenu(self, GreenhouseWindow, socket, db, info_file, wr1, wr2, wr3):* menu nagusira itzultzeko funtzioa.
- Historic_data.py: datu-baseko datuak pantailaratzeko funtzioak jasotzen ditu.
    - *showData(self, db):* datuen historikoa datu-basetik lortu eta pantailaratzen du.
- Login.py: erabiltzaile eta pasahitza sartzeko bista pantailaratzeko.
    - *loginUser(self, LoginWindow, socket, db, info_file, wr1, wr2, wr3, window_to_open):* erabiltzailea datu-basean existitzen dela egiaztatzen du.
- Main_menu.py: eskaintzen diren aukera ezberdinak jasotzen dituen menua.
    - *openGreenhouseWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2, wr3):* berotegiko elementuen egoera erakusten duen pantaila erakutsi.
    - *openDatabaseWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2, wr3):* datuen historikoa erakusteko pantaila erakutsi.
    - *openLoginWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2, wr3, window_to_open):* erabiltzaile eta pasahitza sartzeko pantaila bistaratzeko.
    - *exit(self, MainMenuWindow, wr1, wr2, wr3):* programa exekutatzean sortu diren prozesu ume guztiei 'exit' mezua bidaltzen die hobien bidez, horien exekuzioa eteteko.

- Select_plant.py: landare berriak sortzeko eta berotegiaren parametroak landare zehatz batentzat optimizatzeko behar diren funtzioak ditu.
    - *showSelectedPlant(self, db, socket, info_file):* hautatu den landarearen izena pantailaratzeko.
    - *insertPlantDatabase(self, db):* landare berria datu-basera gehitzen du.
    - *deletePlantDatabase(self, db, info_file):* hautatutako landarea datu-basetik ezabatzen da.
    - *selectPlant(self, db, socket):* berotegiaren konfigurazioa hautatutako landarerako optimizatzeko erabilia.
    - *humtoADC(self, humidity):* Arduino Mega 2560 plakak bidalitako balio analogikoa ehunekoetara moldatzeko funtzioa.
    - *showPlants(self, db):* datu-baseko landareak pantailaratzen ditu.
- User_window.py: erabiltzaileak kudeatzeko bista pantailan erakusteko funtzioak ditu
    - *createUser(self, db):* erabiltzaile berria datu-basean gehitzen da.
    - *deleteUser(self, db):* hautatutako erabiltzailea datu-basetik ezabatzeko.
    - *showUser(self, db):* datu-baseko erabiltzaileak modu antolatu batean pantailaratzen ditu.

9

Ondorioak

Dokumentuari amaiera emango dion kapitulu honetan, proiektuaren inguruko azken hausnarketa egingo da. Gehien bat, erabilitako teknologien inguruan hausnartuko da, eta proiektuak etorkizunean izan ditzakeen hobekuntza batzuk planteatuko dira.

9.1. Ondorioak

Erabilitako txartela eta teknologiak proiektuarentzako egokiak direla ondoriozta daiteke, hasieran planteatuko funtzionalitate guztiak garatzea lortu baita.

Atmega2560 mikrokontrolagailuaren kasuan, moduluen barne funtzionamendua ulertu da. Hala ere, harrigarria izan da Atmega2560k hain modulu gutxi izatea, guztiz ezberdinak diren eta Arduino plataforma erabiltzen duten proiektu ugari baitaude sarean. Hori dela eta, Arduinoren barne egitura askoz ere konplexuagoa izatea espero zen.

Erregistro-mailan lan egiteko jarraitu beharreko prozedura plataforma ezberdinetan oso antzekoa dela ondorioztatu da, eta beraz, *sistema txertatuen diseinua* ikasgaian ikasitako kontzeptuak, beste arkitektura batean aplikagarriak direla ikusi da.

Behe-mailan lan egiteak dituen abantailen eta desabantailen inguruan hausnartzeko ere balio izan du proiektuak. Erregistro-mailan programatzea kasu berezietan da erabilgarria: denbora errealeko sistemetan adibidez, beharrezkoa da erantzun-denborak berehalakoak izatea eta hori lortzeko bidea da programazio metodo hau. Horretaz gain, hain ezaguna eta landua ez dagoen edozein plataforman aplikatu daitezke landutako kontzeptuak, eta ondorioz, proiektuak garatzeko plataformen aukeraketa zabalagoa da eta beharretara egokien moldatzen den soluzioa aurkitzea errazten du.

Desabantaila nagusiena, kodearen eramangarritasun eza da. Erregistro zein hankatxoen txartel zehatz baterako soilik da erabilgarria. Gainera, elementu bakar bat kontrolatzeko *bit*-ekin lan egiteak, arazoak sor ditzake mantenu garaian, kodea guztiz argia ez izanez gero, kontrolatu beharreko erregistro guztien artean baten bat ahaztea oso posible da.

Sareko protokolo estandarizatuek eskaintzen dituzten abantailak egiaztatu dira. Helburu zehatz baterako agindu-multzoa sortzeak eta prozesatzeak zailtasun berezirik ez duela ikusi da, tartean protokolo estandarizatuek informazio-trukea gailuen arteko komunikazioa posible egiten dute eta beraz, protokoloa exekutatuko den plataformatik oso bereizita dago.

Interfaze grafikoari dagokionez, Qt bezalako lan-inguruneek eskaintzen duten erraztasuna ikusi ahal izan da. Modu bisualean sortutako interfazea Python bezalako programazio-lengoaiara automatikoki itzultzeak, interfazearen erabilgarritasuna asko zabaltzen du, lengoaia horretako liburutegiak erabiltzea ahalbidetzen baitu (proiektuaren kasuan, datu-basearen integrazioa sinplea izaten lagundu du).

Azkenik, modu independentean ikusitako kontzeptuak elkartuz, ekosistema bat sor daitekeela ondorioztatu da. Honek, protokolo estandarren garrantzia nabarmendu du, plataformarekiko independentea den komunikazio-bidea sortzea posible egin baitute.

Proiektu bat gauzatzearen ikuspuntu orokorrago batetik, hasieran plangintza oso zehatz bat definitzearen garrantzia azpimarratzekoa da. Lortu nahi diren helburuak eta hori lortzeko erabili beharreko teknologia eta gailuak ikertzea ezinbestekoa dela ikusi da. Denbora estimazioak eta epeak definitzean ere, hausnarketa sakona egiteak proiektuaren garapenean asko laguntzen du, desbiderapen nabarmenak antzematen direnean erantzuteko denbora nahikoa izatea ezinbestekoa baita. Horrekin lotuta, proiektuan sor daitezkeen arazoak aldez aurretik identifikatzeak eta horien kontrako plana definitzeak desbiderapen hauek murrizten laguntzen du. Proiektuan zenbaitetan blokeo egoera gertatu da momenturen batean, aurreikusi zitekeen arazoari aurre nola egin erabaki behar izan baita.

9.2. Etorkizunerako ideiak

Proiektua lantzeko denbora-muga errespetatzearren, hobekuntza moduan planteatu diren hainbat aukera inplementatzeko geratu dira.

- Berotegia berotzea: tenperatura maximora iristean gertatzen denaren antzera, landare bakoitzak tenperatura minimoa izatea inplementa daiteke. Uneko tenperatura landarearentzat egokia den tenperatura minimoa baina baxuagoa izanez gero, berotze sistema martxan jarri.

- Errezelak zabaltzea: fotorresistentzia baten bidez neurtutako argitasunak muga landarearen gaindituz gero, errezelak landareen gainean zabaltzeko sistema bat.

- Korreo elektronikoak bidaltzea: erabiltzaileak zehaztutako denbora-tarte batean balio arriskutsuak mantentzen badira eta sistemak emandako erantzunak eraginik ez dutela antzematen bada, berotegiko jabeari mezu elektroniko baten bidez abisatzea.

- Lurraren hezetasuna aldakorra izatea: orain inplementatu den sistemak, hezetasuna minimotik behera dagoenean etengabe ureztatzen ditu landareak (ureztaketen artean definitutako denbora utziz). Zenbaitetan lurra pixka bat lehortzea onuragarria izan daiteke landareentzat. Inplementatu beharrekoa aldaketa hau ziklikoki egitea litzateke, hau da, egindako ureztaketa kopuru batera iristean, hezetasuna minimotik behera izate onartzea denbora-tarte batez.

- Berotegitik kanpo dagoen eguraldi estazioa: eguraldi estazioaren bidez, hurrengo eguneko eguraldiaren aurreikuspena egingo litzateke eta aurreikuspen horren arabera, berotegiak bere sistemak martxan jartzea edo ez erabaki zezakeen. Adibidez eta aurreko funtzionalitatearen arira: hezetasuna minimotik behera badago baina euria espero bada, landareek ez dute zertan ura beharko, beraz, egun horretan ez da ureztatzea erabakiko du.

Bibliografia

[1] Atmel ATmega640/V-1280/V-1281/V-2560/V-2561/VATmega2560 datu-orria. [2019/06/17]

http://ww1.microchip.com/downloads/en/devicedoc/atmel-2549-8-bit-avr-microcontroller-atmega640-1280-1281-2560-2561_datasheet.pdf

[2] Oyvind Nydal Dahl. (2018/12/05). What is an H-Bridge? [2019/06/17]

https://www.build-electronic-circuits.com/h-bridge/

[3] Overview of ESP32 features eta ESP32 datu-orria. [2019/06/17]

https://www.exploreembedded.com/wiki/Overview_of_ESP32_features._What_do_they_practically_mean%3F

https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

[4] Atmel Studio 7. [2019/06/17]

https://www.microchip.com/mplab/avr-support/atmel-studio-7

[5] Qt. [2019/06/17]

https://www.qt.io/

[6] PyQt. [2019/06/17]

https://www.ecured.cu/PyQt

https://riverbankcomputing.com/software/pyqt/intro

https://pypi.org/project/SIP/

[7] What is Micropython eta Flash/Upload MicroPython Firmware to ESP32. [2019/06/17]

https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/

https://randomnerdtutorials.com/flash-upload-micropython-firmware-esp32-esp8266/

[8] MySQL. [2019/06/17]

https://www.mysql.com/

[9] Arduino. [2019/06/17]

https://es.wikipedia.org/wiki/Arduino

https://www.arduino.cc/en/products.compare

[10] Arduino Port Registers. [2019/06/17]

https://www.arduino.cc/en/Reference/PortManipulation

[11] The difference between TCP and UDP. [2019/06/17]

https://support.holmsecurity.com/hc/en-us/articles/212963869-What-is-the-difference-between-TCP-and-UDP-

A Eranskina: mikrokontrolagailuko kodea

main.c

```
#define F_CPU 16000000UL //Txartelaren maiztasuna zehaztu

#include "timers.h"
#include "UART.h"
#include "analog read.h"
#include "others.h"
#include "PWM.h"
#include "data.h"
#include "greenhouse_elements.h"

#include <avr/io.h>
#include <avr/iom2560.h>

int main(void)
{
	int door_locked_by_user = 0, fan_locked_by_user = 0, window_locked_by_user =0;
	int median_temp = 0, median_soil_hum = 0;
	float real_temp = 100;		//Tenperaturaren balioa hasieran 100 da.

	global interrupts enable();	//Sistema mailako etenak gaitu.
	initialize values();		//Datuak hasieratu
	setup_pins();			//Hankatxoak konfiguratu.
	setup_UART();			//UART modulua konfiguratu
	setup serbo();			//Serbomotorrak konfiguratu
	measure soil and temp();	//Tenperatura eta hezetasun irakurketa hasi.
	timer 1 PWM();			//PWM tenporizadorea martxan jarri
	timer_4_move_PWM();		//PWM mantxo egiteko tenporizadorea martxan.

	while (1)
	{
		/* Hezetasuna eta tenperatura */
		if (soil_hum_ready) //Hezetasun balioak prest daude.
		{
			//Bi sentsoreetatik irakurri diren 10 datuen batezbestekoa
			//kalkulatu
			median soil hum = calculate median(soil hum_reads, 10);
			//soil hum aldagaian batezbestekoa gorde.
			int to char(median_soil_hum, soil_hum);
			soil_hum_ready=0;
		}
		if (temperature_ready)//Tenperatura balioak prest daude
		{
			//Irakurritako balioen batezbestekoa kalkulatu
			median temp = calculate median(temperature reads, 5);
			//Balio analogikotik tenperatura erreala lortu.
			real_temp = calculate_temperature(median_temp);
			//Balio analogikotik tenperatura erreala lortu.
			int to char((int)real temp, temperature);
			temperature_ready = 0;
		}

			/* UART */
			if (command_ready)	//Balizko agindu bat jaso da
			{
				//'WIO' leihoa zabaltzeko agindua jaso da.
				if (received_command[0]=='W' &&
				    received command[1]=='I' &&
				    received_command[2]=='O')
				{
					//Leihoaren kontrola sistema automatikoak ez
					//izateko blokeatu, erabiltzaileak ireki baitu.
					window locked by_user = 1;
					//Leihoa zabaldu
					open window();
					//'WIOOK+' mezuarekin erantzun UART
					//moduluaren bidez.
					send_UART("WIOOK+", 6);
				}
```

```
                        //'WIC' leihoa ixteko agindua jaso da.
                        else if (received command[0]=='W' &&
                                 received command[1]=='I' &&
                                 received_command[2]=='C')
                        {
                               //Leihoaren kontrola sistema automatikoak ez
                               //izateko blokeatu, erabiltzaileak itxi baitu.
                               window locked by_user = 1;
                               //Leihoa itxi.
                               close_window();
                               //'WICOK+' mezuarekin erantzun UART
                               //moduluaren bidez.
                               send_UART("WICOK+", 6);
                        }
                        //'WIS' leihoaren egoera kontsultatzeko agindua jaso
                        //da.
                        else if (received command[0]=='W' &&
                                 received command[1]=='I' &&
                                 received_command[2]=='S')
                        {
                               //'WISOK+leiho_egoera' erantzuna bidali.
                               send_UART("WISOK+", 6);
                               send_UART(window, 5);
                        }
                        //'TES' tenperatura balioa kontsultatzeko agindua
                        //jaso da.
                        else if (received_command[0]=='T' &&
                                 received command[1]=='E' &&
                                 received_command[2]=='S')
                        {
                               //'TESOK+tenperatura bal' erantzuna bidali.
                               send_UART("TESOK+", 6);
                               send_UART(temperature, 5);
                        }
                        //'HUS' lurraren hezetasuna kontsultatzeko agindua
                        //jaso da.
                        else if (received command[0]=='H' &&
                                 received_command[1]=='U' &&
                                 received_command[2]=='S')
                        {
                               // 'HUSOK+hezetasun balioa' erantzuna bidali.
                               send UART("HUSOK+", 6);
                               send_UART(soil_hum, 5);
                        }
                        //'DOO' atea zabaltzeko agindua jaso da.
                        else if (received command[0]=='D' &&
                                 received command[1]=='O' &&
                                 received_command[2]=='O')
                        {
                               //Atearen kontrola sistema automatikoak ez
                               //izateko blokeatu, erabiltzaileak ireki baitu.
                               door locked by user = 1;
                               //Atea ireki gabe badago, zabaldu.
                               if ( (PINB & _BV(1)) )
                                      open_door();
                               //'DOOOK+' erantzuna bidali.
                               send_UART("DOOOK+", 6);
                        }
                        //'DOC' atea ixteko agindua jaso da.
                        else if (received_command[0]=='D' &&
                                 received command[1]=='O' &&
                                 received_command[2]=='C')
                        {
                               //Atearen kontrola sistema automatikoak ez
                               //izateko blokeatu, erabiltzaileak itxi baitu.
                               door_locked_by_user = 1;
                               //Atea ez dago itxita, beraz itxi.
                               if ( (PINB & _BV(0)) )
                               close door();
                               //'DOCOK+' erantzuna bidali.
                               send_UART("DOCOK+", 6);
                        }
```

```
                        //'DOS' atearen egoera kontsultatzeko agindua jaso
                        //da.
                        else if (received command[0]=='D' &&
                                 received_command[1]=='O' &&
                                 received_command[2]=='S')
                        {
                               //'DOSOK+atearen posizioa' erantzuna bidali.
                               send UART("DOSOK+", 6);
                               send_UART(door, 5);
                        }
                        //'FAO' haizagailua martxan jartzeko agindua jaso
                        //da.
                        else if (received command[0]=='F' &&
                                 received command[1]=='A' &&
                                 received_command[2]=='O')
                        {
                               //Haizagailuaren kontrola sistema automatikoak ez
                               //izateko blokeatu, erabiltzaileak jarri baitu
                               //martxan.
                               fan locked by user = 1;
                               //Bentilagailua abiarazi
                               activate_fan();
                               //'FAOOK' erantzuna bidali.
                               send_UART("FAOOK+", 6);
                        }
                        //'FAC' haizagailua gelditzeko agindua jaso da.
                        else if (received_command[0]=='F' &&
                                 received command[1]=='A' &&
                                 received_command[2]=='C')
                        {
                               //Haizagailuaren kontrola sistema
                               //automatikoak ez izateko blokeatu,
                               //erabiltzaileak gelditu baitu.
                               fan locked by user = 1;
                               //Haizagailua gelditu.
                               stop fan();
                               //'FACOK' erantzuna bidali
                               send_UART("FACOK+", 6);
                        }
                        //'FAS' haizagailuaren egoera kontsultatzeko agindua
                        //jaso da.
                        else if (received command[0]=='F' &&
                                 received_command[1]=='A' &&
                                 received_command[2]=='S')
                        {
                               //'HUSOK+haizagailuaren_egoera' erantzuna
                               //bidali.
                               send UART("FASOK+", 6);
                               send_UART(fan, 5);
                        }
                        //'DAC' berotegiko elementu guztien kontrola
                        //automatikoki egiteko agindua jaso da.
                        else if (received command[0]=='D' &&
                                 received_command[1]=='A' &&
                                 received_command[2]=='C')
                        {
                               //leihoaren, haizagailuaren eta atearen
                               //blokeoa deuseztatu.
                               window locked by user = 0;
                               fan_locked_by_user = 0;
                               door locked by user = 0;
                               //'DACOK+' erantzuna bidali.
                               send_UART("DACOK+", 6);
                        }
                        //'WAO' ureztatzeko agindua jaso da.
                        else if (received_command[0]=='W' &&
                                 received command[1]=='A' &&
                                 received_command[2]=='O')
                        {
                               //landareak ureztatu.
                               water_plants();
                               //'WAOOK+' mezua bidali.
                               send_UART("WAOOK+", 6);
                        }
```

```
                        //Parametroak zein landarerentzat optimizatuta
                        //dauden kontsultatzeko agindua jaso da.
                        else if (received command[0]=='W' &&
                        received_command[1]=='P' &&
                        received_command[2]=='S')
                        {
                                //'WPSOK+landare id' erantzuna bidali.
                                send UART("WPSOK+", 6);
                                send_UART(plant_selected, 5);
                        }
                        //Berotegiko muga temperatura aldatu.
                        else if (received command[0]=='T' &&
                        received command[1]=='E' &&
                        received_command[2]=='M')
                        {
                                //UART moduluari datuak jaso ditzala adierazi.
                                receive data = 1;
                                //'TEMOK+' erantzuna bidali.
                                send_UART("TEMOK+", 6);
                        }
                        //lurraren hezetasun minimoa zehazteko datua jaso da.
                        else if (received_command[0]=='H' &&
                                  received command[1]=='U' &&
                                  received_command[2]=='M')
                        {
                                //UART moduluari datuak jaso ditzala adierazi.
                                receive_data = 1;
                                //'HUMOK+' erantzuna bidali.
                                send_UART("HUMOK+", 6);
                        }
                        //parametroak optimizatzeko hautatu den landarearen IDa
                        //jaso da.
                        else if (received_command[0]=='W' &&
                                  received command[1]=='P' &&
                                  received_command[2]=='P')
                        {
                                //UART moduluari datuak jaso ditzala adierazi.
                                receive_data = 1;
                                //'WPPOK+' erantzuna bidali.
                                send_UART("WPPOK+", 6);
                        }
                        command_ready = 0;
                }


        /*Leihoa, atea eta haizagailua (kontrol automatikoa)*/
        //TenpMax <= Uneko tenperatura < (TempMax +10°C) -> Leihoa zabaldu,
        //haizagailua itzali eta atea itxi
        if (max_temperature <= real_temp && real_temp < max_temperature+10)
        {
                if (!door_locked_by_user && (PINB & _BV(0)))
                        close door();
                if (!window locked by user && window[5] != '1')
                        open_window();
                if (!fan_locked_by_user && (PINC & _BV(1))
                        stop_fan();
        }
        //(TenpMax + 10°C) <= Uneko tenperatura bada, atea zabaldu eta
        //haizagailua martxan jarri.
        else if (max_temperature + 10 <= real_temp)
        {
                if ( !door locked by_user && (PINB & _BV(1)))
                        open door();
                if (!window locked by user && window[5] != '1')
                        open_window();
                if ( !fan_locked_by_user && !(PINC & _BV(1))
                        activate_fan();
        }
```

```
                //Tenperatura parametro normalen barruan dago.
                else
                {
                        if (!door_locked_by_user && (PINB & _BV(0)))
                                close door();
                        if (!window locked by user)
                                close window();
                        if (!fan_locked_by_user)
                                stop_fan();
                }



                //Atea itxi dela adierazten duen sakagailua jo du ateak.
                if (!(PINB & _BV(0)) && door_direction==0)
                {
                        //Atea guztiz itxita dagoela adierazi door aldagaian.
                        int to char(0, door);
                        //Atea gelditu.
                        stop_door();
                }

                //Atea ireki dela adierazten sakagailua jo du ateak.
                else if (!(PINB & _BV(1)) && door_direction==1)
                {
                        //Atea guztiz irekita dagoela adierazi door aldagaian.
                        int to char(1, door);
                        //Atea gelditu.
                        stop_door();
                }
                //Sakagailurik ez badago sakatuta, atea mugimenduan dago.
                else if ( (PINB &  BV(0)) && (PINB &  BV(1)) )
                        //Atea mugimenduan dagoela adierazi door aldagaian.
                        int_to_char(2, door);



                /* Ureztatze-sistema */
                // Hezetasuna landareak behar duena baino gutxiagokoa da
                //(median soil hum balioak gora egiten du gero eta hezetasaun
                //gutxiagorekin).
                if ((median_soil_hum > min_hum) && watering_available)
                {
                        //Landareak ureztatu
                        water plants();
                        //Ureztaketa artean denbora tartea errespetatzeko tenporizadorea
                        martxan jarri
                        wait_between_watering();
                }
        }
}
```

Analog_read.c

```
#include <avr/io.h>
#include <avr/iom2560.h>
#include <avr/interrupt.h>

#include "analog read.h"
#include "UART.h"
#include "timers.h"
#include "others.h"

int analog_port = 0, discard_first = 1;
int soil_index = 0, temperature_index = 0, soil_hum_reads[10], temperature_reads[5],
soil_hum_ready = 0, temperature_ready = 0;

void measure_soil_and_temp()
{
        // A0 (Hezetasun sents.1), A1 (Hezetasun sents.2) eta A2 (Tenperatura sents.)
        //hankatxoak sarrera gisa definitu.
```

```
        DDRF &= 0b11111000;


        ADCSRB &= 0b11110000; //1111: Bit erreserbatuak dauden bezala mantendu.
                              //0: 0 kanalaren hautaketa amaitu.
                              //000: Free running mode hautatu.

        ADCSRA = 0b10011111;  // 1: ADC modulua gaitu
                              // 0: bihurketarik ez hasi oraindik.
                              // 0: eskuz adiraziko da noiz hasi bihurketa.
                              // 1: eten seinalea garbitu.
                              // 1: moduluaren etenak gaitu.
                              // 111:  128ko ADC prescalerra hautatu

        ADCSRA |= _BV(ADSC); //Bihurketa martxan jarri
}

ISR(ADC_vect)
{
        //Kanal aldaketa bat gauzatzen denean, sistema egonkortu behar da lehenengo,
        //ondorioz, lehen irakurketa bartertu behar da.
        if (!discard_first)
        {
                //A0 edo A1 hankatxoetako irakurketa bada, hezetasuna da neurtu dena.
                if (analog_port<2)
                {
                        //Irakurketa balio bektorean kargatu.
                        soil hum reads[soil_index] = ADCW;
                        soil_index++;
                }
                //A2 hankatxokoa irakurri bada aldiz, tenperaturari dagokio.
                else
                {
                        //Irakurketa balio bektorean kargatu.
                        temperature reads[temperature_index] = ADCW;
                        temperature_index++;
                }
        }

        switch (analog_port)
        {
                case 0:
                        //A0 hankatxoan 5 irakurketa egin badira, kanalez aldatu behar
                        //da.
                        if (soil_index==5)                    {
                                ADMUX |=  BV(0); //1 kanala hautatu
                                analog port = 1;
                                discard_first = 1;
                        }
                        else
                                discard_first = 0; //Lehen irakurketa bartertu da,
                                                   //hurrengoak balizkoak dira.
                        break;

                case 1:
                        if (!discard_first && soil_index==10)
                        {
                                ADMUX &= 0xEE; //1 kanala desgaitu
                                ADMUX |=  BV(1); //2 kanala hautatu
                                analog_port = 2;
                                discard_first = 1;
                        }
                        else
                                discard first = 0; //Lehen irakurketa baztertu da,
                                                   //hurrengoak balizkoak dira.
                        break;

                case 2:
                        if (!discard_first && temperature_index==5)
                        {
                                ADMUX &= 0xF0;         //0 kanala hautatu
                                analog_port = 0;
                                discard_first = 1;
                        }
```

```
                        else
                                discard_first = 0; //Lehen irakurketa baztertu da.
                        break;
        }

        if (soil index == 10) //Hezetasun sentsore bakoitzeko 5 irakurketa egin
                              //badira.
        {
                soil hum ready = 1;    //Hezetasun balioak prest daudela adierazi.
                soil_index = 0;        //Hurrengo irakurketak egiteko indizea
                                       berrabiarazi.
        }
        if (temperature index == 5) //Tenperatura sentsoreren 5 irakurketa egin
                                       badira.
        {
                temperature_ready = 1;//Tenperatura balioa prest dagoela adierazi.
                temperature_index = 0;//Hurrengo irakurketak egiteko prestatu.
        }

        ADCSRA |= _BV(ADSC); //Hurrengo irakurketa martxan jarri.

}
```

Greenhouse_elements.c

```
#include <avr/io.h>
#include <avr/iom2560.h>

#include "data.h"
#include "others.h"
#include "greenhouse_elements.h"
#include "timers.h"
#include "PWM.h"

int twice = 0;
int door_direction = 0;

void setup_serbo()
{
        DDRB |= 0b10100000; //11 eta 13 pinak irteera moduan definitu (serbomotorrenen
                            //hankatxoak).
}

void setup_pins()
{
        //34, 35 eta 36 pin-ak irteera gisa hasieratu (Ateko motorraren bi hankatxo-ak
        //eta haizagailua)
        DDRC |= 0b00001111;
        //51 eta 53 hankatxoak-ak sarrera gisa hasieratu (ateko karrera amaiera)
        DDRB &= 0b11111100;
        //22 pin-a irteera bezala hasieratu (ur-bonba)
        DDRA |= 0b00000001;
}

void stop_door()
{
        PORTC &= 0b11110011; //34 eta 35 hankatxoak (atearenak) desaktibatu
}

void open_door()
{
        stop door();
        PORTC |= 0b00000100; //35 hankatxoa (atea zabaltzeko) aktibatu
        door_direction = 1;
}

void close_door()
{
        stop door();
        PORTC |= 0b00001000; //34 hankatxoa (atea ixteko) aktibatu
        door_direction = 0;
}
```

```
void activate_fan()
{
         PORTC |= 0b00000010; //36 hankatxoa (haizagailua) aktibatu
         int_to_char(1, fan); //fan aldagaian haizagailua martxan dagoela adierazi.
}

void stop_fan()
{
        PORTC &= 0b11111101; //36 hankatxoa (haizagailua) desaktibatu
        int_to_char(0, fan); //fan aldagaian, haizagailua geldituta dagoela adierazi.
}

void water_plants()
{
        watering_available = 0; //ureztaketa etengabea ez izateko eta denbora tarte
                                 //bat pasatzeko, ureztatze automatikoa blokeatu.
        close window();         //Ureztatzerakoan, leihoa itxi egingo da.
        if (twice<2)
        {
                PORTA = 0b00000001;  //22 hankatxoa (ur-bonba) aktibatu
                timer_5_wait_water_plants(4); //Ureztaketak 4 segundo iraun ditzan,
                                              tenporizadorea martxan jarri
                twice++;
                water_plants();
        }
        else
        {
                twice = 0;
        }
}

void stop_watering_plants()
{
        PORTA = 0b00000000; //22 hankatxoa (ur-bonba) desgaitu
        timer_1_PWM();
}

void wait_between_watering()  //Bi ureztaketen artean 30 segundo itxaroteko.
{
        timer_3_wait_between_watering(2); //Itxarote tenporizadorea martxan jarri.

}
```

Others.c

```
#include "others.h"
#include "UART.h"
#include <avr/io.h>
#include <avr/iom2560.h>
#include <math.h>

//int motako balio bat 5 karaktereko karaktere katea bihurtu
void int_to_char(int value, unsigned char * result)
{
        result[0] = value/10000 + '0';
        value = value % 10000;
        result[1] = value/1000 + '0';
        value = value % 1000;
        result[2] = value/100 + '0';
        value = value % 100;
        result[3] = value/10 + '0';
        value = value % 10;
        result[4] = value + '0';
}


//karaktere katea int motako baliora bihurtzeko funtzioa
int uarray_to_int(unsigned char * index, int total_digits)
{
        int value = 0;
        int multiplier = 1;
        int digit = 0;
```

```
        for (int i=0; i<total_digits-1; i++)
        {
                multiplier *= 10;
        }
        for (int i=0; i<total_digits; i++)
        {
                digit = *index - '0';
                value += digit * multiplier;
                multiplier /= 10;
                index++;
        }
        return value;
}
```

PWM.c

```
#include "PWM.h"
#include "data.h"
#include "others.h"

const int MINDUTYNS[2] = {600, 550};
const int MAXDUTYNS[2] = {2500, 2500};

unsigned int DUTY[2];
unsigned int DUTY CYCLE[2];
unsigned int DUTY_USED[2];

//Leihoa erdian kokatzeko (hasieran soilik erabili da).
void serbo_middle()
{
        DUTY[0] = (MAXDUTYNS[0] - MINDUTYNS[0])/2 + MINDUTYNS[0];
        DUTY[1] = (MAXDUTYNS[1] - MINDUTYNS[1])/2 + MINDUTYNS[1];

        DUTY USED[0] = (MAXDUTYNS[0] - MINDUTYNS[0])/2 + MINDUTYNS[0];
        DUTY USED[1] = (MAXDUTYNS[1] - MINDUTYNS[1])/2 + MINDUTYNS[1];
        load_duty();
}

//Leihoa guztiz zabaltzeko funtzioa
void open_window()
{
        DUTY[0] = MINDUTYNS[0];
        DUTY[1] = MAXDUTYNS[1];
        load_duty();
        int_to_char(1, window);
}

//Leihoa guztiz ixteko funtzioa.
void close_window()
{
        DUTY[0] = MAXDUTYNS[0];
        DUTY[1] = MINDUTYNS[1];
        load_duty();

        int_to_char(0, window);
}

//us-tan tratatu diren posizioak zikloetara bihurtzeko.
void load_duty()
{
        for (int i=0; i<2; i++)
        DUTY CYCLE[i] = (DUTY USED[i]/1000.0) * (PWM FULL/20) - 1;
        //Duty -> us-tan /1000 = ms-tara pasa, * ms bat kontatzeko behar diren ziklo
        //  kopurua

}
```

Timers.c

```
#define F_CPU 16000000UL
#define output 1
#define input 0

#include <avr/io.h>
#include <avr/iom2560.h>
#include "PWM.h"
#include "greenhouse_elements.h"
#include "UART.h"
#include <avr/interrupt.h>


int pwm state = 0, left = 0;
int watering_count = 0, watering_available = 1;


//serboen kotrolerako tenporizadorea.
void timer_1_PWM()
{
	//16MHz-eko maiztasuna. T=1/F -> T = 1/16000000 = 0.0000000625s * 10^9 ns =
	//62'5 ns zikloko
	//Prescalerrik gabe 20ms kontatzeko -> 20000000/62'5 = 320.000 ziklo kontatu
	//beharko lirateke. 8 ko prescalerrarerkin -> 320.000/8 = 40.000 - 1
	TCCR1A = 0;			//TCCR1A erregistroa garbitu.
	TCCR1B = 0;			//TCCR1B erregistroa garbitu.
	TCCR1B |= (1<<CS11);	//8ko prescalerra hautatu.
	TCCR1B |= (1<<WGM12);	//Clear Timer on Compare Match (CTC) modua aktibatu.
	OCR1A =PWM_FULL-1;	//Serboak 8ko prescallerra erabilita, 20msko kontaketa
				//egiteko behar duen denbora kargatu.


	TCNT1 = 0;			//Tenporizadorearen balioa garbitu, 0tik hasteko.

	TIMSK1 = (1<<OCIE4A); //1 tenporizadorearen etenak gaitu
}



//Timer 3, ureztatze automatikoen artean denbora itxaron
void timer_3_wait_between_watering(int s)
{
	TCCR3A = 0;				//TCCR3A erregistroa garbitu.
	TCCR3B = 0;				//TCCR3B erregistroa garbitu.
	TCNT3 = 0;				//Tenporizadorearen balioa garbitu, 0tik hasteko.

	TCCR3B |= (1<<CS30) | (1<<CS32); //1024 prescalerra hautatu
	TCCR3B |= (1<<WGM32);		//Clear Timer on Compare Match (CTC) modua
					//aktibatu.
	OCR3A =15625*s;			//1s = 1.000.000.000ns -> 1.000.000.000/62.5 =
					//16.000.000-1 ziklo kontatu beharko lirateke 1
					//ms kontatzeko. clk/1024ko prescalerra
					//erabiltzen ari garenez -> 15999999/1024 = 15625
					//ziklo.


	TIMSK3 = (1<<OCIE3A);		//3 tenporizadorearen etenak gaitu
}


//Leihoaren mugimendua mantxotzeko erabilitako tenporizadorea
void timer_4_move_PWM()
{
	TCCR4A = 0;			//TCCR4A erregistroa garbitu.
	TCCR4B = 0;			//TCCR4A erregistroa garbitu.
	TCNT4 = 0;			//Tenporizadorearen balioa garbitu, 0tik hasteko.

	TCCR4B |= (1<<CS41);	//8 prescalerra hautatu
	TCCR4B |= (1<<WGM42);	//Clear Timer on Compare Match (CTC) modua aktibatu.
	OCR4A =4000;			//2ms-ko denbora jarri goi muga bezala

	TIMSK4 = (1<<OCIE4A); //4 tenporizadorearen etenak gaitu
}
```

```
//Ureztatze-sistema martxan mantendu beharreko denbora kontrolatzeko
void timer_5_wait_water_plants(int s)
{
        TCCR5A = 0;             //TCCR5A erregistroa garbitu.
        TCCR5B = 0;             //TCCR5B erregistroa garbitu.
        TCNT5 = 0;              //Tenporizadorearen balioa garbitu, 0tik hasteko.

        TCCR5B |= (1<<CS50) | (1<<CS52); //1024 prescalerra hautatu
        TCCR5B |= (1<<WGM52); //Clear Timer on Compare Match (CTC) modua aktibatu.
        OCR5A =15625*s;         //1s = 1.000.000.000ns -> 1.000.000.000/62.5 =
                                //16.000.000-1 ziklo kontatu beharko lirateke 1 ms
                                //kontatzeko. clk/1024ko prescalerra erabiltzen ari
                                //garenez -> 15999999/1024 = 15625 ziklo/s


        TIMSK5 = (1<<OCIE5A); //Enable Timer/Counter3 Interrupt
}

//Tenporizadore 1 etenaren zerbitzu-errutina
ISR(TIMER1_COMPA_vect)
{
        switch (pwm_state)
        {
                case 0: //Lehen serboaren kontrola gauzatu.
                PORTB |=  BV(7);
                OCR1A = DUTY_CYCLE[0];
                pwm_state=1;
                break;

                case 1: //Bigarren serboaren kontrola gauzatu.
                PORTB &= ~ BV(7);
                PORTB |= _BV(5);
                OCR1A = DUTY_CYCLE[1];
                pwm state=2;
                break;

                case 2: //Serboen periodoa errespetatzeko geratzen den denbora kargatu.
                PORTB &= ~_BV(5);
                left = DUTY_CYCLE[0] + DUTY_CYCLE[1] + 2;
                OCR1A = PWM FULL - left - 1;
                pwm state=0;
                load duty();
                break;
        }
}


//3 tenporizadorearen zerbitzu-errutina
ISR(TIMER3_COMPA_vect)
{
        watering_count++;
        if (watering_count==15) //Guztira 30 segundo pasa dira.
        {
                watering_available = 1;       //Ureztatzea posible dela adierazi.
                watering_count = 0;
        }
}


//4 tenporizadorearen zerbitzu-errutina
ISR(TIMER4_COMPA_vect)
{
        if (DUTY USED[0] < DUTY[0])
        DUTY USED[0]++;
        else if  (DUTY_USED[0] > DUTY[0])
        DUTY_USED[0]--;

        if (DUTY USED[1] < DUTY[1])
        DUTY USED[1]++;
        else if  (DUTY USED[1] > DUTY[1])
        DUTY_USED[1]--;
}
```

```
//Ureztatzea gauzatu da.
ISR(TIMER5_COMPA_vect)
{
	TIMSK5 &= ~_BV(OCIE5A);
	TCNT5 = 0;
	stop_watering_plants();

}
```

UART.c

```
#include <avr/io.h>
#include <avr/iom2560.h>

#include <avr/interrupt.h>
#include "UART.h"
#include "PWM.h"
#include "others.h"
#include "data.h"

int char_load=0, command_ready = 0, receive_data = 0, data_ready=0;
unsigned char received command[] = {' ', ' ', ' '};
unsigned char temporary_data[6] = {};

void setup_UART()
{
	UCSR0A = 0b00000010;  //0: Daturik jaso ez dela adierazi.
			      //0: Datu bidalketa ez da gauzatu.
			      //0: Datu erregistroa hutsik dago.
			      //0: Ez dago frame error-ik
			      //0: Ez dago data overrun-ik
			      //0: Ez dago parity bit errorerik
			      //1: TransmisiO abiadura bikoiztu
			      //0: MPCM baliogabetu.

	UCSR0B |= 0b11011000; //1: RX etenak gaitu (hargailuarenak)
			      //1: TX etenak gaitu (igorlearenak)
			      //0: Datu erregistroa hutsik dagoenean etenik ez jaurti
			      //1: Hartzailea aktibatu
			      //1: Igorlea aktibatu
			      //0: Karaktereen tamaina 8 bitekoa dela zehaztu
			      //0: Igorleak 8 bit bidaliko ditu
			      //0: Hartzaileak 8 bit bidaliko ditu

	UCSR0C = 0b00000110;  //00: Komunikazioa asinkronoa izango da.
			      //00: Ez da paritate bitik erabiliko
			      //0 : Stop bita bakarra izango da.
			      //11: Karaktere bakoitza 8 bit-ez errepresentatzen da.

	UBRR0 = 207;          //Konfigurazio honekin, 9600 baud rate-a izateko
			      beharrezkoa den UBRR balioa.
	UDR0 = ' ';
}


void send_UART(unsigned char * message, int letter_count)
{
	int i = 0;
	while (i<letter_count)
	{
		if (UCSR0A & (1<<UDRE0)) //Igortzeko bufferrean datuak kargatu
					  daitezke, bufferra hutsik dago.
		{
			UDR0 = *message; //Bidaltzeko datua bufferrean kargatu.
			message++;       //Karaktere kateko hurrengo karakterea hautatu.
			i++;
		}
	}
}
```

```
ISR(USART0_TX_vect)
{
        DDRC |=  BV(7);
        PORTC |= _BV(7);
}

ISR(USART0_RX_vect)
{
        //Command change
        char temp_char = UDR0;//Jasotako karakterea irakurri.
        if (!receive_data)
        {
                //Karaktere horietako bat jasoz gero, protokoloko aginduaren hasiera
                //izan liteke, beraz, agindua jasotzeko prestatu.
                if (temp_char == 'W' ||
                    temp_char == 'T' ||
                    temp_char == 'H' ||
                    temp_char == 'D' ||
                    temp_char == 'F')
                {
                        char_load = 0;
                        for (int i=0; i<3; i++)//Aurretik jasotako karaktereak ezabatu.
                        received_command[i] = ' ';
                }
                received_command[char_load] = temp_char;

                char_load++;
                if (char_load == 3) //3 karaktere jaso badira, agindu bat izan liteke.
                {
                        //3 karaktere jaso direla adierazi, main programak karaktere
                        //horiek prozesa ditzan.
                        command_ready = 1;
                        char_load = 0;
                }
        }

        //Jaso den aginduak datuak txertatuta baditu ondoren, datuen karga egin
        //beharko da.
        else
        {
                temporary_data[char_load] = temp_char;

                char_load++;
                if (char_load == 6)    //Datu guztiak jaso dira.
                {

                        char load = 0;
                        receive data = 0;
                        //'TEM' agindua jaso bada, tenperatura kargatu.
                        if (received_command[0]=='T' &&
                            received_command[1]=='E' &&
                            received command[2]=='M')
                                max temperature = uarray to int(&temporary data[1], 5);
                        //'HUM' agindua jaso bada, hezetasun minimoa kargatu.
                        else if (received_command[0]=='H' &&
                                 received_command[1]=='U' &&
                                 received command[2]=='M')
                                min hum = uarray to int(&temporary data[1], 5);
                        //'WPP' agindua jaso bada, hautatutako landarea kargatu.
                        else if (received_command[0]=='W' &&
                                 received command[1]=='P' &&
                                 received_command[2]=='P')
                        {
                               int i plant selected=uarray to int(&temporary data[1],5);
                               int_to_char(i_plant_selected, plant_selected);
                        }
                }
        }

}
```

Data.c

```c
#include "data.h"
#include "others.h"

unsigned char temperature[5] = {};
unsigned char soil_hum[5] = {};
unsigned char window[5] = {};
unsigned char door[5] = {};
unsigned char fan[5] = {};
unsigned char plant selected[5] = {};
int max_temperature, min_hum;

void initialize_values()
{
        int to char(2, window);
        int to char(2, door);
        int_to_char(0, fan);
        int_to_char(0, plant_selected);
        max temperature = 30;
        min_hum = 400;

}
```

Analog_reads.h

```c
#ifndef SENSORS H
#define SENSORS_H_

void measure_soil_and_temp();

extern int temperature_reads[5], soil_hum_reads[10], soil_hum_ready,
temperature_ready;

#endif /* SENSORS_H_ */
```

Data.h

```c
#ifndef DATA_H_
#define DATA_H_

extern unsigned char temperature[5];
extern unsigned char soil hum[5];
extern unsigned char window[5];
extern unsigned char door[5];
extern unsigned char fan[5];
extern unsigned char plant selected[5];
extern int max_temperature, min_hum;

void initialize_values();

#endif /* DATA_H_ */
```

Greenhouse_elements.h

```c
#ifndef GREENHOUSE ELEMENTS H
#define GREENHOUSE_ELEMENTS_H_

void setup_serbo();
void setup_pins();
void open_door();
void close door();
void stop door();
void activate_fan();
void stop_fan();
void water_plants();
```

```
void wait between watering();
void stop_watering_plants();

extern int door_direction;

#endif /* GREENHOUSE_ELEMENTS_H_ */
```

Others.h

```
#ifndef OTHERS_H_
#define OTHERS_H_

void int to char(int value, unsigned char * result);
int uarray to int(unsigned char * index, int total digits);
int calculate_median(int values[], int value_count);
float calculate_temperature(int analog_value);

#endif /* OTHERS_H_ */
```

PWM.h

```
#ifndef PWM H
#define PWM_H_

#define PWM_FULL 40000 //20ms kontatzeko ziklo kopurua => 20000000ns/62.5ns/8 = 40000
#define PWM_servo PWM_FULL/2

extern unsigned int DUTY_CYCLE[2], DUTY[2], DUTY_USED[2];

void open_window();
void close_window();
void serbo middle();
void load_duty();

#endif /* PWM_H_ */
```

Timers.h

```
#ifndef TIMERS H
#define TIMERS_H_

extern int pwm_state;
extern int watering_available;

void global interrupts enable();
void global_interrupts_disable();

void timer 1 PWM();.
void timer 3 wait between_watering(int s);
void timer 4 move PWM();
void timer_5_wait_water_plants(int s);

#endif /* TIMERS_H_ */
```

UART.h

```
#ifndef UART_H_
#define UART_H_

void setup UART();
void send UART(unsigned char *, int);
void get_UART();
```

```
extern unsigned char received command[3];
extern int command_ready, data_ready, receive_data;


#endif /* UART_H_ */
```

B Eranskina: interfazearen kodea

Connection.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
from Main_Menu import Ui_MainMenu
import mysql.connector, mmap, os, select, socket, time, datetime


class Ui_ConnectWindow(object):
    def setupUi(self, ConnectWindow):
        ConnectWindow.setObjectName("ConnectWindow")
        ConnectWindow.resize(982, 445)
        ConnectWindow.setStyleSheet("QWidget\n"
"{\n"
"    font-size: 15px;\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    font-weight: 400;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(ConnectWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.connect_btn = QtWidgets.QPushButton(self.centralwidget)
        self.connect_btn.setGeometry(QtCore.QRect(420, 220, 101, 51))
        self.connect_btn.setObjectName("connect_btn")
        self.ip_connect_input = QtWidgets.QLineEdit(self.centralwidget)
        self.ip_connect_input.setGeometry(QtCore.QRect(340, 130, 111, 31))
        self.ip_connect_input.setText("")
        self.ip_connect_input.setObjectName("ip_connect_input")
        self.port_connect_input = QtWidgets.QLineEdit(self.centralwidget)
        self.port_connect_input.setGeometry(QtCore.QRect(580, 130, 111, 31))
        self.port_connect_input.setObjectName("port_connect_input")
        self.ip_connect_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.ip_connect_lbl.setGeometry(QtCore.QRect(210, 130, 111, 31))
        self.ip_connect_lbl.setObjectName("ip_connect_lbl")
        self.ip_connect_lbl.setReadOnly(True)
        self.port_connect_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.port_connect_lbl.setGeometry(QtCore.QRect(490, 130, 81, 31))
```

```
        self.port_connect_lbl.setObjectName("port_connect_lbl")
        self.port_connect_lbl.setReadOnly(True)
        self.error_text = QtWidgets.QTextEdit(self.centralwidget)
        self.error_text.setGeometry(QtCore.QRect(200, 300, 571, 31))
        self.error_text.setObjectName("error_text")
        ConnectWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(ConnectWindow)
        self.statusbar.setObjectName("statusbar")
        ConnectWindow.setStatusBar(self.statusbar)

        self.retranslateUi(ConnectWindow)
        QtCore.QMetaObject.connectSlotsByName(ConnectWindow)

       #Konektatu botoia sakatzean, connect() funtzioa exekutatu:
        self.connect_btn.clicked.connect(lambda: self.connect())


    def connect(self):
        self.error_text.setText("")
        server_ip = ""
        server_port = ""

       #Interfazeko datuak irakurri
        server_ip = self.ip_connect_input.text()
        server_port = self.port_connect_input.text()

       #Zerbitzariarekin komunikatzeko UDP socketa sortu eta hasieratu.
        if server_ip and server_port:
                server_address = (server_ip,int(server_port))
                s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
                try:
                        s.connect(server_address)
                except socket.error:
                        self.error_text.setText("Ezin izan da konexioa gauzatu,
balioak errepasatu itzazu.")

               #MySQL zerbitzariarekin konexioa hasi
                db = mysql.connector.connect(
                        host="localhost",
                        user="root",
                        passwd="root",
                        database="greenhouse"
                )

                #Jasotako komando eta datuak gordetzeko erabiliko den fitxategi
                birtuala hasieratu eta bertan datuak kargatu.
                info_file = mmap.mmap(-1, 66)

       info_file.write("WISOK+00000HUSOK+00000TESOK+00000DOSOK+00000FASOK+00000WPSOK+
00000".encode())

                #Prozesuen arteko komunikaziorako hobiak hasieratu.
                rc1, wr1 = os.pipe()
                rc2, wr2 = os.pipe()
                rc3, wr3 = os.pipe()

                #Datu bidalketa eta harketa prozesuak hasieratu eta menu nagusia
                #zabaldu.
                self.initializeGreenhouseInfo(info_file, s, db, rc1, rc2, rc3, wr1,
wr2, wr3)
                self.openMainMenu(s, db, info_file, wr1, wr2, wr3)



    #Menu nagusia zabaldu
    def openMainMenu(self, socket, db, info_file, wr1, wr2, wr3):
        self.MainMenuWindow = QtWidgets.QMainWindow()
        self.ui = Ui_MainMenu()
        self.ui.setupUi(self.MainMenuWindow, socket, db, info_file, wr1, wr2, wr3)
        self.MainMenuWindow.show()
        ConnectWindow.hide()
```

```
    #Datu bidalketa eta harketa prozesuak hasieratu
    def initializeGreenhouseInfo(self, info_file, socket, db, rc1, rc2, rc3, wr1, wr2,
wr3):
        pid = os.fork()
        if pid==0:                      #1. Prozesu umea (1. maila):
            pid2 = os.fork()
            if pid2==0:                 #2. Prozesu umea (2. maila) berotegiaren
                                        #egoeraren berri izateko komandoak etengabe
                                        #bidaliko ditu.
                os.close(wr1)
                self.temporaryPetition(socket, rc1)
            else:                       #1. Prozesu umeak mikrokontrolagailutik datozkion
                                        #erantzunak jasotzeaz arduratuko da.
                os.close(wr2)
                self.receiveValues(info_file, socket, rc2)
        else:
            pid3 = os.fork()
            if pid3 == 0:
                os.close(wr3)
                self.insertDataToDatabase(db, info_file, rc3)



    #Berotegieran egoeraren berri izateko datuen eskaera etengabe egin
    def temporaryPetition(self, s, rc1):
        print("Egoera esakera periodikoa egingo duen prozesu umea martxan da.")
        i=0
        commands = ["WIS", "HUS", "TES", "DOS", "FAS", "WPS"]
        while True:
            while True:
                for i in range(6):
                   #rc1 irakurketa hodian "exit" irakurtzen bada, bidalketa prozesua
                   #eten.
                    if len(select.select([rc1],[],[],1)[0])==1:
                        message = os.read(rc1, 4)
                        if message.decode()=="exit":
                            print("Bidalketa prozesua eten da")
                            exit(0)
                   #Bestela, tokatzen den komandoa bidali
                    else:
                        s.send(commands[i].encode())
                        print("Bidali da komandoa: " + commands[i])
                        time.sleep(1)



    #Mikrokontrolagailutik berotegiaren inguruko informazioa duten erantzunak jaso
    def receiveValues(self, info_file, s, rc2):
        print("Komandoak jasoko dituen prozesu umea martxan da.")
        while True:
            while True:
               #rc2 irakurketa hodian "exit" irakurriz gero, harketa prozesua eten.
                if len(select.select([rc2],[],[],1)[0])==1:
                    message = os.read(rc2, 4)
                    if message.decode()=="exit":
                        print("Jasotze prozesua eten da")
                        exit(0)
               #Bestela, socket-etik datua irakurri, komandoa sailkatu eta info_file
               #fitxategian datua dagokion tokian gorde.
                else:
                    data_received, _, _ = select.select( [s], [], [], 1)
                    if data_received:
                        data, server_address = s.recvfrom(1024)
                        if not data:
                            break
                        data_string = data.decode()
                        command = data_string[:6]
                        value = data_string[6:]
                        if command=="WISOK+":
                            info_file.seek(6)
                            info_file.write(value.encode())
```

```
                        elif command=="HUSOK+":
                            info_file.seek(17)
                            info_file.write(value.encode())
                        elif command=="TESOK+":
                            info_file.seek(28)
                            info_file.write(value.encode())
                        elif command=="DOSOK+":
                            info_file.seek(39)
                            info_file.write(value.encode())
                        elif command=="FASOK+":
                            info_file.seek(50)
                            info_file.write(value.encode())
                        elif command=="WPSOK+":
                            info_file.seek(61)
                            info_file.write(value.encode())
                        print("Jaso da komandoa: " + command + value)



    #Info_file fitxategiko datuak datu basean kargatu 5 segund
     def insertDataToDatabase(self, db, info_file, rc3):
         while True:
             if len(select.select([rc3],[],[],1)[0])==1:
                 message = os.read(rc3, 4)
                 if message.decode()=="exit":
                     print("Datu base eguneraketa prozesua eten da")
                     exit(0)
             #Uneko hezetasuna irakurri.
             info_file.seek(17)
             current_hum = str(info_file.read(5).decode())
             #Uneko tenperatura irakurri.
             info_file.seek(28)
             current_temp = str(info_file.read(5).decode())
             #Hautatuta dagoen landarearen identifikadorea irakurri eta honen izena
              #lortu.
             info_file.seek(61)
             current_plant_id = str(info_file.read(5).decode())
             current_date = datetime.datetime.now().strftime('%Y-%m-%d %H:%M:%S')
             cursor = db.cursor(buffered=True)
             cursor.execute("SELECT name FROM plants WHERE id=%s" % (current_plant_id))
             current_plant_name = str(cursor.fetchall()[0])[2:-3]
             #Informazioa datu basean txertatu.
             cursor.execute("INSERT INTO data VALUES ('%s', '%i', '%i', '%s')" %
(current_plant_name, int(current_hum), int(current_temp), current_date))
             db.commit()
             cursor.close()
             time.sleep(60)



     def retranslateUi(self, ConnectWindow):
         _translate = QtCore.QCoreApplication.translate
         ConnectWindow.setWindowTitle(_translate("ConnectWindow", "Konexio leihoa"))
         self.connect_btn.setText(_translate("ConnectWindow", "Konektatu"))
         self.ip_connect_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">IP
helbidea:</span></p></body></html>"))
         self.port_connect_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-
size:12pt;\">Portua:</span></p></body></html>"))
```

```
        self.error_text.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))


if __name__ == "__main__":
    import sys, socket
    app = QtWidgets.QApplication(sys.argv)
    ConnectWindow = QtWidgets.QMainWindow()
    ui = Ui_ConnectWindow()
    ui.setupUi(ConnectWindow)
    ConnectWindow.show()

       app.exec_()
```

Greenhouse_info.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import socket, select, os, signal, time, mmap, sys, images
import Main_Menu

class Ui_GreenhouseWindow(object):
    def setupUi(self, GreenhouseWindow, socket, db, info_file, wr1, wr2, wr3):
        GreenhouseWindow.setObjectName("GreenhouseWindow")
        GreenhouseWindow.resize(982, 445)
        GreenhouseWindow.setStyleSheet("QWidget\n"
"{\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    font-weight: 400;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(GreenhouseWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.soil_hum_data = QtWidgets.QTextEdit(self.centralwidget)
        self.soil_hum_data.setGeometry(QtCore.QRect(520, 10, 121, 31))
        self.soil_hum_data.setObjectName("soil_hum_data")
        self.soil_hum_data.setReadOnly(True)
        self.tenp_data = QtWidgets.QTextEdit(self.centralwidget)
        self.tenp_data.setGeometry(QtCore.QRect(140, 10, 121, 31))
```

```
        self.tenp_data.setObjectName("tenp_data")
        self.tenp_data.setReadOnly(True)
        self.door_data = QtWidgets.QTextEdit(self.centralwidget)
        self.door_data.setGeometry(QtCore.QRect(140, 100, 121, 31))
        self.door_data.setObjectName("door_data")
        self.door_data.setReadOnly(True)
        self.window_data = QtWidgets.QTextEdit(self.centralwidget)
        self.window_data.setGeometry(QtCore.QRect(140, 180, 121, 31))
        self.window_data.setObjectName("window_data")
        self.window_data.setReadOnly(True)
        self.window_close_btn = QtWidgets.QPushButton(self.centralwidget)
        self.window_close_btn.setGeometry(QtCore.QRect(300, 170, 91, 51))
        self.window_close_btn.setObjectName("window_close_btn")
        self.window_open_btn = QtWidgets.QPushButton(self.centralwidget)
        self.window_open_btn.setGeometry(QtCore.QRect(410, 170, 91, 51))
        self.window_open_btn.setObjectName("window_open_btn")
        self.fan_stop_btn = QtWidgets.QPushButton(self.centralwidget)
        self.fan_stop_btn.setGeometry(QtCore.QRect(300, 250, 91, 51))
        self.fan_stop_btn.setObjectName("fan_stop_btn")
        self.fan_start_btn = QtWidgets.QPushButton(self.centralwidget)
        self.fan_start_btn.setGeometry(QtCore.QRect(410, 250, 91, 51))
        self.fan_start_btn.setObjectName("fan_start_btn")
        self.auto_btn = QtWidgets.QPushButton(self.centralwidget)
        self.auto_btn.setGeometry(QtCore.QRect(170, 330, 181, 51))
        self.auto_btn.setObjectName("auto_btn")
        self.door_open_btn = QtWidgets.QPushButton(self.centralwidget)
        self.door_open_btn.setGeometry(QtCore.QRect(410, 90, 91, 51))
        self.door_open_btn.setObjectName("door_open_btn")
        self.fan_data = QtWidgets.QTextEdit(self.centralwidget)
        self.fan_data.setGeometry(QtCore.QRect(140, 260, 121, 31))
        self.fan_data.setObjectName("fan_data")
        self.fan_data.setReadOnly(True)
        self.door_close_btn = QtWidgets.QPushButton(self.centralwidget)
        self.door_close_btn.setGeometry(QtCore.QRect(300, 90, 91, 51))
        self.door_close_btn.setObjectName("door_close_btn")
        self.hum_ico = QtWidgets.QToolButton(self.centralwidget)
        self.hum_ico.setGeometry(QtCore.QRect(480, 10, 31, 31))
        icon = QtGui.QIcon()
        icon.addPixmap(QtGui.QPixmap(":/img/img/hezetasuna.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.hum_ico.setIcon(icon)
        self.hum_ico.setIconSize(QtCore.QSize(28, 28))
        self.hum_ico.setObjectName("hum_ico")
        self.tenp_ico = QtWidgets.QToolButton(self.centralwidget)
        self.tenp_ico.setGeometry(QtCore.QRect(100, 10, 31, 31))
        icon1 = QtGui.QIcon()
        icon1.addPixmap(QtGui.QPixmap(":/img/img/tenperatura.png"),
QtGui.QIcon.Normal, QtGui.QIcon.Off)
        self.tenp_ico.setIcon(icon1)
        self.tenp_ico.setIconSize(QtCore.QSize(28, 28))
        self.tenp_ico.setObjectName("tenp_ico")
        self.door_ico = QtWidgets.QToolButton(self.centralwidget)
        self.door_ico.setGeometry(QtCore.QRect(100, 100, 31, 31))
        icon2 = QtGui.QIcon()
        icon2.addPixmap(QtGui.QPixmap(":/img/img/door.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.door_ico.setIcon(icon2)
        self.door_ico.setIconSize(QtCore.QSize(28, 28))
        self.door_ico.setObjectName("door_ico")
        self.water_plants_btn = QtWidgets.QPushButton(self.centralwidget)
        self.water_plants_btn.setGeometry(QtCore.QRect(670, 0, 141, 51))
        self.water_plants_btn.setObjectName("water_plants_btn")
        self.window_ico = QtWidgets.QToolButton(self.centralwidget)
        self.window_ico.setGeometry(QtCore.QRect(100, 180, 31, 31))
        icon3 = QtGui.QIcon()
        icon3.addPixmap(QtGui.QPixmap(":/img/img/window.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.window_ico.setIcon(icon3)
        self.window_ico.setIconSize(QtCore.QSize(28, 28))
        self.window_ico.setObjectName("window_ico")
        self.window_ico_2 = QtWidgets.QToolButton(self.centralwidget)
        self.window_ico_2.setGeometry(QtCore.QRect(100, 260, 31, 31))
        icon4 = QtGui.QIcon()
```

```
        icon4.addPixmap(QtGui.QPixmap(":/img/img/fan.png"), QtGui.QIcon.Normal,
QtGui.QIcon.Off)
        self.window_ico_2.setIcon(icon4)
        self.window_ico_2.setIconSize(QtCore.QSize(28, 28))
        self.window_ico_2.setObjectName("window_ico_2")
        self.menu_back_btn = QtWidgets.QPushButton(self.centralwidget)
        self.menu_back_btn.setGeometry(QtCore.QRect(840, 320, 101, 51))
        self.menu_back_btn.setObjectName("menu_back_btn")
        GreenhouseWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(GreenhouseWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 982, 26))
        self.menubar.setObjectName("menubar")
        GreenhouseWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(GreenhouseWindow)
        self.statusbar.setObjectName("statusbar")
        GreenhouseWindow.setStatusBar(self.statusbar)

        #"Menura itzuli" botoia sakatzean, goToMenu prozesua exekutatu
        self.menu_back_btn.clicked.connect(lambda: self.goToMenu(GreenhouseWindow,
socket, db, info_file, wr1, wr2, wr3))

        #Denboragailua martxan jarri
        self.startTimer(info_file)

        self.retranslateUi(GreenhouseWindow, info_file, socket, db)
        QtCore.QMetaObject.connectSlotsByName(GreenhouseWindow)




    def retranslateUi(self, GreenhouseWindow, info_file, socket, db):
        _translate = QtCore.QCoreApplication.translate
        GreenhouseWindow.setWindowTitle(_translate("GreenhouseWindow", "Berotegiaren
egoera"))
        self.soil_hum_data.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.tenp_data.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.door_data.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.window_data.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.window_close_btn.setText(_translate("GreenhouseWindow", "Itxi"))
        self.window_open_btn.setText(_translate("GreenhouseWindow", "Ireki"))
        self.fan_stop_btn.setText(_translate("GreenhouseWindow", "Gelditu"))
```

```
        self.fan_start_btn.setText(_translate("GreenhouseWindow", "Abiatu"))
        self.auto_btn.setText(_translate("GreenhouseWindow", "Modu automatikora
itzuli"))
        self.door_open_btn.setText(_translate("GreenhouseWindow", "Ireki"))
        self.door_close_btn.setText(_translate("GreenhouseWindow", "Itxi"))
        self.hum_ico.setText(_translate("GreenhouseWindow", "..."))
        self.tenp_ico.setText(_translate("GreenhouseWindow", "..."))
        self.door_ico.setText(_translate("GreenhouseWindow", "..."))
        self.water_plants_btn.setText(_translate("GreenhouseWindow", "Ureztatu"))
        self.window_ico.setText(_translate("GreenhouseWindow", "..."))
        self.window_ico_2.setText(_translate("GreenhouseWindow", "..."))
        self.menu_back_btn.setText(_translate("GreenhouseWindow", "Menura itzuli"))


       #Interfazeko botoi ezberdinak sakatzen direnean, dagokion komandoa bidali
       #socketaren bidez
        self.door_close_btn.clicked.connect(lambda: self.sendCommand("DOC", socket,
info_file))
        self.door_open_btn.clicked.connect(lambda: self.sendCommand("DOO", socket,
info_file))
        self.window_close_btn.clicked.connect(lambda: self.sendCommand("WIC", socket,
info_file))
        self.window_open_btn.clicked.connect(lambda: self.sendCommand("WIO", socket,
info_file))
        self.fan_stop_btn.clicked.connect(lambda: self.sendCommand("FAC", socket,
info_file))
        self.fan_start_btn.clicked.connect(lambda: self.sendCommand("FAO", socket,
info_file))
        self.auto_btn.clicked.connect(lambda: self.sendCommand("DAC", socket,
info_file))
        self.water_plants_btn.clicked.connect(lambda: self.sendCommand("WAO", socket,
info_file))



    #Aginduak socketaren bidez bidali ESP32ra eta interfazeko datuak eguneratzeko
    #prozesua exekutatu.
    def sendCommand(self, command, s, info_file):
        s.send(command.encode())
        print("Komandoa bidali da: " + command)
        self.updateGUI(info_file)



    #Denboragailuak 250ms-ro interfazea eguneratzeko agindua emango du.
    def startTimer(self, info_file):
        self.timer = QtCore.QTimer()
        self.timer.start(250)
        self.timer.timeout.connect(lambda: self.updateGUI(info_file))



    #info_file fitxategian dauden datuak, dagokien lekuetan pantailaratu interfazearen
    #bidez.
    def updateGUI(self, info_file):
        #Datuak fitxategitik irakurri
        window = info_file[6:11].decode()
        soil_humidity = info_file[17:22].decode()
        temperature = info_file[28:33].decode()
        door = info_file[39:44].decode()
        fan = info_file[50:55].decode()


        #Datuak pantailan kargatu
        #Leihoa
        if window == "00000":
                self.window_data.setText("Itxita")
        elif window == "00001":
                self.window_data.setText("Irekita")
        elif window == "00002":
                self.window_data.setText("Mugimenduan")
```

```
        #Lurraren hezetasuna.
        soil_humidity_percent = round(100 - (int(soil_humidity)-275)*100/(1023-275),2)
        if soil_humidity_percent > 100:
                soil_humidity_percent = 100.00
        elif soil_humidity_percent < 0:
                soil_humidity_percent = 0.00
        self.soil_hum_data.setText( "%" + str(soil_humidity_percent) )

        #Inguruko tenperatura.
        temperature_show = temperature.lstrip("0")
        if len(temperature_show) == 0:
            temperature_show = "0"
        self.tenp_data.setText(temperature_show + "Â°C")

        #Atea.
        if door == "00000":
                self.door_data.setText("Itxita")
        elif door == "00001":
                self.door_data.setText("Irekita")
        elif window == "00002":
                self.window_data.setText("Mugimenduan")

        #Bentilagailua.
        if fan == "00000":
                self.fan_data.setText("Geldi")
        elif fan == "00001":
                self.fan_data.setText("Martxan")



    #"Menura itzuli" botoia sakatzean, menua ireki
    def goToMenu(self, GreenhouseWindow, socket, db, info_file, wr1, wr2, wr3):
        self.MainMenuWindow = QtWidgets.QMainWindow()
        self.ui=Main_Menu.Ui_MainMenu()
        self.ui.setupUi(self.MainMenuWindow, socket, db, info_file, wr1, wr2, wr3)
        self.MainMenuWindow.show()

           GreenhouseWindow.hide()
```

Historic_data.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import Main_Menu, datetime


class Ui_DatabaseWindow(object):
    def setupUi(self, DatabaseWindow, socket, db, info_file, wr1, wr2, wr3):
        DatabaseWindow.setObjectName("DatabaseWindow")
        DatabaseWindow.resize(982, 445)
        DatabaseWindow.setStyleSheet("QWidget\n"
"{\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    font-weight: 400;\n"
"    text-align: center;\n"
"}\n"
"\n"
```

```
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(DatabaseWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.database_info = QtWidgets.QTableWidget(self.centralwidget)
        self.database_info.setGeometry(QtCore.QRect(110, 80, 800, 241))
        self.database_info.setRowCount(1)
        self.database_info.setColumnCount(4)
        self.database_info.setObjectName("database_info")


self.database_info.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.Stret
ch)
        self.database_info.setHorizontalHeaderLabels(['Landarea', 'Hezetasuna',
'Tenperatura', 'Data'])
        self.menu_back_btn = QtWidgets.QPushButton(self.centralwidget)
        self.menu_back_btn.setGeometry(QtCore.QRect(460, 340, 101, 51))
        self.menu_back_btn.setObjectName("menu_back_btn")
        DatabaseWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(DatabaseWindow)
        self.statusbar.setObjectName("statusbar")
        DatabaseWindow.setStatusBar(self.statusbar)

        #Leihoa zabaltzean, datu baseko datuak pantailaratu
        self.showData(db)

        #"Menura itzuli" sakatzean menu nagusira itzuli
        self.menu_back_btn.clicked.connect(lambda: self.goToMenu(DatabaseWindow,
socket, db, info_file, wr1, wr2, wr3))

       #Datu baseko datuak birkargatzeko denboragailua martxan jarri
        self.startTimer(db, info_file)

        self.retranslateUi(DatabaseWindow)
        QtCore.QMetaObject.connectSlotsByName(DatabaseWindow)



    #30 segundoz behin datu baseko datuak pantailaratu
    def startTimer(self, db, info_file):
        self.timer = QtCore.QTimer()
        self.timer.start(30000)
        self.timer.timeout.connect(lambda: self.showData(db))


    #Datuak modu antolatuan pantailaratu
    def showData(self, db):
        self.database_info.setRowCount(0)
        cursor = db.cursor()
        cursor.execute("SELECT * FROM data ORDER BY date DESC")
        database_data = cursor.fetchall()
        row_number = 0
       #Lortu den datu bakoitzeko, gelaxkaz gelaxka kargatu.
        for data in database_data:
            self.database_info.insertRow(row_number)
            self.database_info.setItem(row_number, 0,
QtWidgets.QTableWidgetItem(str(data[0])))
```

```
            soil_humidity_percent = round(100 - (int(data[1])-275)*100/(1023-275),2)
            if soil_humidity_percent > 100:
                soil_humidity_percent = 100.00
            elif soil_humidity_percent < 0:
                soil_humidity_percent = 0.00
            self.database_info.setItem(row_number, 1,
QtWidgets.QTableWidgetItem("%"+str(soil_humidity_percent)))
            self.database_info.setItem(row_number, 2,
QtWidgets.QTableWidgetItem(str(data[2])+'Â°C'))
            self.database_info.setItem(row_number, 3,
QtWidgets.QTableWidgetItem(data[3].strftime('%Y-%m-%d %H:%M:%S')))
            row_number+=1
        cursor.close()



    #Datu historikoak dituen leihoa itxi eta menu nagusia zabaldu.
    def goToMenu(self, DatabaseWindow, socket, db, info_file, wr1, wr2, wr3):
        self.MainMenuWindow = QtWidgets.QMainWindow()
        self.ui=Main_Menu.Ui_MainMenu()
        self.ui.setupUi(self.MainMenuWindow, socket, db, info_file, wr1, wr2, wr3)
        self.MainMenuWindow.show()
        DatabaseWindow.hide()



    def retranslateUi(self, DatabaseWindow):
        _translate = QtCore.QCoreApplication.translate
        DatabaseWindow.setWindowTitle(_translate("DatabaseWindow", "Historikoa"))
        self.menu_back_btn.setText(_translate("DatabaseWindow", "Menura itzuli"))
```

Login.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
from Select_plant import Ui_SelectPlantWindow
from User_window import Ui_UserWindow
import mysql.connector

class Ui_LoginWindow(object):
    def setupUi(self, LoginWindow, MainMenuWindow, socket, db, info_file, wr1, wr2,
wr3, window_to_open):
        LoginWindow.setObjectName("LoginWindow")
        LoginWindow.resize(668, 346)
        LoginWindow.setStyleSheet("QWidget\n"
"{\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
```

```
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(LoginWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.pass_login_input = QtWidgets.QLineEdit(self.centralwidget)
        self.pass_login_input.setGeometry(QtCore.QRect(360, 130, 131, 31))
        self.pass_login_input.setObjectName("pass_login_input")
        self.pass_login_input.setEchoMode(QtWidgets.QLineEdit.Password)
        self.error_text = QtWidgets.QTextEdit(self.centralwidget)
        self.error_text.setGeometry(QtCore.QRect(40, 270, 571, 31))
        self.error_text.setObjectName("error_text")
        self.password_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.password_lbl.setGeometry(QtCore.QRect(180, 130, 161, 31))
        self.password_lbl.setObjectName("password_lbl")
        self.username_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.username_lbl.setGeometry(QtCore.QRect(180, 80, 171, 31))
        self.username_lbl.setObjectName("username_lbl")
        self.login_user_btn = QtWidgets.QPushButton(self.centralwidget)
        self.login_user_btn.setGeometry(QtCore.QRect(260, 190, 101, 51))
        self.login_user_btn.setObjectName("login_user_btn")
        self.user_login_input = QtWidgets.QLineEdit(self.centralwidget)
        self.user_login_input.setGeometry(QtCore.QRect(360, 80, 131, 31))
        self.user_login_input.setText("")
        self.user_login_input.setObjectName("user_login_input")
        LoginWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(LoginWindow)
        self.statusbar.setObjectName("statusbar")
        LoginWindow.setStatusBar(self.statusbar)


        self.login_user_btn.clicked.connect(lambda: self.loginUser(LoginWindow,
MainMenuWindow, socket, db, info_file, wr1, wr2, wr3, window_to_open))

        self.retranslateUi(LoginWindow)
        QtCore.QMetaObject.connectSlotsByName(LoginWindow)


    def loginUser(self, LoginWindow, MainMenuWindow, socket, db, info_file, wr1, wr2,
wr3, window_to_open):
        username = self.user_login_input.text()
        password = self.user_login_input.text()
        if username and password:
            cursor = db.cursor(buffered=True)
            cursor.execute("SELECT * FROM users WHERE username='%s' and password='%s'"
% (username, password))
            if cursor.rowcount == 0:
                self.error_text.setText("Erabiltzailea eta pasahitza ez dira egokiak")
                cursor.close()
            else:
                cursor.close()
                LoginWindow.hide()
                if window_to_open == "plantWindow":
                    self.SelectPlantWindow = QtWidgets.QMainWindow()
                    self.ui=Ui_SelectPlantWindow()
                    self.ui.setupUi(self.SelectPlantWindow, socket, db, info_file,
wr1, wr2, wr3)
                    self.SelectPlantWindow.show()
                    MainMenuWindow.hide()
                elif window_to_open == "userWindow":
                    self.UserWindow = QtWidgets.QMainWindow()
                    self.ui=Ui_UserWindow()
```

```
                    self.ui.setupUi(self.UserWindow, socket, db, info_file, wr1, wr2,
wr3)
                    self.UserWindow.show()
                    MainMenuWindow.hide()
        else:
            self.error_text.setText("Erabiltzailea eta pasahitza sartu")



    def retranslateUi(self, LoginWindow):
        _translate = QtCore.QCoreApplication.translate
        LoginWindow.setWindowTitle(_translate("LoginWindow", "Kredentzialak"))
        self.error_text.setHtml(_translate("LoginWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.password_lbl.setHtml(_translate("LoginWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-
size:12pt;\">Pasahitza:</span></p></body></html>"))
        self.username_lbl.setHtml(_translate("LoginWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-
size:12pt;\">Erabiltzailea:</span></p></body></html>"))
        self.login_user_btn.setText(_translate("LoginWindow", "Sartu"))


if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    LoginWindow = QtWidgets.QMainWindow()
    ui = Ui_LoginWindow()
    ui.setupUi(LoginWindow)
    LoginWindow.show()

       sys.exit(app.exec_())
```

Main_Menu.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
from Greenhouse_info import Ui_GreenhouseWindow
from Login import Ui_LoginWindow
import select, os, time, sys
import Historic_data

class Ui_MainMenu(object):
    def setupUi(self, MainMenuWindow, socket, db, info_file, wr1, wr2, wr3):
        MainMenuWindow.setObjectName("MainMenuWindow")
        MainMenuWindow.resize(982, 502)
        MainMenuWindow.setStyleSheet("QWidget\n"
"{\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
```

```
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    font-weight: 400;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(MainMenuWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.database_btn = QtWidgets.QPushButton(self.centralwidget)
        self.database_btn.setGeometry(QtCore.QRect(370, 110, 191, 61))
        self.database_btn.setObjectName("database_btn")
        self.add_user_btn = QtWidgets.QPushButton(self.centralwidget)
        self.add_user_btn.setGeometry(QtCore.QRect(370, 290, 191, 61))
        self.add_user_btn.setObjectName("add_user_btn")
        self.greenhouse_state_btn = QtWidgets.QPushButton(self.centralwidget)
        self.greenhouse_state_btn.setGeometry(QtCore.QRect(370, 20, 191, 61))
        self.greenhouse_state_btn.setObjectName("greenhouse_state_btn")
        self.select_plant_btn = QtWidgets.QPushButton(self.centralwidget)
        self.select_plant_btn.setGeometry(QtCore.QRect(370, 200, 191, 61))
        self.select_plant_btn.setObjectName("select_plant_btn")
        self.exit_btn = QtWidgets.QPushButton(self.centralwidget)
        self.exit_btn.setGeometry(QtCore.QRect(370, 380, 191, 61))
        self.exit_btn.setObjectName("exit_btn")
        MainMenuWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(MainMenuWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 982, 26))
        self.menubar.setObjectName("menubar")
        MainMenuWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(MainMenuWindow)
        self.statusbar.setObjectName("statusbar")
        MainMenuWindow.setStatusBar(self.statusbar)


       #Botoiak sakatzean ekintza ezberdinak burutu:
        self.greenhouse_state_btn.clicked.connect(lambda:
self.openGreenHouseWindow(MainMenuWindow, socket, db, info_file, wr1, wr2, wr3))
        self.database_btn.clicked.connect(lambda:
self.openDatabaseWindow(MainMenuWindow, socket, db, info_file, wr1, wr2, wr3))
        self.select_plant_btn.clicked.connect(lambda:
self.openLoginWindow(MainMenuWindow, socket, db, info_file, wr1, wr2, wr3,
"plantWindow"))
        self.add_user_btn.clicked.connect(lambda: self.openLoginWindow(MainMenuWindow,
socket, db, info_file, wr1, wr2, wr3, "userWindow"))
        self.exit_btn.clicked.connect(lambda: self.exit(MainMenuWindow, wr1, wr2,
wr3))
```

```
        self.retranslateUi(MainMenuWindow)
        QtCore.QMetaObject.connectSlotsByName(MainMenuWindow)



    #Berotegiaren egoera ikusteko leihoa zabaldu.
    def openGreenHouseWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2,
wr3):
        self.GreenHouseWindow = QtWidgets.QMainWindow()
        self.ui=Ui_GreenhouseWindow()
        self.ui.setupUi(self.GreenHouseWindow, socket, db, info_file, wr1, wr2, wr3)
        self.GreenHouseWindow.show()
        MainMenuWindow.hide()



    #Datu historikoak ikusteko leihoa zabaldu.
    def openDatabaseWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2,
wr3):
        self.DatabaseWindow = QtWidgets.QMainWindow()
        self.ui=Historic_data.Ui_DatabaseWindow()
        self.ui.setupUi(self.DatabaseWindow, socket, db, info_file, wr1, wr2, wr3)
        self.DatabaseWindow.show()
        MainMenuWindow.hide()



    #Login egiteko leihoa zabaldu.
    def openLoginWindow(self, MainMenuWindow, socket, db, info_file, wr1, wr2, wr3,
window_to_open):
        self.LoginWindow = QtWidgets.QMainWindow()
        self.ui=Ui_LoginWindow()
        self.ui.setupUi(self.LoginWindow, MainMenuWindow, socket, db, info_file, wr1,
wr2, wr3, window_to_open)
        self.LoginWindow.show()



    #Irten botoia sakatzean, prozesu guraso eta ume guztiak eteteko agindua bidali
    def exit(self, MainMenuWindow, wr1, wr2, wr3):
        os.write(wr1, "exit".encode())
        os.write(wr2, "exit".encode())
        os.write(wr3, "exit".encode())
        os.close(wr1)
        os.close(wr2)
        os.close(wr3)
        time.sleep(1)
        MainMenuWindow.close()
        sys.exit(0)



    def retranslateUi(self, MainMenuWindow):
        _translate = QtCore.QCoreApplication.translate
        MainMenuWindow.setWindowTitle(_translate("MainMenuWindow", "Menu Nagusia"))
        self.database_btn.setText(_translate("MainMenuWindow", "Historikoa"))
        self.add_user_btn.setText(_translate("MainMenuWindow", "Erabiltzaile berriak
gehitu"))
        self.greenhouse_state_btn.setText(_translate("MainMenuWindow", "Berotegiaren
egoera"))
        self.select_plant_btn.setText(_translate("MainMenuWindow", "Landare mota
hautatu"))
        self.exit_btn.setText(_translate("MainMenuWindow", "Irten"))
```

Select_plant.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import socket, select
import Main_Menu

class Ui_SelectPlantWindow(object):
    def setupUi(self, SelectPlantWindow, socket, db, info_file, wr1, wr2, wr3):
        SelectPlantWindow.setObjectName("SelectPlantWindow")
        SelectPlantWindow.resize(984, 512)
        SelectPlantWindow.setStyleSheet("QWidget\n"
"{\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
"    border-color: white;\n"
"    border-width: 2px;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(SelectPlantWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.menu_back_btn = QtWidgets.QPushButton(self.centralwidget)
        self.menu_back_btn.setGeometry(QtCore.QRect(840, 390, 101, 51))
        self.menu_back_btn.setObjectName("menu_back_btn")
        self.plant_database_info = QtWidgets.QTableWidget(self.centralwidget)
        self.plant_database_info.setGeometry(QtCore.QRect(470, 70, 319, 192))
        self.plant_database_info.setRowCount(0)
        self.plant_database_info.setColumnCount(3)
        self.plant_database_info.setObjectName("plant_database_info")
        self.plant_database_info.setHorizontalHeaderLabels(['Izena', 'Tenp. max.',
'Hez. min.'])

self.plant_database_info.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView
.Stretch)
        self.delete_plant_btn = QtWidgets.QPushButton(self.centralwidget)
        self.delete_plant_btn.setGeometry(QtCore.QRect(490, 290, 101, 51))
        self.delete_plant_btn.setObjectName("delete_plant_btn")
        self.select_plant_btn = QtWidgets.QPushButton(self.centralwidget)
        self.select_plant_btn.setGeometry(QtCore.QRect(650, 290, 101, 51))
        self.select_plant_btn.setObjectName("select_plant_btn")
        self.max_temp_input = QtWidgets.QLineEdit(self.centralwidget)
        self.max_temp_input.setGeometry(QtCore.QRect(240, 120, 131, 31))
        self.max_temp_input.setObjectName("max_temp_input")
        self.max_hum_lbl = QtWidgets.QTextEdit(self.centralwidget)
```

```
        self.max_hum_lbl.setGeometry(QtCore.QRect(60, 180, 161, 31))
        self.max_hum_lbl.setObjectName("max_hum_lbl")
        self.max_hum_lbl.setReadOnly(True)
        self.error_text = QtWidgets.QTextEdit(self.centralwidget)
        self.error_text.setGeometry(QtCore.QRect(40, 390, 571, 31))
        self.error_text.setObjectName("error_text")
        self.max_temp_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.max_temp_lbl.setGeometry(QtCore.QRect(60, 120, 161, 31))
        self.max_temp_lbl.setObjectName("max_temp_lbl")
        self.max_temp_lbl.setReadOnly(True)
        self.plant_name_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.plant_name_lbl.setGeometry(QtCore.QRect(60, 70, 171, 31))
        self.plant_name_lbl.setObjectName("plant_name_lbl")
        self.plant_name_lbl.setReadOnly(True)
        self.create_plant_btn = QtWidgets.QPushButton(self.centralwidget)
        self.create_plant_btn.setGeometry(QtCore.QRect(170, 290, 101, 51))
        self.create_plant_btn.setObjectName("create_plant_btn")
        self.plant_name_input = QtWidgets.QLineEdit(self.centralwidget)
        self.plant_name_input.setGeometry(QtCore.QRect(240, 70, 131, 31))
        self.plant_name_input.setText("")
        self.plant_name_input.setObjectName("plant_name_input")
        self.max_hum_input = QtWidgets.QLineEdit(self.centralwidget)
        self.max_hum_input.setGeometry(QtCore.QRect(240, 180, 131, 31))
        self.max_hum_input.setObjectName("max_hum_input")
        self.selected_plant_text = QtWidgets.QTextEdit(self.centralwidget)
        self.selected_plant_text.setGeometry(QtCore.QRect(640, 20, 221, 31))
        self.selected_plant_text.setObjectName("selected_plant_text")
        self.selected_plant_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.selected_plant_lbl.setGeometry(QtCore.QRect(440, 10, 201, 51))
        self.selected_plant_lbl.setObjectName("selected_plant_lbl")
        self.selected_plant_lbl.setReadOnly(True)
        SelectPlantWindow.setCentralWidget(self.centralwidget)
        self.statusbar = QtWidgets.QStatusBar(SelectPlantWindow)
        self.statusbar.setObjectName("statusbar")
        SelectPlantWindow.setStatusBar(self.statusbar)


        #Botoi bakoitza sakatzean ekintza ezberdin bat gauzatu:
        self.create_plant_btn.clicked.connect(lambda: self.insertPlantDatabase(db))
        self.delete_plant_btn.clicked.connect(lambda: self.deletePlantDatabase(db,
info_file))
        self.select_plant_btn.clicked.connect(lambda: self.selectPlant(db, socket))
        self.menu_back_btn.clicked.connect(lambda: self.goToMenu(SelectPlantWindow,
socket, db, info_file, wr1, wr2, wr3))


        #Datu baseko datuak pantailan kargatu:
        self.showPlants(db)


        self.retranslateUi(SelectPlantWindow)
        QtCore.QMetaObject.connectSlotsByName(SelectPlantWindow)

        #Irekitzean, hautatuta dagoen landarea pantailaratu:
        self.showSelectedPlant(db, socket, info_file)



    #Zein landare hautatuta dagoen itzuli
    def showSelectedPlant(self, db, socket, info_file):
        socket.send("WPS".encode())
        info_file.seek(61)
        selected_plant_id = str(info_file.read(5).decode()).lstrip("0")
        cursor = db.cursor(buffered=True)
        cursor.execute("SELECT name FROM plants WHERE id='%s'" % selected_plant_id)
        plant_name = str(cursor.fetchall()[0])[2:-3]
        self.selected_plant_text.setText(plant_name)


    #Erabiltzaileak definitutako datuekin landare berria datu basean gorde.
    def insertPlantDatabase(self, db):
        self.error_text.setText("")
        try:
            plant_name = self.plant_name_input.text()
```

```
            temp_max = int(self.max_temp_input.text())
            hum_max = int(self.max_hum_input.text())
            if plant_name and temp_max and hum_max:
                cursor = db.cursor(buffered=True)
                cursor.execute("SELECT MAX(id) FROM plants")
                max_id = str(cursor.fetchall()[0])
                if max_id[1:-2] == "None":
                    max_id_int=0
                else:
                    max_id_int=int(max_id[1:-2])
                #Erabiltzaileak sartutako izenarekin landarerik existitzen den
                #aztertu, eta ez bada existitzen gorde.
                cursor.execute("SELECT name FROM plants WHERE name='%s'" % plant_name)
                if cursor.rowcount == 0:
                    cursor.execute("INSERT INTO plants VALUES ('%i', '%s', '%i',
'%i')" % (max_id_int+1, plant_name, int(temp_max), int(hum_max)))
                    db.commit()
                    self.showPlants(db)
                else:
                    self.error_text.setText("Izen horretako landarea existitzen da
jada.")
                cursor.close()
            else:
                self.error_text.setText("Datu guztiak beharrezkoak dira.")
        except ValueError:
            self.error_text.setText("Tenperaturak eta hezetasunak zenbakizkoak izan
behar dute.")



    #Erabiltzaileak hautatu dituen landarea datu basetik ezabatu.
    def deletePlantDatabase(self, db, info_file):
        selected_plant_list = self.plant_database_info.selectionModel().selectedRows()
        cursor = db.cursor(buffered=True)
        #Hautatutako landareak banan banan ezabatu.
        for plant_index in selected_plant_list:
            plant_name = str(self.plant_database_info.item(plant_index.row(),
0).text())
            cursor.execute("SELECT id FROM plants WHERE name='%s'" % (plant_name))
            delete_plant_id = str(cursor.fetchall()[0])[1:-2]
            info_file.seek(61)
            selected_plant_id = str(info_file.read(5).decode()).lstrip("0")
            if delete_plant_id != selected_plant_id:
                cursor.execute("DELETE FROM plants WHERE name='%s'" % (plant_name))
                #Aldaketak datu basean gorde.
                db.commit()
            else:
                self.error_text.setText("Ezin da hautatutako landarea ezabatu")
        self.showPlants(db)
        cursor.close()



    #Berotegiaren parametroak zein landaretarako optimizatuko diren hautatu.
    def selectPlant(self, db, socket):
        selected_plant_list = self.plant_database_info.selectionModel().selectedRows()
        cursor = db.cursor()
        #Landare bakarra hautatu dela ziurtatu.
        if len(selected_plant_list)==0:
            self.error_text.setText("Ez duzu landarerik hautatu")
        elif len(selected_plant_list)>1:
            self.error_text.setText("Landare bakarra hautatu daiteke")
        else:
            #Hautatutako landarearen izena taulatik eskuratu.
            plant_name =
str(self.plant_database_info.item(selected_plant_list[0].row(), 0).text())
            max_temp = str(self.plant_database_info.item(selected_plant_list[0].row(),
1).text())[:-2]
            max_hum =
str(self.humToADC(str(self.plant_database_info.item(selected_plant_list[0].row(),
2).text())[1:]))
```

```
            #Landareak datu basean duen id-a eskuratu.
            cursor.execute("SELECT id FROM plants WHERE name='%s'" % (plant_name))
            selected_plant_id = str(cursor.fetchall()[0])[1:-2]

            #Hautatutako landarearen identifikatzailea, tenperatura maximoa eta
            #hezetasun maximoa zehazteko aginduak bidali.
            command = ["WPP+", "TEM+", "HUM+"]
            data = [selected_plant_id, max_temp, max_hum]
            for i in range(3):
                set_plant_id_message = command[i]+data[i].zfill(5)
                socket.send(set_plant_id_message.encode())
                print("Bidali da komandoa: " + set_plant_id_message)
            self.selected_plant_text.setText(plant_name)
        cursor.close()


    def humToADC(self, humidity):
        return int(int(humidity)*1023/100)


    #Datu baseko informazioa kargatu.
    def showPlants(self, db):
        self.plant_database_info.setRowCount(0)
        cursor = db.cursor()
        cursor.execute("SELECT name, maxtemp, minhum FROM plants")
        database_plants = cursor.fetchall()
        row_number = 0
        for plant in database_plants:
            self.plant_database_info.insertRow(row_number)
            #Landare izena:
            self.plant_database_info.setItem(row_number, 0,
QtWidgets.QTableWidgetItem(plant[0]))
            #Tenperatura:
            self.plant_database_info.setItem(row_number, 1,
QtWidgets.QTableWidgetItem(plant[1]+"Â°C"))
            #Hezetasuna:
            self.plant_database_info.setItem(row_number, 2,
QtWidgets.QTableWidgetItem("%"+plant[2]))
            row_number+=1
        self.plant_database_info.setRowHidden(0, True)
        cursor.close()




    #Menu nagusia pantailaratu.
    def goToMenu(self, SelectPlantWindow, socket, db, info_file, wr1, wr2, wr3):
        self.MainMenuWindow = QtWidgets.QMainWindow()
        self.ui=Main_Menu.Ui_MainMenu()
        self.ui.setupUi(self.MainMenuWindow, socket, db, info_file, wr1, wr2, wr3)
        self.MainMenuWindow.show()
        SelectPlantWindow.hide()




    def retranslateUi(self, SelectPlantWindow):
        _translate = QtCore.QCoreApplication.translate
        SelectPlantWindow.setWindowTitle(_translate("SelectPlantWindow", "Landare
kudeaketa"))
        self.menu_back_btn.setText(_translate("SelectPlantWindow", "Menura itzuli"))
        self.delete_plant_btn.setText(_translate("SelectPlantWindow", "Ezabatu"))
        self.select_plant_btn.setText(_translate("SelectPlantWindow", "Hautatu"))
        self.max_hum_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Hezetasun
min.:</span></p></body></html>"))
        self.error_text.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
```

```
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.max_temp_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Tenperatura
max.:</span></p></body></html>"))
        self.plant_name_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC
\"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Landare
izena:</span></p></body></html>"))
        self.create_plant_btn.setText(_translate("MainWindow", "Sortu"))
        self.selected_plant_text.setHtml(_translate("MainWindow", "<!DOCTYPE HTML
PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.selected_plant_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML
PUBLIC \"-//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Hautatuta
dagoen landarea:</span></p></body></html>"))
```

User_window.py

```
from PyQt5 import QtCore, QtGui, QtWidgets
import Main_Menu

class Ui_UserWindow(object):
    def setupUi(self, UserWindow, socket, db, info_file, wr1, wr2, wr3):
        UserWindow.setObjectName("UserWindow")
        UserWindow.resize(984, 512)
        UserWindow.setStyleSheet("QWidget\n"
"{\n"
#"    font-size: 15px;\n"
"    background:#a0ce4e;\n"
"}\n"
"\n"
"QToolButton\n"
"{\n"
"    background: #a0ce4e;\n"
"    border:0px;\n"
"    \n"
"}\n"
"\n"
"QPushButton\n"
"{\n"
"    border-style: solid;\n"
```

```
"    border-color: white;\n"
"    border-width: 2px;\n"
"    font-weight: 400;\n"
"    text-align: center;\n"
"}\n"
"\n"
"\n"
"QPushButton:hover\n"
"{\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"\n"
"QTextEdit\n"
"{\n"
"    border: 0px;\n"
"}\n"
"\n"
"QLineEdit\n"
"{\n"
"    border-width: 0px;\n"
"    border-color: #DDDDDD;\n"
"    color:#DDDDDD;\n"
"}\n"
"")
        self.centralwidget = QtWidgets.QWidget(UserWindow)
        self.centralwidget.setObjectName("centralwidget")
        self.create_user_btn = QtWidgets.QPushButton(self.centralwidget)
        self.create_user_btn.setGeometry(QtCore.QRect(120, 330, 101, 51))
        self.create_user_btn.setObjectName("create_user_btn")
        self.username_input = QtWidgets.QLineEdit(self.centralwidget)
        self.username_input.setGeometry(QtCore.QRect(190, 30, 131, 31))
        self.username_input.setText("")
        self.username_input.setObjectName("username_input")
        self.password_input = QtWidgets.QLineEdit(self.centralwidget)
        self.password_input.setGeometry(QtCore.QRect(190, 80, 131, 31))
        self.password_input.setObjectName("password_input")
        self.password_input.setEchoMode(QtWidgets.QLineEdit.Password)
        self.password_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.password_lbl.setGeometry(QtCore.QRect(10, 80, 121, 31))
        self.password_lbl.setObjectName("password_lbl")
        self.password_lbl.setReadOnly(True)
        self.username_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.username_lbl.setGeometry(QtCore.QRect(10, 30, 171, 31))
        self.username_lbl.setObjectName("username_lbl")
        self.username_lbl.setReadOnly(True)
        self.name_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.name_lbl.setGeometry(QtCore.QRect(10, 140, 121, 31))
        self.name_lbl.setObjectName("name_lbl")
        self.name_lbl.setReadOnly(True)
        self.name_input = QtWidgets.QLineEdit(self.centralwidget)
        self.name_input.setGeometry(QtCore.QRect(190, 140, 131, 31))
        self.name_input.setObjectName("name_input")
        self.surname_1_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.surname_1_lbl.setGeometry(QtCore.QRect(10, 190, 121, 31))
        self.surname_1_lbl.setObjectName("surname_1_lbl")
        self.surname_1_lbl.setReadOnly(True)
        self.surname_1_input = QtWidgets.QLineEdit(self.centralwidget)
        self.surname_1_input.setGeometry(QtCore.QRect(190, 190, 131, 31))
        self.surname_1_input.setText("")
        self.surname_1_input.setObjectName("surname_1_input")
        self.surname_2_lbl = QtWidgets.QTextEdit(self.centralwidget)
        self.surname_2_lbl.setGeometry(QtCore.QRect(10, 240, 121, 31))
        self.surname_2_lbl.setObjectName("surname_2_lbl")
        self.surname_2_lbl.setReadOnly(True)
        self.surname_2_input = QtWidgets.QLineEdit(self.centralwidget)
        self.surname_2_input.setGeometry(QtCore.QRect(190, 240, 131, 31))
        self.surname_2_input.setObjectName("surname_2_input")
        self.menu_back_btn = QtWidgets.QPushButton(self.centralwidget)
        self.menu_back_btn.setGeometry(QtCore.QRect(840, 390, 101, 51))
        self.menu_back_btn.setObjectName("menu_back_btn")
        self.error_text = QtWidgets.QTextEdit(self.centralwidget)
        self.error_text.setGeometry(QtCore.QRect(40, 400, 571, 31))
        self.error_text.setObjectName("error_text")
```

```
        self.user_database_info = QtWidgets.QTableWidget(self.centralwidget)
        self.user_database_info.setGeometry(QtCore.QRect(440, 20, 415, 291))
        self.user_database_info.setRowCount(1)
        self.user_database_info.setColumnCount(4)
        self.user_database_info.setObjectName("user_database_info")
        self.user_database_info.setHorizontalHeaderLabels(['Erab. izena', 'Izena',
'Abizena1', 'Abizena2'])

self.user_database_info.horizontalHeader().setSectionResizeMode(QtWidgets.QHeaderView.
Stretch)
        self.delete_user_btn = QtWidgets.QPushButton(self.centralwidget)
        self.delete_user_btn.setGeometry(QtCore.QRect(605, 330, 101, 51))
        self.delete_user_btn.setObjectName("delete_user_btn")
        UserWindow.setCentralWidget(self.centralwidget)
        self.menubar = QtWidgets.QMenuBar(UserWindow)
        self.menubar.setGeometry(QtCore.QRect(0, 0, 984, 26))
        self.menubar.setObjectName("menubar")
        UserWindow.setMenuBar(self.menubar)
        self.statusbar = QtWidgets.QStatusBar(UserWindow)
        self.statusbar.setObjectName("statusbar")
        UserWindow.setStatusBar(self.statusbar)


       #Sakatutako botoi bakoitzeko ekintza gauzatu
        self.create_user_btn.clicked.connect(lambda: self.createUser(db))
        self.delete_user_btn.clicked.connect(lambda: self.deleteUser(db))
        self.menu_back_btn.clicked.connect(lambda: self.goToMenu(UserWindow, socket,
db, info_file, wr1, wr2, wr3))

        #Erabiltzaileak datu basetik kargatu
        self.showUser(db)

        self.retranslateUi(UserWindow)
        QtCore.QMetaObject.connectSlotsByName(UserWindow)



    #Erabiltzaile berri bat sortu.
    def createUser(self, db):
        self.error_text.setText("")
        #Interfazeko datuak irakurri.
        username = self.username_input.text()
        password = self.password_input.text()
        name = self.name_input.text()
        surname_1 = self.surname_1_input.text()
        surname_2 = self.surname_2_input.text()

        #Datu guztiak sartu direla egiaztatu.
        if username and password and name and surname_1 and surname_2:
            cursor = db.cursor(buffered=True)
            #Erabiltzaileari identifikatzailea esleitzeko, datu basean dagoen handiena
             #eskuratu.
            cursor.execute("SELECT MAX(id) FROM users")
            max_id = str(cursor.fetchall()[0])
            if max_id[1:-2] == "None":
                max_id_int=0
            else:
                max_id_int=int(max_id[1:-2])

            #Datu basean erabiltzeaile hori existitzen ez dela ziurtatu.
            cursor.execute("SELECT username FROM users WHERE username='%s'" %
username)
            if cursor.rowcount == 0:
                cursor.execute("INSERT INTO users VALUES ('%i', '%s', '%s', '%s',
'%s', '%s')" % (int(max_id_int+1), username, password, name, surname_1, surname_2))
                db.commit()
            else:
                self.error_text.setText("Erabiltzaile izen hori ez dago erabilgarri")
            cursor.close()
        else:
            self.error_text.setText("Datu guztiak sartzea beharrezkoa da.")
        self.showUser(db)
```

```
    #Erabiltzaileak datu basetik ezabatu.
    def deleteUser(self, db):
        self.error_text.setText("")
        selected_user_list = self.user_database_info.selectionModel().selectedRows()
        cursor = db.cursor()
        for user_index in selected_user_list:
            username = str(self.user_database_info.item(user_index.row(), 0).text())
            cursor.execute("DELETE FROM users WHERE username='%s'" % (username))
            db.commit()
        self.showUser(db)
        cursor.close()



    #Datu baseko erabiltzaileak interfazean pantailaratu.
    def showUser(self, db):
        self.user_database_info.setRowCount(0)
        cursor = db.cursor()
        cursor.execute("SELECT username, name, surname1, surname2 FROM users")
        database_users = cursor.fetchall()
        row_number = 0
        for user in database_users:
            self.user_database_info.insertRow(row_number)
            for column_number in range(4):
                self.user_database_info.setItem(row_number, column_number,
QtWidgets.QTableWidgetItem(user[column_number]))
            row_number+=1
        self.user_database_info.setRowHidden(0, True)
        cursor.close()



    #Menu nagusira itzuli.
    def goToMenu(self, UserWindow, socket, db, info_file, wr1, wr2, wr3):
        self.MainMenuWindow = QtWidgets.QMainWindow()
        self.ui=Main_Menu.Ui_MainMenu()
        self.ui.setupUi(self.MainMenuWindow, socket, db, info_file, wr1, wr2, wr3)
        self.MainMenuWindow.show()
        UserWindow.hide()


    def retranslateUi(self, UserWindow):
        _translate = QtCore.QCoreApplication.translate
        UserWindow.setWindowTitle(_translate("UserWindow", "Erabiltzaile kudeaketa"))
        self.create_user_btn.setText(_translate("UserWindow", "Sortu"))
        self.password_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-
size:12pt;\">Pasahitza:</span></p></body></html>"))
        self.username_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Erabiltzaile
izena:</span></p></body></html>"))
        self.name_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
```

```
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-
size:12pt;\">Izena:</span></p></body></html>"))
        self.surname_1_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Abizena
1:</span></p></body></html>"))
        self.surname_2_lbl.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\" margin-top:0px; margin-bottom:0px; margin-left:0px; margin-right:0px; -
qt-block-indent:0; text-indent:0px;\"><span style=\" font-size:12pt;\">Abizena
2:</span></p></body></html>"))
        self.menu_back_btn.setText(_translate("MainWindow", "Menura itzuli"))
        self.error_text.setHtml(_translate("MainWindow", "<!DOCTYPE HTML PUBLIC \"-
//W3C//DTD HTML 4.0//EN\" \"http://www.w3.org/TR/REC-html40/strict.dtd\">\n"
"<html><head><meta name=\"qrichtext\" content=\"1\" /><style type=\"text/css\">\n"
"p, li { white-space: pre-wrap; }\n"
"</style></head><body style=\" font-family:\'MS Shell Dlg 2\'; font-size:15px; font-
weight:400; font-style:normal;\">\n"
"<p style=\"-qt-paragraph-type:empty; margin-top:0px; margin-bottom:0px; margin-
left:0px; margin-right:0px; -qt-block-indent:0; text-indent:0px;\"><br
/></p></body></html>"))
        self.delete_user_btn.setText(_translate("UserWindow", "Ezabatu"))
```

C Eranskina: ESP32ko kodea

ESP32.py

```
import usocket, select, network
from machine import UART


def connect():
  station = network.WLAN(network.STA_IF);

  station.active(True);

  station.connect("ArduinoTest", "123456abc")

  station.isconnected()
  station.ifconfig()

def main():

  #Initializar UART
  uart = UART(1, 9600)
  uart.init(9600, bits=8, parity=None, stop=1, tx=17, rx=16)

  #Wi-Fira konektatu
  connect()

  #UDP konexioak onartu
  s = usocket.socket(usocket.AF_INET, usocket.SOCK_DGRAM)
  sockaddr = ('', 3030)
  s.bind(sockaddr)

  #Mezuak iristea itxaron
  while True:
    while True:
      datuak_jasotzeko, _, _ = select.select( [s], [], [], 1)
      if datuak_jasotzeko:
        data, client_addr = s.recvfrom(1024)
        if not data:
          break

        print(data)
        data_string = data.decode()

        uart.write(data_string)

        uart_data = uart.read()
        while not uart_data:
          uart_data = uart.read()
        uart_string = uart_data.decode()
        s.sendto(uart_string.encode(), client_addr)
        print(uart_data)

  s.close()

if __name__ == "__main__":
  main()
```