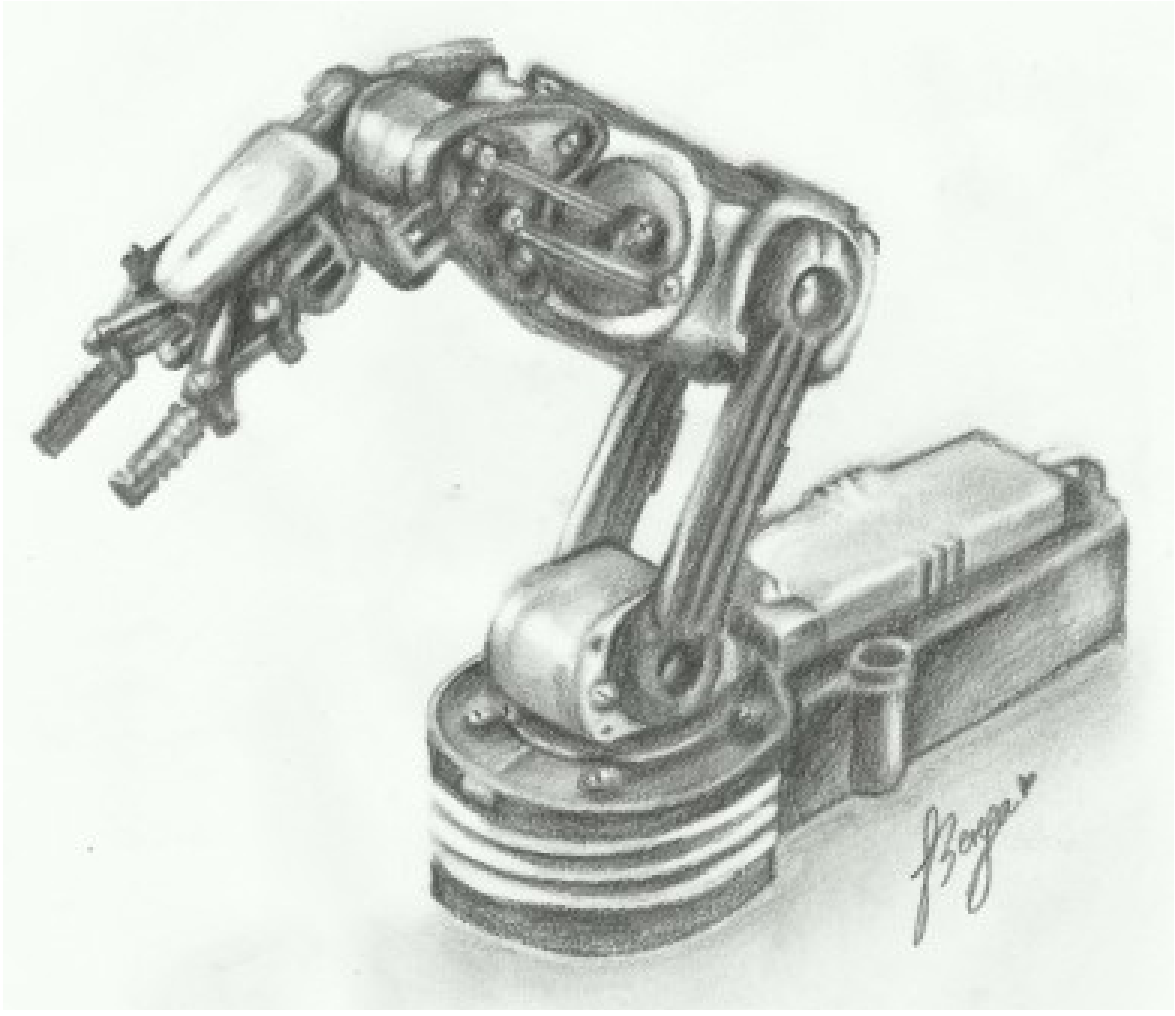
ROBOTIKAREN HASTAPENAK

ARDUINO PLAKA ERABILITA



Eskerrik asko, Teresa Baskaran,
alaba maitea,
azaleko irudia marrazten hartu duzun lan handiagatik,
eta gehiagogatik…

instagram: @teresa_baskaran

Aurkibidea

1. SARRERA

Robotikaren hastapenak Arduino plaka erabilita liburua DBH 4ko ikasleekin lan egiteko dago diseinatuta, hau da, 15 eta 16 urtekoekin.

Robotika gero eta gehiago lantzen da 4. DBHko teknologia-liburuetan. Hain zuzen ere, horixe ezartzen du Curriculumak, eta uste dut teknologiako irakasleak ahalegintzen direla euren ikasleei robotika azaltzen, gutxienez, era teorikoan. Alde praktikoari dagokionez, ordea, nahiko zaila da proiektuak gauzatzea testuliburuetan datorren informazioa erabiliz, haietan proposaturikoak ez dira-eta oso egingarriak.

Arduino plaka: robotika lantzeko baliabide eskuragarria

Zorionez, orain dela urte batzuk merkatuan dira mikrokontrolagailua duten hainbat plaka merke, besteak beste, Arduino plaka −sortu, elektronikako zirkuituak astialdian muntatzea gogoko duten zaleentzat sortu ziren−, eta aukera paregabea eskaintzen du 4. DBHn robotika ikasten hasteko.

Batetik, Arduino plakak merkeak dira, hain zuzen, behar-beharrezko osagai elektronikoak dituelako bakarrik. Garatu beharreko proiektuen arabera, ordea, aukera ematen dute beste plaka osagarri batzuk (shield) gehitzeko.

Bestetik, Arduino plakaren hardwarea librea da, eta, ondorioz, plaka klonikoak edo ofizialak erosi daitezke, norberak muntatu…

Gainera, Arduino plaka kontrolatzeko softwarea Windowsen, Linuxen, zein Android sistema eragileetan instalatu daiteke, eta, horri esker, konputagailuetan, tabletetan zein Smartphone gailuetan ere instalatu daiteke.

Azkenik, Arduino plaka kontrolatzeko erabiltzen den programazio-lengoaiak ere erraztu egiten du programak idaztea eta ulertzea (C lengoaian oinarritutako lengoaia erraztua da), eta, gainera, robotikaren hastapenetan erabiltzen diren programak ere samurrak dira.

Liburu honen edukiak

Liburuak bi atal nagusi ditu. Lehenengoan, Arduino plakarekin lanean hasi aurretik jakin beharrekoak azaldu dira: plakaren hardwarearen ingurukoak, hura kontrolatzeko behar den softwareari buruzkoak eta nola programatzen den. Bigarrenean, proiektuak muntatzeko argibideak eta hamalau proiektu-proposamen daude (zirkuitua, beharrezko osagaiak eta programa).

Esan behar da, bestalde, liburuan sartu ez bada ere, ezinbestekoa dela elektronika analogikoaren eta digitalaren oinarriak landuta izatea proposatutako proiektuei ekin aurretik. Baina dagoeneko eduki horiek lantzen dira 4. DBHn, eta eskura daude hainbat testu-liburutan.

Liburu honetako hamalau proiektuak zailtasunaren arabera ordenatu dira: hasierako proiektu errazen ondoren, zailtxoagoak datoz. Kontu handiz aukeratu ditut proiektuok, eta ez dut sartu programa zaila behar duen proiekturik, nahiz eta ikusgarria izan; adin horretako ikasleek ulertzeko erako proiektuak besterik ez.

Hala ere, horrek ez du esan nahi proposatutako proiektuak elkartu eta emaitza ikusgarri edo konplexuagoak lortu ezin daitezkeenik. Baina hori ikasleen trebetasunaren eta irudimenaren menpe geratzen da, eta irakaslearen lana izango da sormen hori sustatzeko baldintzak jartzea. Adibidez, aparkaleku itxia diseinatzen badute:

1) Ibilgailua aparkalekuaren sarreran dagoela jakiteko ultrasoinu sentsorea erabili daiteke.

2) Aparkalekuaren sarrerako langa jasotzeko serbomotorea erabili daiteke.

3) Aparkalekuan zenbat toki dauden libre bistaratu daiteke LCD pantailan.

Proiektu hauen bidez, ez da espezialistarik sortu nahi; bai, ordea, ikasleek eginez ikastea. Hori lortzeko, ezinbestekoa da, besteak beste, taldean ganoraz lan egiten ikastea, intuizioari eta sormenari lekua uztea (eduki hutsak ikastera mugatu gabe), ekinaren ekinez sakontzeko gogoa piztea eta autonomia maila egokia lortzea; azken baten, inguruan dituzten gailuak eta sistemak behatu eta proiektu errazak diseinatzea.

1. ARDUINO PLAKAREKIN LANEAN HASI AURRETIK JAKIN BEHARREKOAK

Sarreran esan den bezala, lan honen azken helburua ikasleek Arduino plakak erabili eta hainbat proiektu gauzatzea da. Proiektuei ekin aurretik, ordea, lehenbizi, Arduino plakaren hardwarea eta softwarea azalduko dira.

1.1. ARDUINO UNO R3 PLAKAREN HARDWAREA

Arduino Uno R3 plaka aukeratu da lan egiteko, eta hari dagozkio honako azalpen hauek; hala ere, beste motaren bat darabilenari ere lagungarri izango zaizkio, hemen aipaturikoak nahiko azalpen orokorrak baitira.

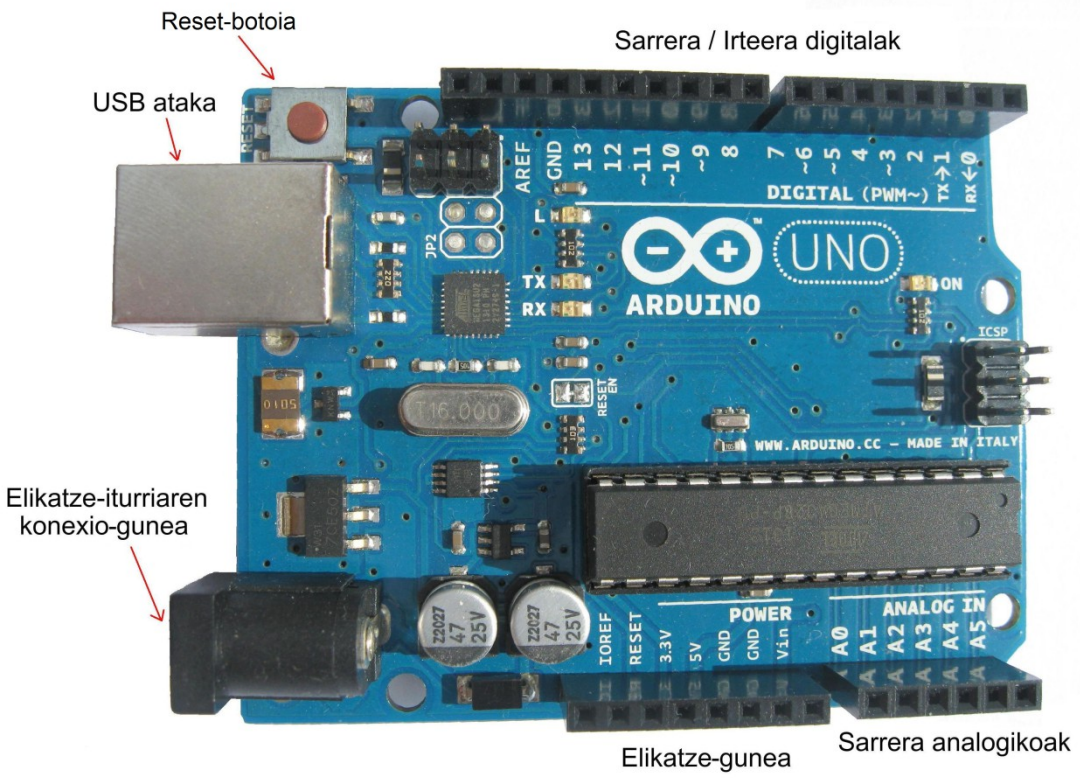
Hardwareari buruzko azalpenean honakoak azalduko dira, hurrenez hurren:

- Arduino UNO R3 plakaren osagaiak
- Arduino UNO R3 plakaren ezaugarri orokorrak
- Arduino UNO R3 plaka elikatzeko aukerak
- Non erosi Arduino UNO R3 plaka

1.1.1. ARDUINO UNO R3 PLAKAREN OSAGAIAK

Arduino Uno R3 plakak osagai hauek ditu:

- ATmega328 mikrokontrolagailua
- 14 sarrera/irteera digital: 0, 1, 2, ~3, 4, ~5, ~6, 7, 8, ~9, ~10, ~11, 12 eta 13 (horietatik sei PWM motakoak dira, hain zuzen ere, ikur hau dutenak: ~)
- 6 sarrera analogiko: A0, A1, A2, A3, A4 eta A5.
- ICSP konexio-gunea (*In Circuit Serial Programming*)
- 16 MHZ-eko erresonadore zeramikoa
- USB ataka
- elikatze-iturriaren konexio-gunea
- reset-botoia



1.1.2. ARDUINO UNO R3 PLAKAREN EZAUGARRI OROKORRAK

Hauek dira Arduino Uno R3 plakaren ezaugarri orokorrak:

- lan-tentsioa: 5 V
- gomendatutako elikatze-tentsioak: 7 V-tik 12 V-ra
- elikatze-tentsioaren mugak: gutxienez, 6 V; gehienez, 20 V
- gehieneko korrontea pineko sarrera zein irteera legez: 40 mA
- gehieneko korrontea 3,3 V-ko pinean: 50 mA
- Flash memoria: 32 KB (horietatik 0,5 KB *Bootloader* programak erabiltzen ditu)
- SRAM memoria: 2 KB
- EEPROM memoria: 1 KB

1.1.3. ARDUINO UNO R3 PLAKA ELIKATZEKO AUKERAK

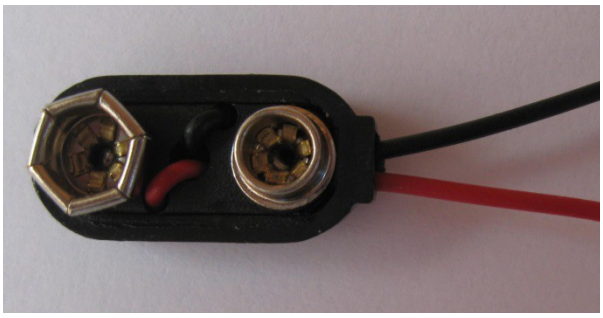
Hainbat era daude Arduino plaka elikatzeko, eta zer komeni den huraxe erabiliko da, aukeratutako proiektuaren arabera:

- Proiektua ez bada osorik mugitzen, USB ataka bidez, pila bidez edo elikatze-iturri bat konektatuz elikatu daiteke. Azken aukeran, 2,1 mm-ko konektorea erabili behar da, erdian positiboa duena.
- Proiektua osorik mugitzen bada, hau da, ibilgailua-edo bada, elikatzeko era bakarra pila da. Horretarako, lau pila erabili daitezke, 6 V lortzeko, edo 9 V-ko pila bakarra.

Pilak konektatzeko gune bi daude aukeran:

1) **Elikatze-iturriaren konexio-gunea**: gune hau erabiliz gero, lehen aipatutako 2,1 mm-ko konektorea lortu beharko da, eta pila multzoa konektatu.

2) **Elikatze-gunea (*power):*** gune hau erabiliz gero, berriz, V_{IN} sarreran konektatuko da pilaren positiboa, eta GND (masa) sarreran negatiboa.



1.1.4. NON EROSI ARDUINO UNO R3 PLAKA

Arduino plakak erosteko leku ofiziala web-orri honetan dago: *arduino.cc*, eta hauexek dira jarraitu beharreko urratsak, gaur egun:

1. Web-orri horretan sartu eta *buy* (erosi) aukeratuta, beste web-orri batera jotzen da: *store.arduino.cc*.
2. Bigarren orri horretan, *distributors* (banatzaileak) aukeratuta, *arduino.cc/en/main/buy* web-orrira sartzen da.
3. Hirugarren orri horretan, Arduinoaren banatzaile ofizialak daude, kontinente eta naziotan sailkatuak. Ondoren, denda bat aukeratu, eta denda horretan proiektuak egiteko behar diren osagaiak erosi behar dira, hala nola Arduino plakak, *Arduino starter kit*a, eta abar.

1.2. ARDUINO IDE SOFTWAREA

Arduino plaka eta konputagailua ondo konektatu direla egiaztatu ondoren, IDE softwarea zelan instalatu jakin behar da, baina, horretan hasi aurretik, zenbait kontzeptu argitu behar dira.

1.2.1. AURRETIKO KONTZEPTUAK

- IDE (*Integrated Development Environment*): garapen-ingurune integratua da IDE; izan ere,

hiru funtzio integratzen ditu bere baitan: programak idazteko testu-editorea dauka, programa konpilatzen du eta programa hori Arduino plakara bidaltzen du.

- *Sketch* (literalki, *Zirriborro*): IDEren menuan *sketch* deitzen zaie programei, Arduino plaka hasiera baten artistentzat eta horrelakoentzat sortu ei zen eta.

1.2.2. ZELAN INSTALATU IDE SOFTWAREA

KONPUTAGAILUAN INSTALATZEKO

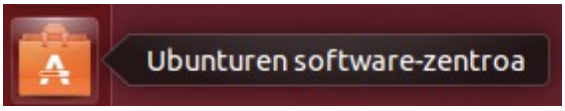
IDE softwarearen bertsio ofiziala *arduino.cc/download* lekuan eskuratzen da. Leku horretan, hainbat bertsioren artean aukeratu behar da, norberaren gailuak zer sistema eragile duen kontuan hartuta.
Merkatuan dauden sistema eragile gehienentzat dago softwarea eskuragarri web-orri ofizialean:

Sistema eragilea	Web-orria
Windows	*arduino.cc/windows*
OS X 10.5 edo berriagoa	*arduino.cc/mac*
Linux	*arduino.cc/linux*

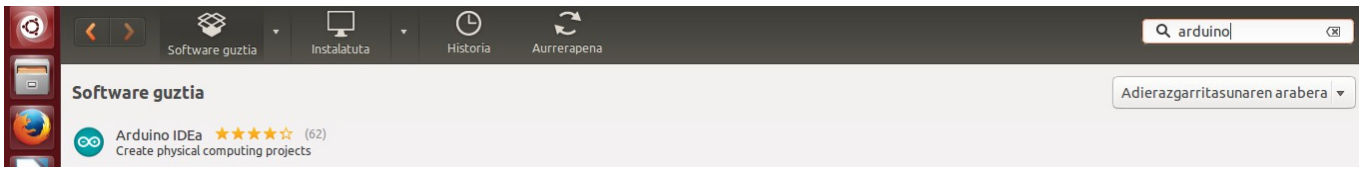
Lan honetan, hala ere, Linux Ubuntu-n zelan instalatzen den azaltzen da.

Zelan instalatu Arduino IDE-a Ubuntun

1. Joan Ubunturen software-zentrora.



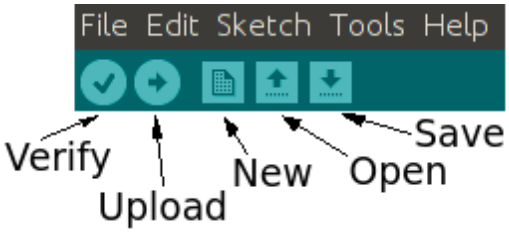
2. Idatzi *arduino* hitza bilatzeko elkarrizketa-koadroan, eta sakatu ENTER.



3. Bilatutako programen artean, aukeratu Arduino IDEa eta instalatu.

IDE softwarearen menua Ubuntun

IDE softwarearen menua antzekoa da sistema eragile guztietan. Behin Ubuntu sistema eragilearentzako IDE softwarea instalatu ondoren, abiatu eta honako menu hau agertuko da:



Goiko lerroa

- *File* (fitxategia)
- *Edit* (editatu)
- *Sketch* (programa)
- *Tools* (tresnak)
- *Help* (laguntza)

Beheko lerroa

- *Verify* (egiaztatu),
- *Upload* (bidali),
- *New* (berria),
- *Open* (ireki)
- *Save* (gorde).

ANDROID SISTEMA ERAGILEDUN GAILUETAN INSTALATZEKO

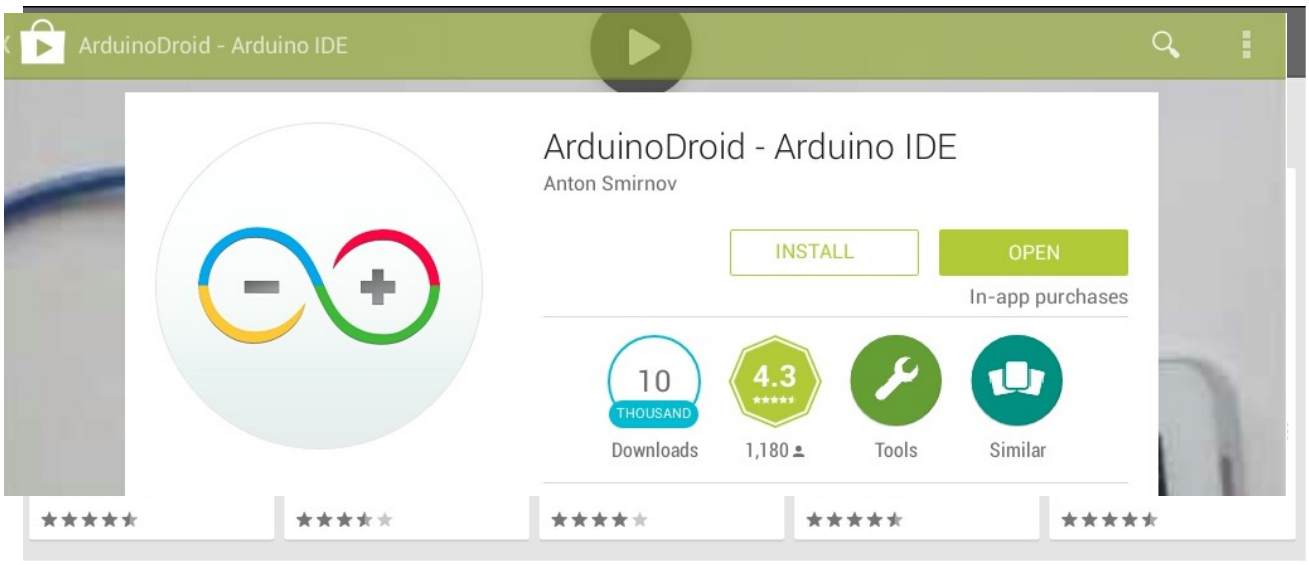
Nahiz eta ofiziala ez izan, badago aukera doaneko Arduino IDE softwarea instalatzeko Android sistema eragilea duten gailuetan. Hala ere, telefono-pantailak oso txikiak direla kontuan hartuta, eroso lan egitekotan, gutxienez 10 hazbeteko pantailekin lan egin behar da; izan ere, pantailan traba egiten dute softwarearen iragarkiek. Iragarkirik eduki nahi ez bada, erosi egin behar da softwarea.

Instalazio-prozesua

1. Joan *Play Store* gunera.

2. Idatzi *arduino ide* bilatzeko elkarrizketa koadroan , eta hainbat aplikazio azalduko dira.

3. Aukeratu Anton Smirnov-ek diseinatutako ArduinoDroid-Arduino IDE aplikazioa.
4. Instalatu.



1.2.3. ZELAN KONEKTATU ELKARREKIN KONPUTAGAILUA ETA ARDUINO PLAKA

Arduinoaren eta konputagailuaren arteko datu-transferentzia egin ahal izateko, honako pauso hauek jarraitu behar dira.

1. Konektatzea
    a. Konektatu Arduino plaka eta konputagailua kable bidez. Horretarako, erabili irudian ageri den moduko USB kablea.

b. Joan *Tools* (Tresnak) menura, eta zehaztu zer Arduino mota erabiliko den eta zein USB atakatan ezarri den.

Tools menuan, honako aukera hauek daude:

- *Auto format* *Ctrl+T*
- *Archive Sketch*
- *Fix Encoding & Reload*
- *Serial Monitor* *Ctrl+Shift+M*
- *Board*
- *Serial Port*
- *Programmer*
- *Burn Bootloader*

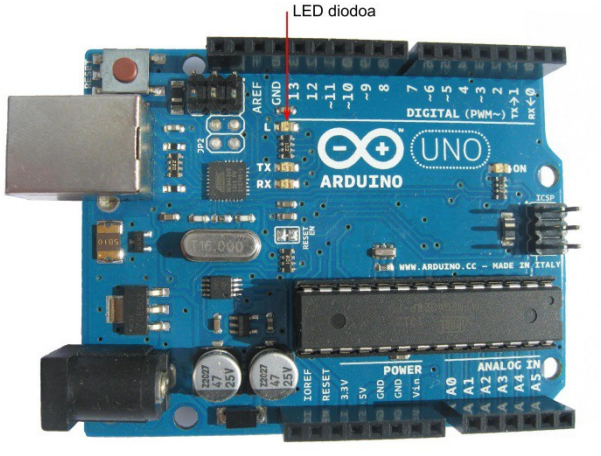
b.1. Sakatu *Board* eta aukeratu erabiliko den Arduino plaka mota.

b.2. Sakatu *Serial Port* eta aukeratu han eskaintzen den USB ataka.

Hori horrela, datuak seriean transferitu ahal izango dira Arduino plakaren eta konputagailuaren artean.

1.2.4. ZELAN EGIAZTATU KONPUTAGAILUAREN ETA ARDUINO PLAKAREN ARTEKO KONEXIOA

(Hemen azalduko den egiaztatze-prozedura, Arduino Uno R3 plakari dagokio, batik bat.)



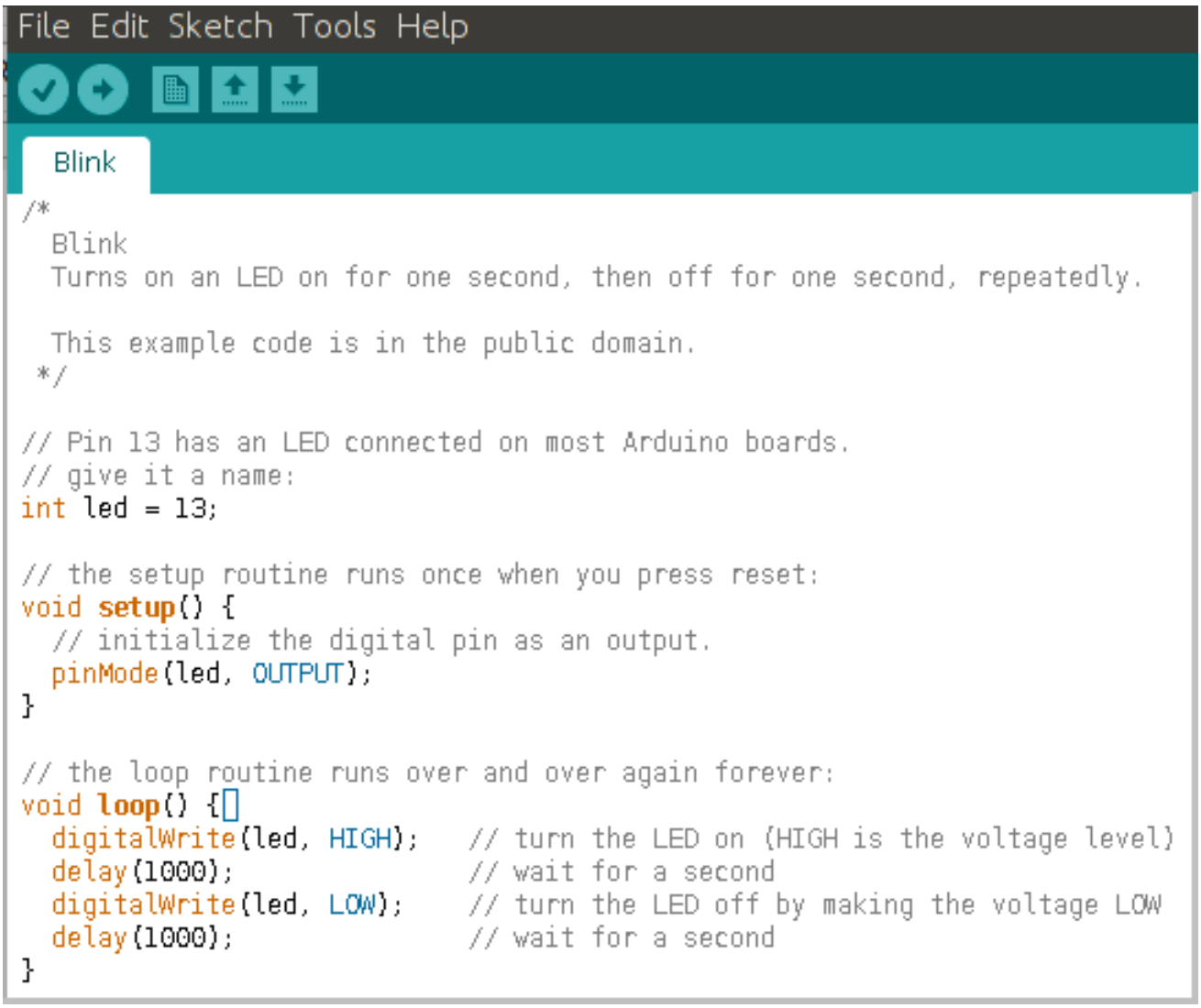
Konexioa ondo egin den egiaztatzeko, Arduino UNO R3 plakan L hizkiaren ondoan dagoen LED diodoa erabiltzen da.

Beste mota batzuetan ez bezala, Arduino Uno R3 plaka erosi berriak *Blink* programa dakar memorian, eta, horregatik, L hizkiaren ondoan dagoen LED diodoa etengabe piztu eta amatatzen da,

eta egoera bakoitzak 1 s irauten du. Konputagailura ondo konektatuta dagoen egiaztatzeko, blink programa aldatu egin behar da, hain zuzen ere, LED diodoaren pizte-itzaltze maiztasuna aldatuta. Egindako aldaketek LED diodoan eraginik badute, ongi konektatu den seinale izango da.

Arduino IDE softwareak dagoeneko eginda dauden hainbat programa eskaintzen ditu, eta, esan bezala, konexioa egiaztatzeko, *Blink* programa erabiltzen da. Hona hemen jarraitu behar diren pausoak:

1. Aukeratu *File* eta, ondoren, *Examples.*
   IDE softwareak berak dakartzan hainbat programa-ereduen multzoak aterako dira.
2. Sakatu *01.Basics* programa-ereduen multzoan, eta aukeratu *Blink* programa ateratzen direnen artean. Pantailan programa hau azalduko da:



3. *Blink* programan `delay (1000);` azaltzen den lerro bietan, `1000` dagoen lekuan idatzi `100`.
4. Sakatu *Verify,* programa egiaztatzeko eta konpilatzeko, hau da, programa kode bitarrera bihurtzeko.
5. Sakatu *Upload,* programa bitarra Arduino plakara bidaltzeko.

Arduino plaka konputagailura ondo konektatuta badago, L hizkiaren ondoan dagoen LED diodoa etengabe piztu eta amatatuko da, eta egoera bakoitzak 0,1 s (100 ms) iraungo du, hau da, nabarmen arinago.
Hori ez bada gertatzen, arazoa non dagoen bilatu beharko da.

Ohiko arazoak lau dira:

1. Plaka hondatuta egotea.
2. USB kablea hondatuta egotea.
3. Plaka mota gaizki aukeratzea menuan (*Tools, Board*).
4. Gaizki adieraztea konputagailuaren zein USB atakatan dagoen konektatuta plaka (*Tools, Serial Port*).

(Azken arazo horrek makina bat buruhauste eman ohi ditu Windows XP sistema eragilea erabiltzen denean)

2. PROGRAMAZIOA

2.1. ARDUINO PLAKA PROGRAMATZEA

Arduinoa programatzeko, honako prozesu hau jarraitzen da:

1. Erabiltzaileak, Arduino IDEra sartu, eta nahi den programa idazten du; horretarako, Arduino lengoaia erabiltzen da.
2. Programa idatzi ondoren, IDE softwareak programa hori egiaztatzen du.
3. Ondoren, behar izanez gero, programa-bibliotekak gehitzen ditu IDEak (*libraries*).
4. Gero, idatzitako programa C lengoaiara itzultzen du.
5. Eta, ostean, konpilatu egiten du C lengoaiara itzulitakoa; hau da, kode bitarrera bihurtzen du. Kontuan hartu behar da, hain zuzen ere, kode bitarra bakarrik ulertzen dutela mikrokontrolagailuek.
6. Azkenik, IDEak kode bitarra plakara bidaltzen du, han exekutatzeko.

Beraz, erabiltzaileak pauso hauek jarraitu behar ditu programatzeko:

1. Sakatu *New* beheko lerroan, programa berria idazten hasteko.
2. Idatzi programa.
3. Gorde. Gordetzeko aukera bi daude:
   a. *File* aukeratzea goiko lerroan, eta, ondoren, *Save As* sakatzea (gorde honela) irekitako menuan*:* aukera honi esker, programa gorde eta komeni den izena jartzea dago.
   b. *Save* aukeratzea beheko lerroan: programa gorde egingo da, baina izenik ezarri gabe.

   (Kontuan izan programa bakoitzeko karpeta bat sortzen duela eta bertan gordetzen duela idatzitako programa)
4. Sakatu *Verify*.
5. Sakatu *Upload*.

2.1.1. ARDUINOREN PROGRAMAZIO LENGOAIA

C lengoaian oinarrituta dago Arduino plaka programatzeko erabiltzen den lengoaia, baina ez da C estandarra, ezaugarri batzuk falta ditu-eta. Lengoaia hori, C estandarra baino errazagoa denez, oso aproposa da robotikako lehen urratsak ematen hasteko, eta, gainera, robotikako proiektu batzuk gauzatzeko nahiko baliabide eskaintzen ditu.

Hurrengo lerroetan lengoaia horretan idatzitako programen egitura, aldagai motak eta funtzioak azalduko dira.

2.2. ARDUINO PROGRAMEN EGITURA

Arduino lengoaian idatzitako programa baten oinarrizko egitura hauxe da:

A) Iruzkina
B) `#include`
C) Aldagaiak
D) `void setup()` funtzioa
E) `void loop()` funtzioa

A) IRUZKINA

Programatzaileek iruzkinak txertatzen dituzte programetan beste erabiltzaile batzuek programa horien nondik norakoak erraz ulertzeko. Izan ere, lan itzela izan daiteke orain dela bizpahiru urte idatzitako programa luzea ulertzea.

Programaren hasieran, beti txertatzen da iruzkina, programaren zeregin orokorra azaltzeko, eta, gainera, programan zehar ere tartekatzen dira iruzkinak, aldagai bat zertarako erabiltzen den edo programa zatitxo baten funtzionamendua azaltzeko.

Iruzkinak emateko era bi daude:

A) Esaldi luzea bada, ikur hauen artean: `/*  */`

Adibidez:

```
/*   Programa honek semaforoa kontrolatzeko
     balio du.                              */
```

B) Lerro bateko esaldia bada, ikur hauekin hasita: `//`

```
// sentsorea, 2 pinean
```

DBHko ikasleekin lan egiteko, proposaturiko proiektuetako programetan, hasierako iruzkina besterik ez da txertatu, programak ez astuntzeko.

B) #include <izena.h>

Funtzio hau erabilita, programa-biblioteka bat (*library*) gehitzen zaio programari. Jakina, beharrizanen arabera, era bateko edo besteko programa-biblioteka gehitzen dira. Haiek eskainitako funtzio bereziak erabilita, oso erraz kontrolatuko dira LCD pantaila, serbomotorea, eta abar.

Adibidez:

```
#include <Servo.h>
```

(Kontuan izan amaieran ez dela puntu eta koma (;) idazten)

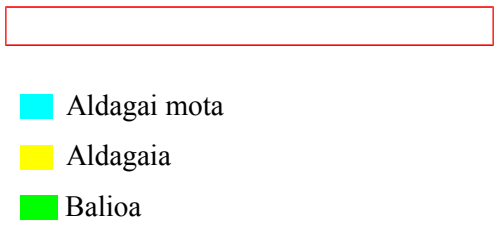
C) ALDAGAIAK

Arduino plakarako programa idaztea, hain zuzen ere, agindu-segida bat idaztea da, lan jakin bat egin dezan. Hori dela eta, zehatz-mehatz azaldu behar diogu zer aldagai erabili eta zer eragiketa egin. Aldagaiak, bestalde, asko izan daitezke, besteak beste, osagai elektronikoa non konektatu den eta sentsorearen balioa (tenperatura, argitasuna…).

Programaren hasieran, aldagaiak zein diren idatzi behar da, eta batzuetan hasierako balioa ere ematen zaie. Horrez gain, aldagaia zer motatakoa den ere zehaztu behar da. Kontuan izan beti puntu eta komarekin (;) amaitu behar direla aldagaien lerroak. Ganoraz ez bada idazten, konpilatzaileak egindako akatsari buruzko informazioa bistaratuko du, eta zuzendu egin beharko da.

```
int ledgorria= 5;

int tenperatura = 18;
```



Kontuan izan aldagaiei jarritako izenak gogorarazi egin behar duela zertarako erabiltzen den aldagai hori. Aldagai batzuk *orokorrak* izaten dira, hau da, programa osoan erabiltzeko aukera dago; hala ere, aldagai bat *lokala* izan daiteke, eta programaren zatitxo batek bakarrik erabiliko du.

Aldagai bakoitzak zer zehaztasun-maila behar duen kontuan izanda, leku gehiago edo gutxiago hartuko du memorian. Horregatik, aldagaiak hartuko duen balioaren arabera, aldagai mota zehaztu behar da.

- ALDAGAI MOTAK

Arduinoren programazio-lengoaian erabiltzen diren aldagai mota ohikoenak hauek dira: `byte`, `int`, `long` eta `float`.

aldagai mota	byte kopurua	balio-tartea	erabilera
byte	1	0tik 255era	zenbaki positibo osoak
int	2	-32768tik 32767ra	zenbaki positibo zein negatibo osoak
long	4	-2 147 483 648tik 2 147 483 647ra	zenbaki positibo zein negatibo osoak
float	4	$-3{,}4028235\cdot10^{38}$tik $3{,}4028235\cdot10^{38}$ra	zenbaki frakzionario positiboak zein negatiboak

Beharbada, ikasleren batek galdetuko du ea zergatik ez den zehazten aldagai guztiak `float` motakoak direla. Bada, kontua da mikrokontrolagailuak duen memoria oso txikia dela eta ezinbestekoa dela ahal den eta memoria gutxien hartzen duen aldagai mota erabiltzea.

Hona hemen aldagai mota gehiago:

mota	tamaina	balio-tartea

boolean	byte 1	true (1)	false (0)
byte	byte 1	0	255
char	byte 1	-128	127
unsigned char	byte 1	0	255
int	byte 2	-32768	32767
unsigned int	byte 2	0	65535
long	4 byte	-2 147 483 648	2 147 483 647
unsigned long	4 byte	0	4 294 967 295
float	4 byte	$-3{,}4028235 \cdot 10^{38}$	$3{,}4028235 \cdot 10^{38}$
array	aldakorra	datuen zerrenda	

D) `void setup()` FUNTZIOA

`void setup()` funtzioa derrigorrez idatzi behar da programan. Haren zereginik ohikoenak honako hauek dira:

1) Mikrokontrolagailuaren zein pin izango diren datuak sartzeko eta zein datuak irteteko.

2) Serieko datu-transferentzia zer abiaduratan gauzatuko den (zenbat bit segundoko, bps) Arduino plakaren eta konputagailuaren artean.

Funtzio hau giltza honekin hasi { eta honekin } amaitu behar da.

Adibidea:

```
void setup(){
pinMode(ledgorria, OUTPUT);
pinMode(ledhoria, OUTPUT);
}
```

E) `void loop()` FUNTZIOA

Funtzio hau giltza honekin hasi { eta honekin } amaitu behar da. Giltza bi horien artean dagoen programa zatia etengabe exekutatzen da.

Begizta horretatik irteteko ohiko erak bi dira: plakak duen *reset* botoia sakatuta edo plakari elikadura kenduta.

2.3. FUNTZIOAK

Programaren egitura jakindakoan, programatzeko erabili beharreko funtziorik ohikoenak zelan idatzi behar diren eta funtzioen artean zer aukera dagoen jakin behar da.

2.3.1. FUntzioak idazteko arauak

Funtzioak idazteko zenbait arau bete behar dira:

- Funtzioetako gako-hitzak ingelesez idazten dira.
- Funtzioa izendatzen duen gako-hitza araututa dago, eta ezin da beste era baten idatzi. Bestela, akatsa egin dela esango du konpilatzaileak. Adibidez: `pinmode` edo `pin Mode` idatzita, arazoak izango dira; izan ere `pinMode` idatzi behar da.
- Funtzio gehienen amaieran puntu eta koma (;) idatzi behar da.
- Funtzio baten esparrua giltza hauen bitartez zehazten da: {}

2.3.2. Funtziorik ohikoenak

Hona hemen Arduino plaka programatzeko erabilitako funtziorik ohikoenak:

A) `balioa=analogRead(pina)`

Funtzio honek sarrera analogikoak irakurtzeko balio du, hau da, sarrerako tentsioa. Arduino Uno R3 plakak sei sarrera analogiko ditu: A0, A1, A2, A3, A4 eta A5. Sarrerako ohiko tentsioa, berriz, 0 V eta 5 V bitartekoa da proposatutako proiektuetan.

`AnalogRead` funtzioaren bidez, sarrerako tentsioaren balio analogikoa zenbaki bitar bihurtzen du Arduino plakaren ADC bihurgailuak (*Analog to Digital Converter)*. Hain zuzen ere, ADC bihurgailua 10 bit-ekoa da, eta, ondorioz, 2^{10} (=1024) balio eman ditzake. Ematen duen zenbaki bitar osoa aldagaietako baten gordetzen da. Esaterako, programan honako hau idazten bada:

```
balioa = analogRead(2);
```

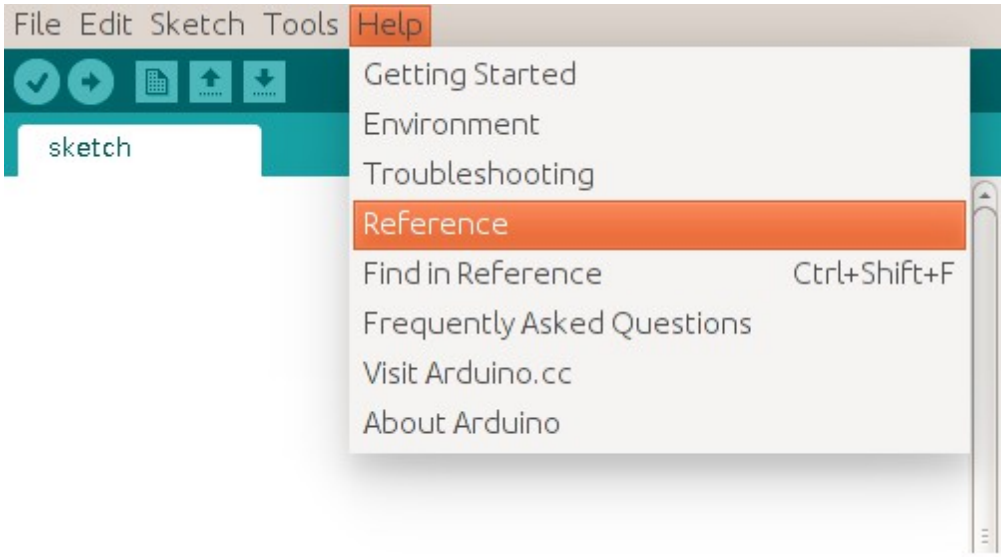
2 zenbakiko pineko tentsioaren lagina hartu, eta, balio bitar bihurtu ondoren, `balioa` aldagaian gordetzen du. Egiten duen bihurketa hauxe da:

Hona hemen sarrerako tentsio jakin batzuetarako `analogRead` funtzioak ematen dituen balioak:

Sarrerako tentsioa (V)	`analogRead` funtzioak emandako balio bitarra
0	00 0000 0000
1	00 1100 1100
2	01 1110 0001
3	10 0110 0101
4	11 0011 0010
5	11 1111 1111

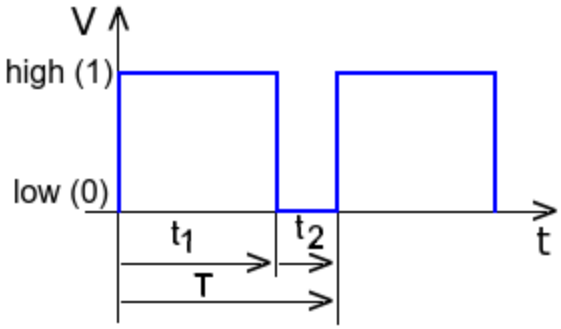
Hala ere, sarrerako gehieneko tentsioa 5 V baino txikiagoa izatea nahi bada, doiketa batzuk egin behar dira `analogRead` funtzioak bihurketa ondo egin dezan:

1. Ireki Arduino IDE programa.
2. Egin klik *Help* menuan.
3. Aukeratu *Reference* menuan, laguntza-orrira joateko.
4. Aukeratu *analogReference()*.
   Agertuko den orrian dakar zelan egiten den doiketa.



B) `analogWrite(pina, balioa);`

Funtzio hau erabiltzen bada, lan-ziklo (*duty cycle*) aldakorra duen uhina ateratzen da Arduino Uno R3 plakako 3, 5, 6, 9, 10 eta 11 irteeretatik (ikur hau dute: ~), eta funtzioan zehaztu beharko da zer balio nahi den eta horietako zein pinetatik aterako den uhina.



Pin horiek PWM (*pulse width modulation*) ezaugarria daukate, hau da, aukera dago t_1 denboraren iraupena 0tik T-raino aldatzeko (ikusi irudia).

`analogWrite` funtzioa erabilita, aipatutako irteeretatik ateratako maiztasun konstanteko uhinaren lan-zikloa aldatuko da: T konstantea dela kontuan hartuta, t_1 handituta, t_2 txikitu egiten da.

$$T = t_1 + t_2$$

$$\text{lan}-\text{zikloa} = \frac{t_1}{t_1 + t_2} \cdot 100$$

- Funtzioari eman dakiokeen balio txikiena 0 da, eta kasu horretan lan-zikloa = % 0 izango da.
- Funtzioari eman dakiokeen balio handiena 255 da, eta, ondorioz, lan-zikloa = % 100 izango

da.

`analogWrite()` funtzioaren aplikazioetako bat LED baten distira edo motor baten abiadura aldatzea izan daiteke. `analogWrite()` funtzioari emandako balioa txikia bada, t_l txikia izango da, eta, adibidez, LEDaren distira txikia izango da, eta emandako balioa handia bada, LEDaren distira handia izango da.

Kontuan izan, `analogWrite()` funtzioak ez duela balio pin analogikoak kontrolatzeko; horretarako, analogRead() funtzioa dago.

C) `delay(balioa);`

Funtzio honi esker, mikrokontrolagailuak, itxaron egiten du denbora jakin baten, funtzio berririk exekutatu barik. Horretarako, zenbaki bat idatzi behar da parentesi artean, mikrokontrolagailuak zenbat milisegundoz itxarongo duen zehazteko. Adibidez, `delay(4000);` idazten bada, mikrokontrolagailuak beste zeregin bat egin aurretik, 4000 ms = 4 s itxarongo du.

D) `delayMicroseconds(balioa);`

`Delay` funtzioak egiten duen lan bera egiten du `delayMicroseconds` funtzioak, baina mikrosegundoak baliatzen ditu, milisegundoak erabili beharrean. Adibidez,

```
delayMicroseconds(40);
```

idazten bada, mikrokontrolagailuak beste zeregin bat egin aurretik, 40 µs itxarongo du.
Kontuan izan funtzio hau zehatza dela 3 µs-tik 16383 µs-ra; beraz, denbora luzeagoz itxaroteko `delay()` funtzioa erabili behar da.

E) `digitalWrite(pina, balioa);`

Funtzio honek LOW eta HIGH balioak irteera digitaletara bidaltzeko balio du.

Horretarako, batetik, zer pin erabiliko den zehaztu behar da, eta, bestetik, zer balio irtengo den pin horretatik: LOW edo HIGH, hau da, 0 edo 1.

```
...
int led_hankaa = 7;
....
digitalWrite(led_hankaa, HIGH);
....
```

Adibidez, alboko programa zatiak HIGH balioa bidaltzen du 7 zenbakiko irteera digitalera.

F) `if  zerbait {...}`

Funtzio honen bidez, zereginak exekutatzeko baldintza edo baldintzak idazten dira: `zerbait` izeneko lekuan idatzitako baldintza(k) egia bad(ir)a, giltzen artean idatzitakoa exekutatuko da; bestela, hurrengo funtzioa exekutatuko da.

Adibidez:

```
if (digitalRead(5) == HIGH) {
      digitalWrite(7, HIGH);}
```

5 zenbakiko pin digitalean tentsio altua badago, 7 zenbakiko pin digitalean tentsio altua ezarriko du mikrokontrolagailuak, eta hurrengo funtzioa exekutatuko da. Baldintza ez bada betetzen, hurrengo funtzioa exekutatuko da.
(Kontuan izan konparazioak egiteko berdin ikurra (=) bi aldiz idatzi behar dela)

G) `if zerbait {...} else {...}`

Funtzio hau `if zerbait {}` funtzioa da oinarrian, baina, baldintza betetzen ez bada, bigarren giltzen artekoa exekutatuko da.

```
...
if (sentsore_balioa >200){
  digitalWrite(led_hankaa, LOW);}
  else{
    digitalWrite(led_hankaa, HIGH);}
...
```

Adibidez: alboko programa zatiak sentsorearen balioa irakurtzen du, eta balioa 200 baino handiagoa bada, LED bat amatatzen du; bestela, LED bat pizten du.

H) `map()funtzioa`

Funtzio honen bidez, bihurketa egiten da hiru balio-tarteen artean.

Adibidez:

```
balioa = map(sentsore_balioa, 0, 1023, 0, 179);
```

- Lehenengo balio-tartea 0tik 5era da; horiek dira, izan ere, pin analogiko baten gutxieneko eta gehieneko ohiko tentsioak.
- Bigarren balio-tartea 0tik 1023ra da, ADC zirkuituak 1024 balio ezberdin eman ditzakeelako.
- Hirugarren balio-tartea 0tik 179ra idazten bada, lortzen den balioa serbomotorearen biraketa-angelua zehazteko erabiltzen da.

Formula eran jarrita, honela da:

$$balioa = \frac{sentsore\_balioa \cdot 1023}{5} \cdot \frac{179}{1023}$$

Horrela lortzen den balio txikiena 0 izango da, eta handiena, 179.

I) `pinMode(pina,era);`

Funtzio honen bidez, pin jakin batek irteera (`OUTPUT`) edo sarrera (`INPUT`) legez funtzionatuko duen zehazten da. Hala ere, badago funtzionatzeko hirugarren era bat ere: `INPUT_PULLUP`. Funtzionatzeko era horretan, mikrokontrolagailuaren barneko *pullup* erresistentzia konektatuko da (berez ez da konektatuta egoten). Hala ere, azken aukera hori ez da erabiliko proposatutako proiektuetan.

Adibidez: alboko programa zatiak 7 zenbakiko pin digitalak irteera legez funtzionatuko duela zehazten du.

```
int led_pina = 7;
void setup()
{
     pinMode(led_pina, OUTPUT);
}
```

J) `pulseIn(pina, balioa)`

Funtzio honek pin jakin bateko balioak zenbat iraungo duen kalkulatzen du. Balioa HIGH edo LOW izan daiteke bakarrik.
Adibidez:

```
iraupena = pulseIn(7, HIGH);
```

Funtzio honek denbora kontatzen du, hasi 7 zenbakiko pinean HIGH balioa azaltzen den unetik eta LOW azaltzen denean amaituta; horrela, pultsuaren iraupena mikrosegundotan ematen du `iraupena` izeneko aldagaian.

K) `Serial.begin(abiadura);`

Funtzio honen bidez, Arduino plakaren eta konputagailuaren artean serieko datu-transferentzia zein abiaduratan gauzatuko den zehaztu behar da, zenbat bit segundoko (bps).
Komunikazio abiadura ohikoenak 300, 1.200, 2.400, 4.800, 9.600, 14.400, 19.200, 28.800, 38.400, 57.600 edo 115.200 bit segundoko dira. Proposatutako proiektuetan, 9.600 bps erabili da, eta programetan honela dago idatzita:

```
Serial.begin(9600);
```

L) `Serial.print(balioa, formatua)`

Funtzio honen bidez, Arduino plakak testua bidaltzen du konputagailuaren pantailara. Formatuaren balioa idatzita, aukera asko ematen ditu funtzio honek, baina zailegia litzateke hori guztia ikasmaila honetan.

Hala ere, zehaztasun gehiago jakiteko, arestian aipatu den lekuan begiratu daiteke (IDE programa> *Help> Reference> Serial> print()*).

Adibidez:

```
Serial.print("Gelako tenperatura epela da.")
```

idazten bada, konputagailuaren pantailan esaldi hau azalduko da:

```
Gelako tenperatura epela da.
```

Ostera,

```
Serial.print(distantzia);
```

idazten bada, konputagailuaren pantailan `distantzia` aldagaiaren balioa azalduko da.

Oharra:

`Serial.print` funtzioa erabiltzeko, aurretik serieko datu-transferentziaren abiadura zehaztu behar da.

M) `Serial.println(balioa, formatua)`

Funtzio hau `Serial.print` funtzioaren antzerakoa da, baina honako ezberdintasun hau du: testua pantailan bistaratu ondoren, kurtsorea hurrengo lerroaren hasierara joaten da, hau da, teklatuan *enter* sakatzen denean legez.

3. PROIEKTUAK

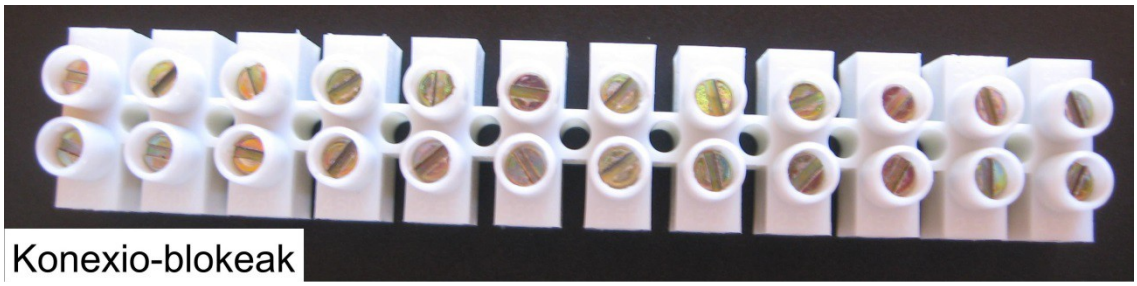
3.1. PROIEKTUAK MUNTATZEN HASI AURRETIK, ZEHAZTAPEN BATZUK

Proiektuak muntatzen hasi aurretik, komeni da jakitea zelan muntatzen diren proiektuak, *board* plaka zelan erabiltzen den eta zelan elikatzen diren proiektuak.

3.1.1. PROIEKTUAK MUNTATZEKO ERAK

Proiektuak muntatzeko era bi daude:

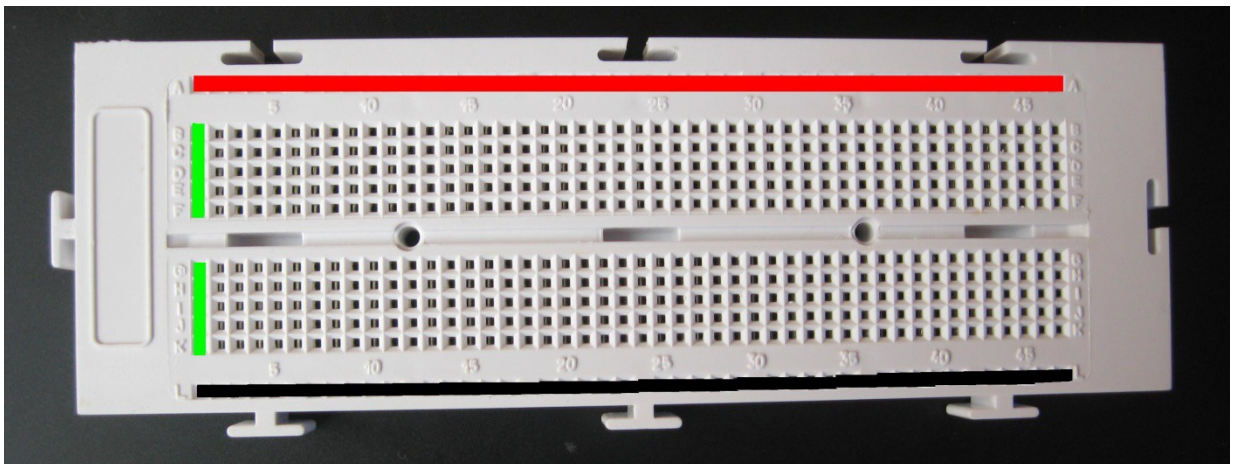
1) Osagai elektronikoak zirkuitu-inprimatu baten kokatu ondoren, soldatzea edo *wire-wrapping* teknika erabiltzea.
2) Osagai elektronikoak *Board* plakan kokatzea eta behar denean konexio-blokeak erabiltzea.


Konexio-blokeak

3.1.2. *BOARD* PLAKA

Institutuan lan egiteko, proiektuak *board* plaketan muntatzea proposatzen da; izan ere, horrek erraztu egiten du gero osagai elektronikoak askatzea eta beste proiektu baten erabiltzea.

Merkatuan mota askotako *board* plakak daude. Lan honetan, eredu gisa erabilitako *board* plakak zulotxoz jositako hiru gune ditu:



1)

Goiko lerro horizontala (gorriz markatuta): elkarrekin azpitik konektatuta daude lerro honetako zulotxoak, eta, normalean, elikatze-iturriaren edo pilaren positiboa konektatzen da.

2) Beheko lerro horizontala (beltzez markatuta): elkarrekin azpitik konektatuta daude lerro honetako zulotxoak, eta, normalean, elikatze-iturriaren edo pilaren negatiboa konektatzen da.

3) Erdiko gunea (berdez markatu da lehen zutabea): gune honetan konexio leku bi daude, erdiko errenkadaren albo banatan. Erdiko leku bi horiek ez daude elkarrekin elektrikoki konektatuta, ez eta goiko zein beheko lerroekin ere. Lerro bertikal bakoitzeko 5 zulotxoak, aldiz, elkarrekin konektatuta daude, baina ez alboko beste 5 zulotxoekin.

3.1.3. PROIEKTUAK ELIKATZEKO ERAK

Proiektuak elikatzeko, pilak edo elikatze-iturria erabili daiteke.

Proposatutako proiektuetan, 9 V-ko pila erabili da zirkuitua eta Arduino plaka elikatzeko, 14. proiektuan izan ezik. Hain zuzen ere, 6 V ematen dituen elikatze-iturria erabili da. Proiekturen baten 5 V behar izan denean, Arduino plakaren elikatze-gunetik hartu da.

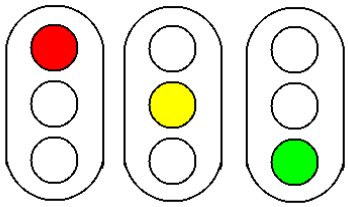
3.2. PROIEKTU PROPOSAMENAK

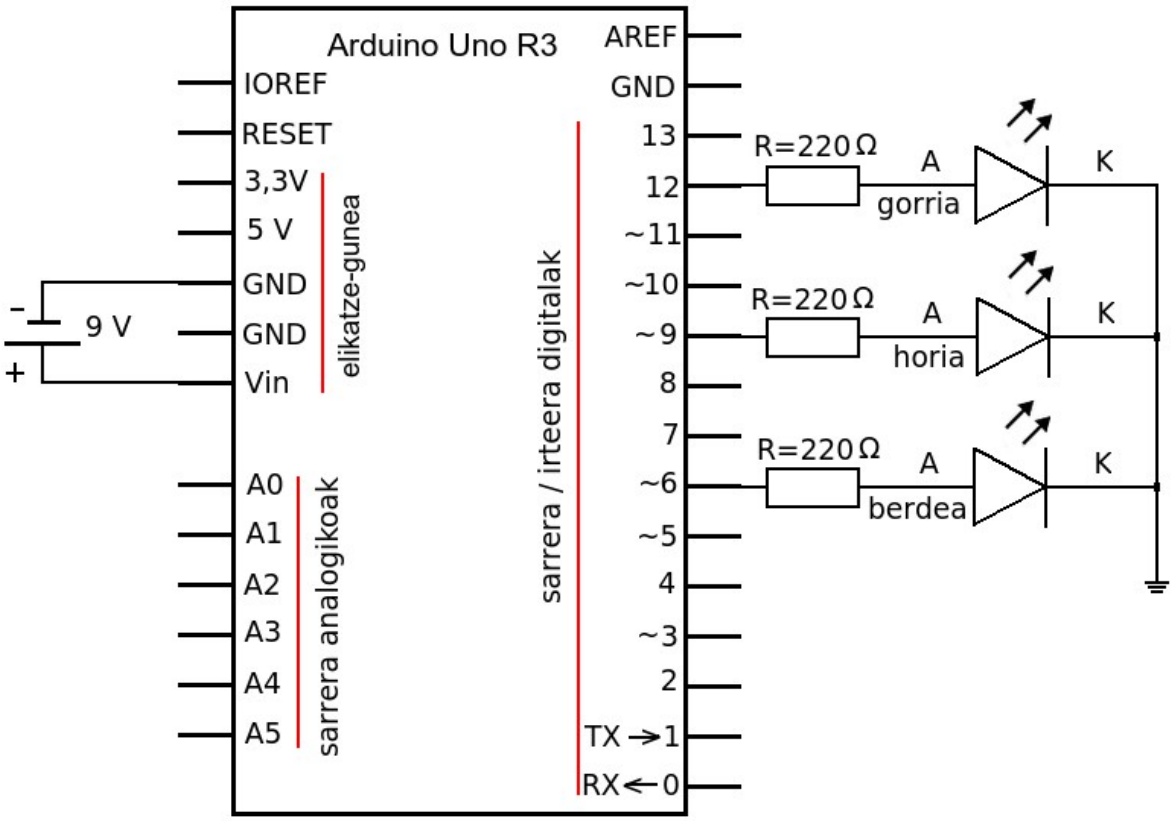
1. PROIEKTUA: IBILGAILUENTZAKO SEMAFOROA

Proiektu honetan, ibilgailuak kontrolatzeko hiru argi dituen semaforoa muntatzea proposatzen da.

Bestalde, badago proiektu hau oinarri hartuta beste aldaera batzuk muntatzea ere:

- LED horia kenduta, trenak kontrolatzeko semaforoa egin daiteke.
- LED urdinak erabilita eta `delay` denborak doituta,





ibilgailuen argiak diseinatu daitezke.

Zirkuitua

Osagaiak

Proiektu honetan osagai hauek behar dira:

* Arduino plaka

* 220 Ω-eko 3 erresistentzia

* 3 LED (gorria, horia eta berdea)

* 9 V-ko pila

Programa

semaforoa.ino

```
/*
  Programa honek semaforoa kontrolatzeko
balio du.                    */

int ledgorria = 12;
int ledhoria = 9;
int ledberdea = 6;



void setup(){

  pinMode(ledgorria, OUTPUT);
  pinMode(ledhoria, OUTPUT);
  pinMode(ledberdea, OUTPUT);
}

void loop(){
  digitalWrite(ledgorria, HIGH);
  digitalWrite(ledhoria, LOW);
  digitalWrite(ledberdea, LOW);
  delay(4000);
  digitalWrite(ledgorria, LOW);
  digitalWrite(ledhoria, HIGH);
  digitalWrite(ledberdea, LOW);
  delay(1000);
  digitalWrite(ledgorria, LOW);
  digitalWrite(ledhoria, LOW);
  digitalWrite(ledberdea, HIGH);
delay(4000);
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, **aldagaien** izenak erabaki dira: `ledgorria`, `ledhoria` eta `ledberdea`.
LED gorria 12 zenbakiko pinean, LED horia 9koan eta LED berdea 6koan muntatu beharko dira, aldagaien balioak horiek izatea erabaki delako.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

Gero, aipatutako **hankek** irteera legez (output) funtzionatuko dutela zehaztu da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

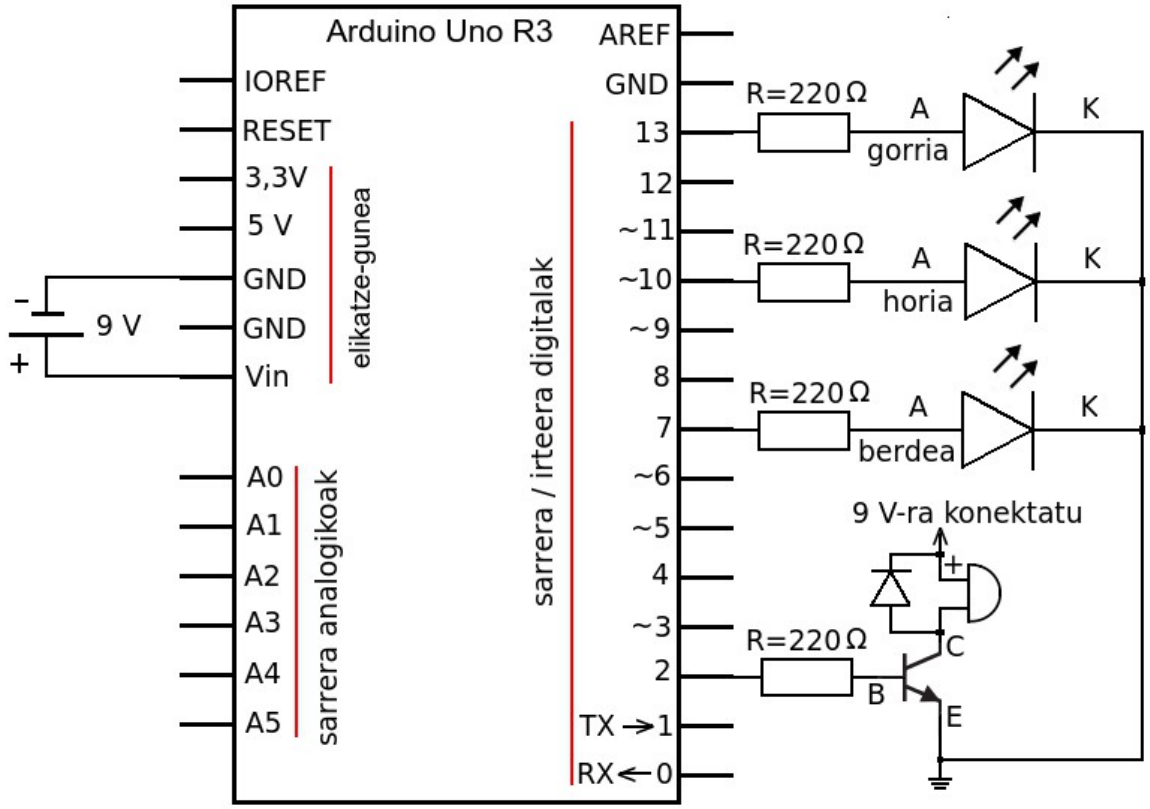
1) LED gorria piztu (HIGH) eta beste LED biak amatatu (LOW).
2) Itxaron (`delay`) 4000 ms = 4 s.
3) LED horia piztu eta beste LED biak amatatu.
4) Itxaron 1000 ms = 1 s.
5) LED berdea piztu eta beste LED biak amatatu.
6) Itxaron 4000 ms = 4 s.

Oharra: Programa bitarraren tamaina 1.188 byte da.

2. PROIEKTUA: ITSUENTZAT EGOKITUTAKO SEMAFOROA

Proiektu honetan, semaforoaren argi berdea piztuta dagoen bitartean, txirrin batek burrunba egiten du.

Zirkuitua



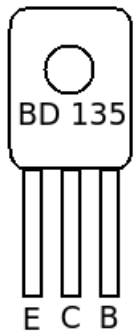
Osagaiak

Osagai hauek dira beharrezko proiektua egiteko:

* Arduino plaka
* 220 Ω-eko 4 erresistentzia
* 3 LED (gorria, horia eta berdea)
* BD 135 transistorea edo baliokidea
* txirrin bat
* diodoa
* 9 V-ko pila

BD 135 TRANSISTOREA

Oraingoan, transistore bat jarri behar zaio proiektuari, plakak ezin baitu eman txirrinak funtzionatzeko behar duen beste korronte.



Programa

itsuentzako_semaforoa.ino

```
/*  Programa honek oinezkoentzat, itsuak
barne, semaforoa diseinatzeko balio du,
                  */

int ledgorria = 13;
int ledhoria = 10;
int ledberdea = 7;
int txirrina = 2;

void setup(){
  pinMode(ledgorria, OUTPUT);
  pinMode(ledhoria, OUTPUT);
  pinMode(ledberdea, OUTPUT);
  pinMode(txirrina, OUTPUT);
}

void loop(){
  digitalWrite(ledgorria, HIGH);
  digitalWrite(ledhoria, LOW);
  digitalWrite(ledberdea, LOW);
  digitalWrite(txirrina, LOW);

  delay(4000);

  digitalWrite(ledgorria, LOW);
  digitalWrite(ledhoria, HIGH);
  digitalWrite(ledberdea, LOW);

  delay(1000);

  digitalWrite(ledgorria, LOW);
  digitalWrite(ledhoria, LOW);
  digitalWrite(ledberdea, HIGH);
  digitalWrite(txirrina, HIGH);

  delay(4000);
}
```

Programaren azalpena

Programaren iruzkina.

LED gorria 13 zenbakiko pinean, LED horia 10ekoan, LED berdea 7koan eta txirrina 2koan muntatu beharko dira, aldagaien balioak horiek izatea erabaki delako.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

Gero, aipatutako pinak irteera legez (*output*) funtzionatuko dutela zehaztu da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) LED gorria piztu (HIGH), eta beste LED biak eta txirrina amatatu (LOW).
2) Itxaron (`delay`) 4000 ms = 4 s.
3) LED horia piztu eta beste LED biak amatatu.
4) Itxaron 1000 ms = 1 s.
5) LED berdea eta txirrina piztu, eta beste LED biak amatatu.
6) Itxaron 4000 ms = 4 s.

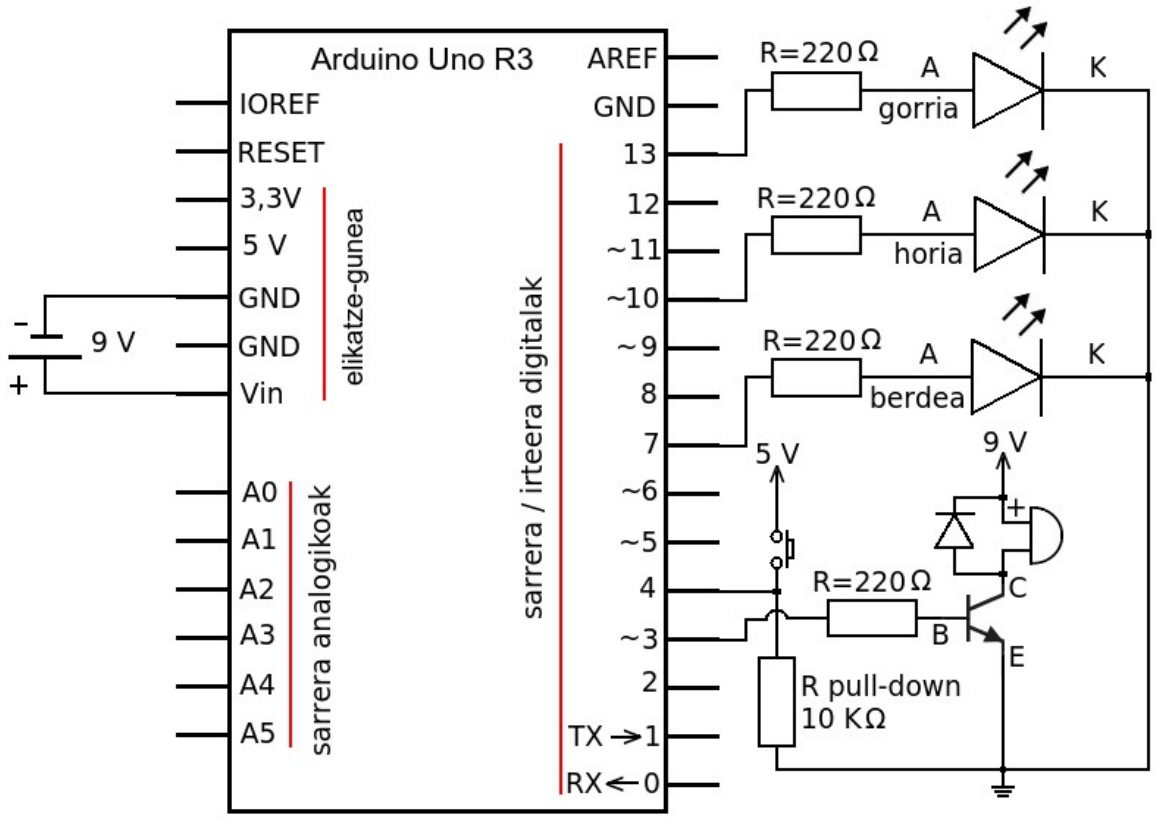
Oharra: Programa bitarraren tamaina 1.236 byte da.

•

3. PROIEKTUA: ITSUENTZAT EGOKITUTAKO PULTSADOREDUN SEMAFOROA

Itsuentzat egokitutako semaforoan, argi gorria piztuta egongo da pultsadorea sakatu arte; pultsadorea sakatu ondoren, argi berdea piztuko da, eta txirrinaren burrunba entzungo da denbora batez. Azkenik, argi gorria piztuko da berriro.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* 220 Ω-eko 4 erresistentzia
* 10 KΩ-eko erresistentzia
* 3 LED (gorria, horia eta berdea)
* BD 135 transistorea edo baliokidea
* Txirrina
* Diodoa
* Pultsadorea
* 9 V-ko pila

R PULL DOWN ERRESISTENTZIA

Azpimarratu behar da R pull-down erresistentzia jarri behar dela honako proiektu honetan. Erresistentzia horren zeregina hauxe da: pultsadorea sakatu barik dagoenean, 4 zenbakiko hankan **0** (LOW) maila izatea.

Programa

pultsadoredun_semaforoa.ino

```
/* Programa hau itsuentzat egokitutako
pultsadoredun semaforoa egiteko da  */

int ledgorria = 13;
int ledhoria = 10;
int ledberdea = 7;
int txirrina = 3;
int pultsadorea = 4;

void setup(){

  pinMode(ledgorria, OUTPUT);
  pinMode(ledhoria, OUTPUT);
  pinMode(ledberdea, OUTPUT);
  pinMode(txirrina, OUTPUT);
  pinMode(pultsadorea, INPUT);


}

void loop(){
  digitalWrite(ledgorria, HIGH);
  digitalWrite(ledhoria, LOW);
  digitalWrite(ledberdea, LOW);
  digitalWrite(txirrina, LOW);

  if (digitalRead(pultsadorea) == HIGH) {


  digitalWrite(ledberdea, HIGH);
  digitalWrite(ledgorria, LOW);
  digitalWrite(txirrina, HIGH);
delay(4000);


}
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaien izenak erabaki dira: `ledgorria`, `ledhoria`, `ledberdea`, `txirrina` eta `pultsadorea`. LED gorria 13 zenbakiko pinean muntatu beharko da, 10ekoan LED horia, 7koan LED berdea, 3koan txirrina eta 4koan pultsadorea.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

Gero, pinak irteera legez (*output*) funtzionatuko dutela zehaztu da; pultsadoreak, ostera, sarrera legez (*input*) funtzionatuko du.

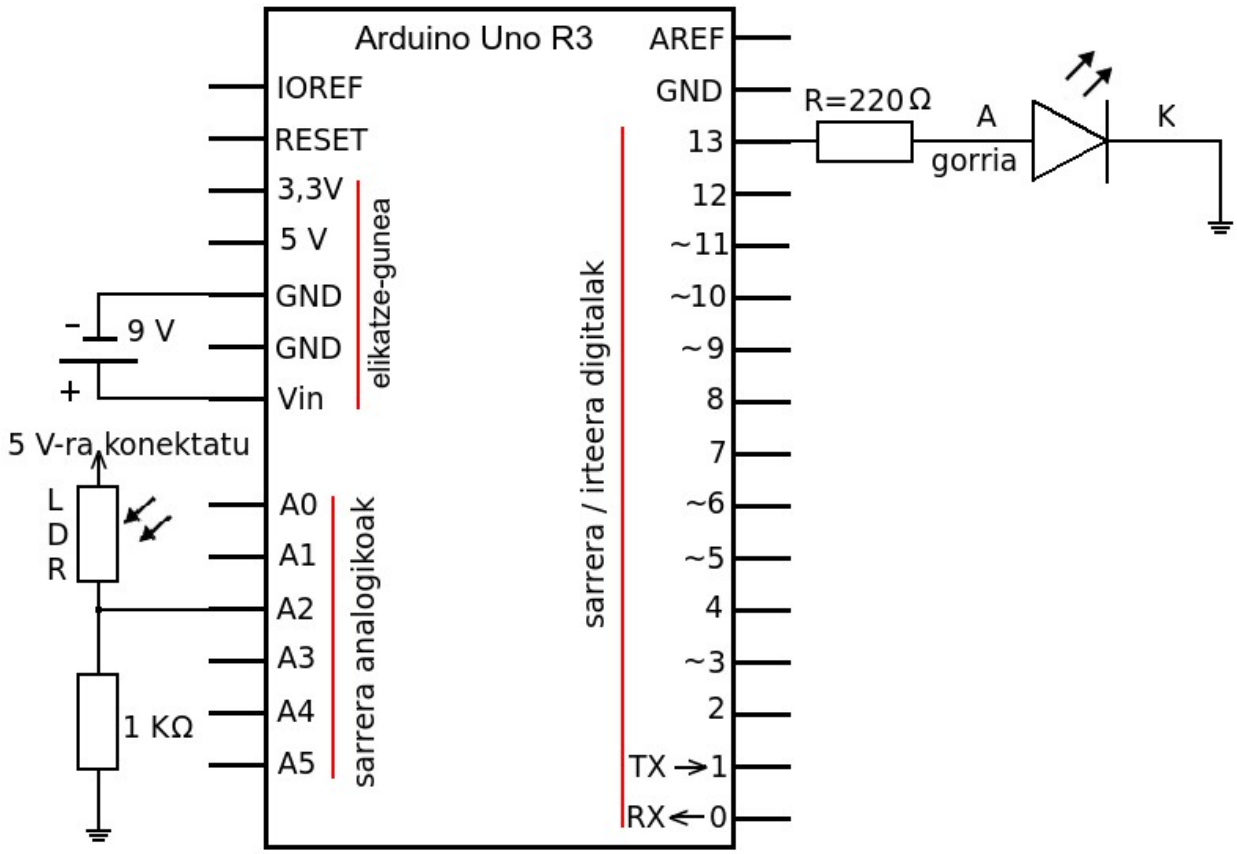
Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:
1) LED gorria piztu (HIGH) eta beste LED biak eta txirrina amatatu (LOW).
2) Pultsadorea sakatuta badago, LED gorria amatatu egingo da, LED berdea eta txirrina piztu, eta 4000 ms = 4 s-ko etena egingo du; ondoren, begizta errepikatuko du, hau da, lehenengo urratsera joango da. Baina pultsadorea sakatuta ez badago, lehenengo urratsera joango da zuzenean.

Oharra: Programa bitarraren tamaina 1.240 byte da.

4. PROIEKTUA: ARGITASUN ESKASA DAGOENEAN LED BAT PIZTEA

Proiektu honetan, LED gorri bat piztuko da argitasun maila balio jakin batetik behera dagoenean.

Zirkuitua



Osagaiak

Honako osagai elektroniko hauek behar dira:

* Arduino plaka
* 220 Ω-eko erresistentzia
* 1 KΩ-eko erresistentzia
* LED gorria
* LDRa
* 9 V-ko pila

Programa

gauean_piztu.ino

```
/* Programa honek argitasun eskasa
dagoenean LED bat pizten du   */

int sents_hank = 2;
int led_hank = 13;
int sents_bal = 0;



void setup(){

  pinMode(led_hank, OUTPUT);

}

void loop(){
sents_bal = analogRead(sents_hank);
if (sents_bal >200){
  digitalWrite(led_hank, LOW);}
  else{
    digitalWrite(led_hank, HIGH);}
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaien izenak erabaki dira: `sents_hank`, `led_hank`, eta `sents_bal`. LDRa 2 zenbakiko pin analogikoan eta LED gorria 13ko digitalean muntatu beharko dira, aldagaien balioak horiek izatea erabaki delako. Bestalde, `sents_bal` aldagaiaren hasierako balioa 0 izango da.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

13 zenbakiko pina irteera izango da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) 2 zenbakiko pin analogikoan dagoen tentsioaren lagina hartu ondoren, ADC zirkuituak balio bitar bihurtuko du, eta horrela lortutako balioa `sents_bal` deituriko aldagaian gordeko da.

2) Lorturiko balioa 200 baino handiagoa bada, LED diodoa amatatuko du; bestela LED diodoa piztuko du.

Oharra: Programa bitarraren tamaina 956 byte da..

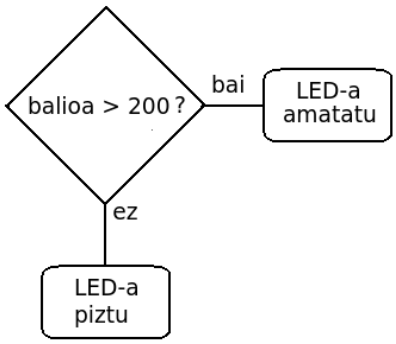
LDR-AREN FUNTZIONAMENDUA

- Egun argiz LDRaren barneko erresistentzia txikia izango da; beraz, 2 zenbakiko pinean dagoen tentsioa handia izango da, eta `sents_bal` ere bai. Horrela bada, LED diodoa amatatu egingo da.

- LDRa atzamarraz ondo tapatzen bada, bere barneko erresistentzia handia izango da; beraz, 2 zenbakiko pinean dagoen tentsioa txikia izango da, eta `sents_bal` ere bai. Ondorioz, LED diodoa piztuko da.

Programan galdera (baldintza) horrela egin da:

```
if (sents_bal >200){
  digitalWrite(led_hank, LOW);}
  else{
    digitalWrite(led_hank, HIGH);}
```

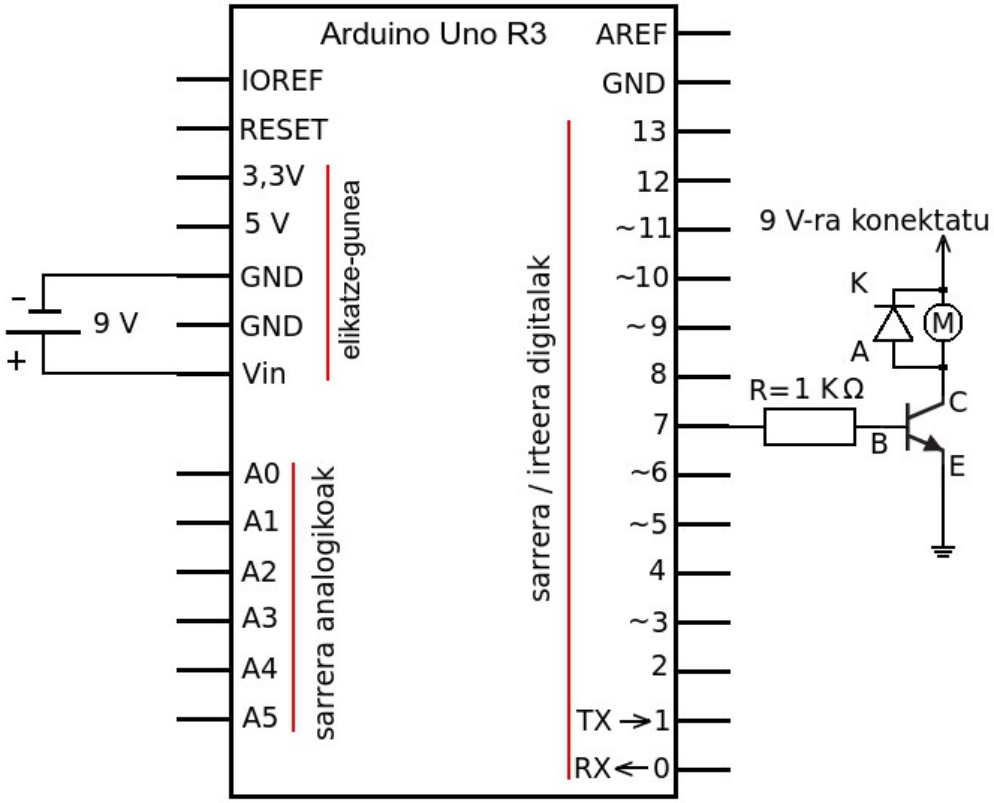


Hau da, `sents_bal` aldagaiaren balioa 200 baino handiagoa bada, LEDa amatatu dadila; bestela, piztu dadila.

5. PROIEKTUA: KORRONTE ZUZENEKO MOTORRA KONTROLATZEA

Proiektu honetan, korronte zuzeneko motor txiki bat aldizka piztu eta amatatu egiten da.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* Diodoa
* 1 KΩ-eko erresistentzia
* Korronte zuzeneko motor txikia
* BD 135 transistorea edo baliokidea
* 9 V-ko pila

Lana osatzeko aukera

Lantalde batek motor elektriko bidez mugitzen den uhala muntatzea, eta beste lan-talde batek proposatutako proiektua gauzatzea.

Programa

motorra_piztu.ino

```
/* Programa honek korronte zuzeneko
   motorra kontrolatzeko balio du      */

int motor_pina = 7;




void setup() {

  pinMode(motor_pina, OUTPUT);
}
void loop() {
  digitalWrite(motor_pina, HIGH);
  delay(100);
  digitalWrite(motor_pina, LOW);
  delay(100);
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaiaren izena erabaki da: `motor_pina`, eta 7 zenbakiko pinean muntatu beharko da.

Oharra: ez da zertan derrigorrean pin hori erabili; nahi izanez gero, beste bat erabili daiteke.

7 zenbakiko pina irteera izango da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) motorra piztu
2) 100 ms itxaron
3) motorra amatatu
4) 100 ms itxaron

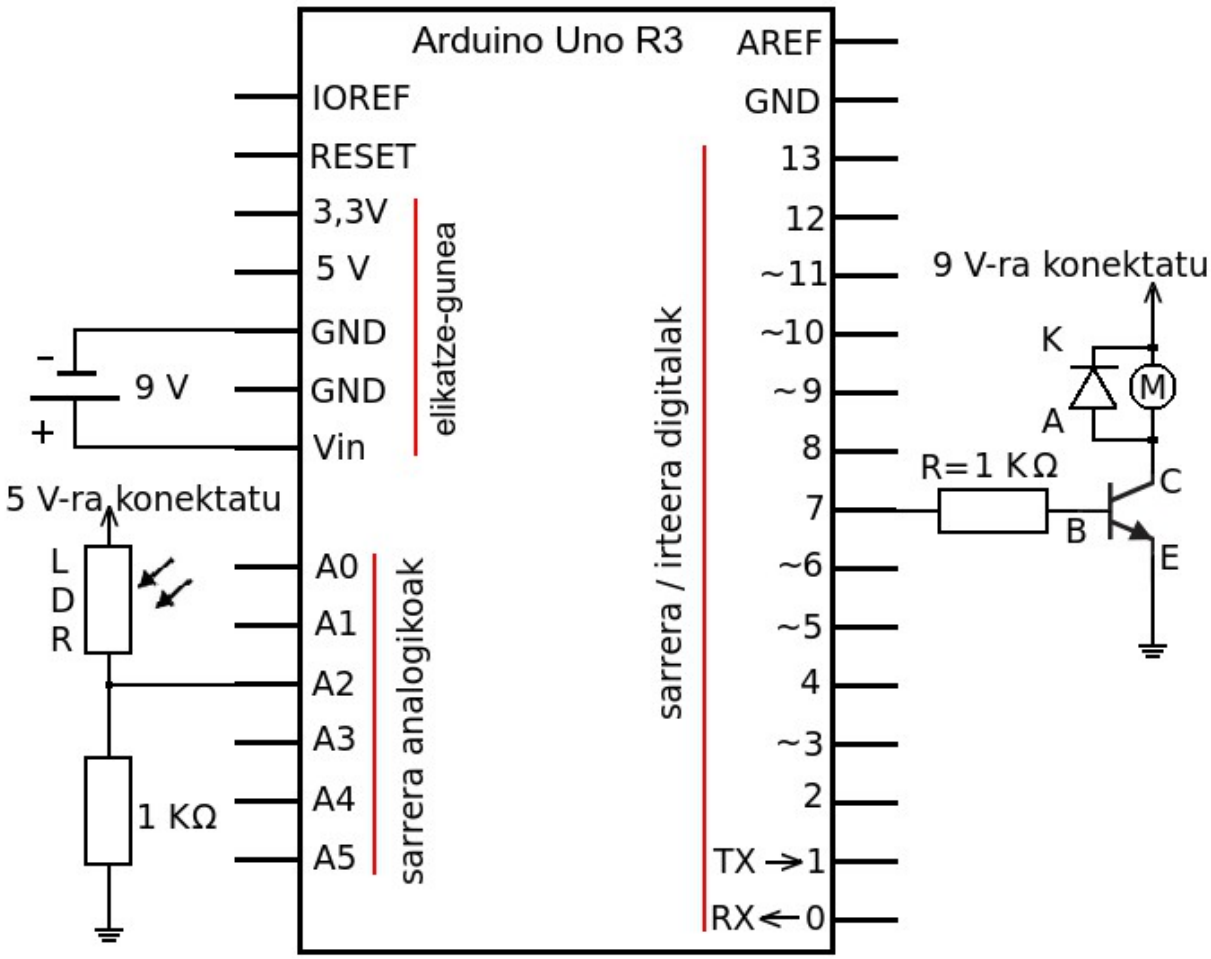
Oharra: Programa bitarraren tamaina 1.056 byte da.

6. PROIEKTUA: ARGITASUN ESKASA DAGOENEAN MOTOR BAT

PIZTEA

Proiektu honetan, korronte zuzeneko motor txiki bat piztuko da argitasun maila balio jakin batetik behera dagoenean.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* Diodoa
* 1 KΩ-eko erresistentzia 2
* Korronte zuzeneko motor txikia
* BD 135 transistorea edo baliokidea
* LDRa
* 9 V-ko pila

Programa

motorra_piztu2.ino

```
/* Programa honek argitasun eskasa
dagoenean motor  bat pizteko balio du  */

int sentsore_hank = 2;
int motor_pina = 7;
int sentsore_bal = 0;




void setup(){

  pinMode(motor_pina, OUTPUT);

}

void loop(){
sentsore_bal = analogRead(sentsore_hank);
if (sentsore_bal >200){
  digitalWrite(motor_pina, LOW);}
  else{
    digitalWrite(motor_pina, HIGH);}
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaien izenak erabaki dira: `sentsore_pina`, `motor_pina`, eta `sentsore_bal`. LDRa 2 zenbakiko pin analogikoan eta motor txikia 7ko digitalean muntatu beharko dira, aldagaien balioak horiek izatea erabaki delako. Bestalde, `sentsore_balioa` aldagaiaren hasierako balioa 0 izango da.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

7 zenbakiko pina irteera izango da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) 2 zenbakiko pin analogikoan dagoen tentsioaren lagina hartu ondoren, ADC zirkuituak balio bitar bihurtuko du tentsio hori; ostean, lortutako balioa `sentsore_balioa` deituriko aldagaian gordeko da.

2) Lorturiko balioa 200 baino handiagoa bada, motorra amatatuko da; bestela, motorra piztu egingo da.

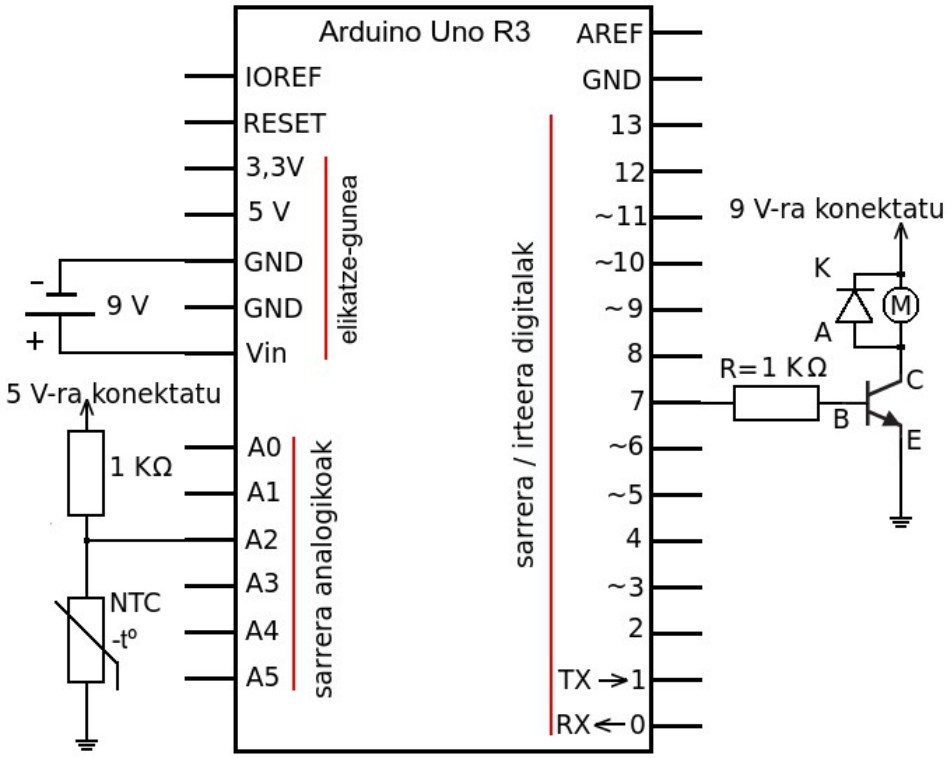
Oharra: Programa bitarraren tamaina 956 byte da..

7. PROIEKTUA: BERO DAGOENEAN HAIZAGAILUAREN MOTORRA PIZTEA

Tenperatura balio jakin batetik gora dagoenean, korronte zuzeneko motor txiki bat piztuko da proiektuan.

Proiektu hau egiten denean, arazoa egoten da ikasgela edo tailerreko tenperatura nahiko konstantea delako. Hori dela eta, zuzenean berotu behar da NTC erresistentzia. Hain zuzen ere, NTCa berotu egingo da motor txikia piztu arte; gero, laga egingo zaio berotzeari, eta, denbora bat igaro ondoren, motorra berez amatatuko da.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* Diodoa
* 1 KΩ-eko erresistentzia 2
* korronte zuzeneko motor txikia
* BD 135 transistorea edo baliokidea
* NTCa
* 9 V-ko pila

Programa

haizagailua.ino

```
/* Programa honek bero dagoenean
haizagailu  bat pizteko balio du    */

int sents_hank = 2;
int motor_pina = 7;
int sentsore_balioa = 0;




void setup(){

  pinMode(motor_pina, OUTPUT);

}

void loop(){
sentsore_balioa = analogRead(sents_hank);
if (sentsore_balioa >200){
  digitalWrite(motor_pina, LOW);}
  else{
    digitalWrite(motor_pina, HIGH);}
}
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaien izenak erabaki dira: `sents_hank`, `motor_pina`, eta `sentsore_balioa`. NTCa 2 zenbakiko pin analogikoan eta motor txikia 7ko digitalean muntatu beharko dira, aldagaien balioak horiek izatea erabaki delako. Bestalde, `sentsore_balioa` aldagaiaren hasierako balioa 0 izango da.

Oharra: ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

7 zenbakiko pina irteera izango da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) 2 zenbakiko pin analogikoan dagoen tentsioaren lagina hartu ondoren, ADC zirkuituak balio bitar bihurtuko du tentsio hori; ostean, lortutako balioa `sentsore_balioa` deituriko aldagaian gordeko da.

2) Lorturiko balioa 200 baino handiagoa bada, motorra amatatuko da; bestela, motorra piztuko da.

Oharra: Programa bitarraren tamainua 956 byte da..

NTC-ARI BURUZKO AZALPENA

- Hotz dagoenean, NTCaren barne erresistentzia handia izaten da; beraz, 2 zenbakiko pineko tentsioa ere bai. Ondorioz, `sentsore_balioa` aldagaiaren balioa ere handia izango da, eta, balio hori 200 baino handiagoa izango denez, haizagailuaren motorra amatatu egingo da.

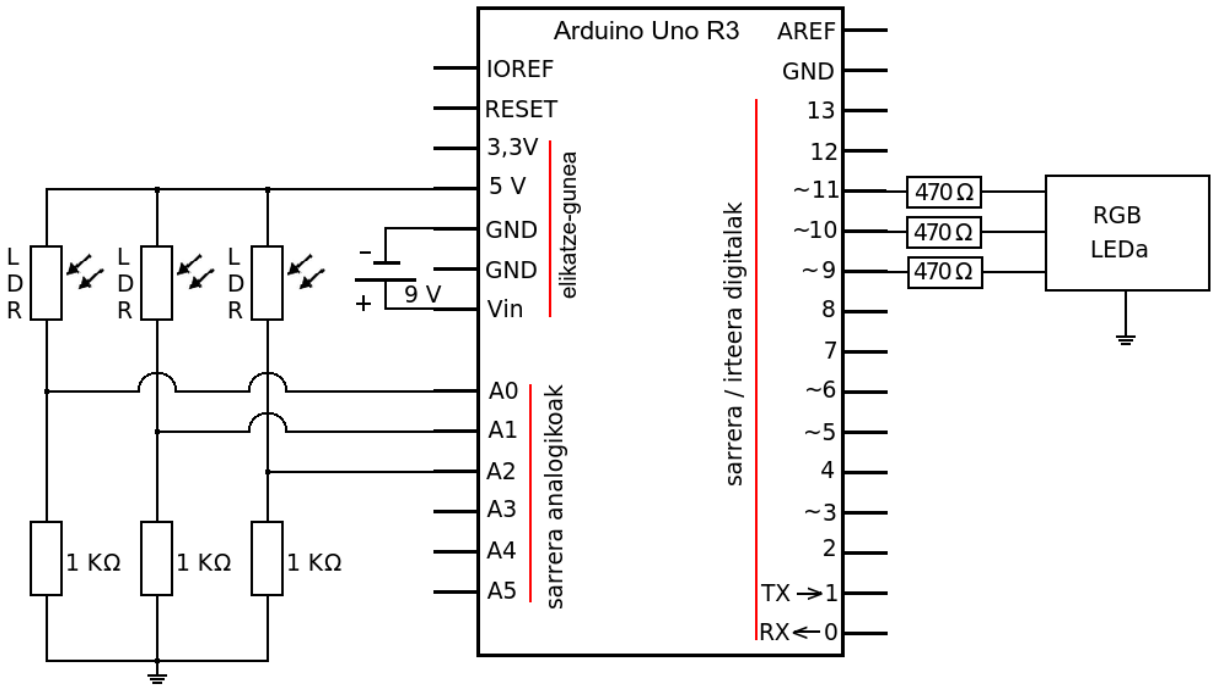
- NTCa berotzen bada, bere barne-erresistentzia txikitu egingo da; beraz, 2 zenbakiko pineko tentsioa ere bai. Ondorioz, `sentsore_balioa` aldagaiaren balioa ere txikitu egingo da, eta, balio hori 200 baino txikiagoa izatera helduko denez, haizagailuaren motorra piztuko da.

- NTCa berotzeari laga ondoren, alderantzizkoa gertatuko da: NTCaren erresistentzia eta V_{NTC} handitu egingo dira, eta, beraz, `sentsore_balioa` 200 baino handiagoa izango denez, haizagailuaren motorra amatatu egingo da.

8. PROIEKTUA: RGB MOTAKO LED BATEN ARGIAK NAHASTEA

Proiektu honetan, hiru LDR jarri dira. LDR bakarra atzamarraz tapatutakoan, RGB motako LEDak kolore bakarreko argia igorriko du. LDR bi tapatuz gero, kolore biren nahasketa igorriko du, eta, hiru LDRak tapatuz gero, hiru koloreen nahasketa igorriko du.

Zirkuitua



Osagaiak

Hona hemen erabili beharreko osagaiak:

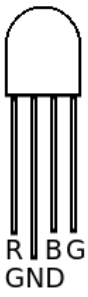
* Arduino plaka
* 470 Ω-eko hiru erresistentzia
* 1 KΩ-eko hiru erresistentzia
* Hiru LDR
* RGB motako LEDa
* 9 V-ko pila

RGB motako LEDa

Osagai elektroniko honek 4 pin ditu: LED gorriaren anodoa (R), LED urdinaren anodoa (B), LED berdearen anodoa (G) eta GND; azken hori luzeagoa da.

Katodo guztiak GND pinera elkartuta daude.

Kontuan izan behar da hiru erresistentzia jarri behar direla osagai honen barneko LED diodoak ez hondatzeko.



Programa

rgb_argiak_nahastea.ino

```
/* Programa honek RGB motako LED baten
argiak nahasten  ditu                    */

int sents_hank_R = 0;
int sents_hank_G = 1;
int sents_pina_B = 2;
int led_R=11;
int led_G=9;
int led_B=10;
int sents_bal_R = 0;
int sents_bal_G = 0;
int sents_bal_B = 0;

void setup(){
  pinMode(led_R, OUTPUT);
  pinMode(led_G, OUTPUT);
  pinMode(led_B, OUTPUT);
  }

void loop(){

sents_bal_R = analogRead(sents_hank_R);
if (sents_bal_R >200){
  digitalWrite(led_R, LOW);}
  else{
    digitalWrite(led_R, HIGH);}
delay(100);

sents_bal_G= analogRead(sents_hank_G);
if (sents_bal_G >200){
  digitalWrite(led_G, LOW);}
  else{
    digitalWrite(led_G, HIGH);}
delay(100);
```

Programaren azalpena

Programaren iruzkina.

Programa hau errazago ulertzeko, 4. proiektuaren programaren azalpena irakurtzea komeni da, zirkuituaren funtzionamendu berdina du eta.

Osagai batzuk hiru aldiz jarri dira, hala nola LDRak eta erresistentziak.
Hiru LED independente jarri beharrean, RGB motako LEDa jarri da, baditu-eta hiru LED barnean.
Sentsoreak eta LEDak bereizteko, R, G eta B hizkiak erabili dira, hau da, gorria, berdea eta urdina.

Oharra:
Programa honetan, pin analogiko eta digital jakin batzuk erabili dira; hala ere, ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

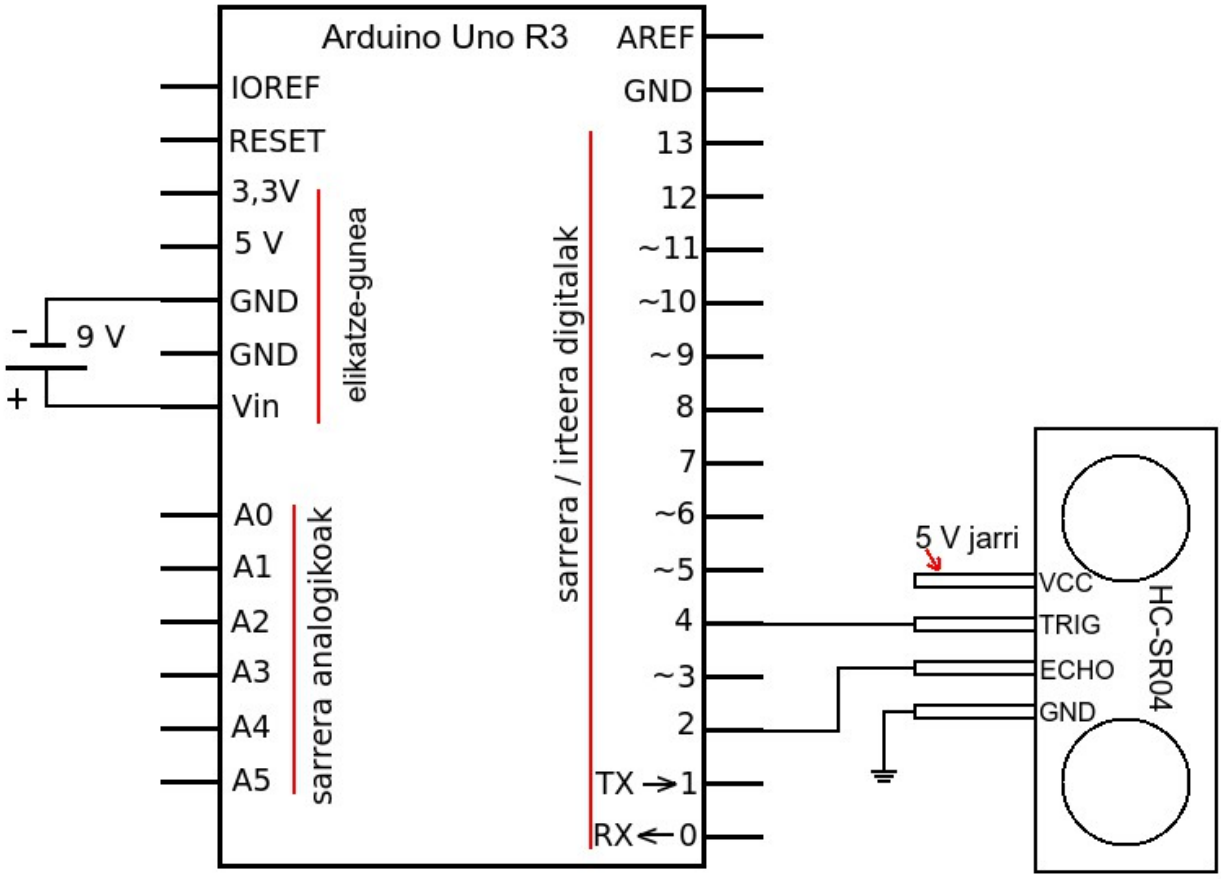
Oharra: Programa bitarraren tamaina 1.236 byte da.

```
sents_bal_B = analogRead(sents_hank_B);
if (sents_bal_B >200){
  digitalWrite(led_B, LOW);}
  else{
    digitalWrite(led_B, HIGH);}
delay(100);
}
```

9. PROIEKTUA: DISTANTZIAK NEURTZEA

Distantziak neurtzeko balio du proiektu honek, eta, horretarako, ultrasoinu-sentsore bat erabiliko da. Emaitza konputagailuaren pantailan bistaratuko da.

Zirkuitua

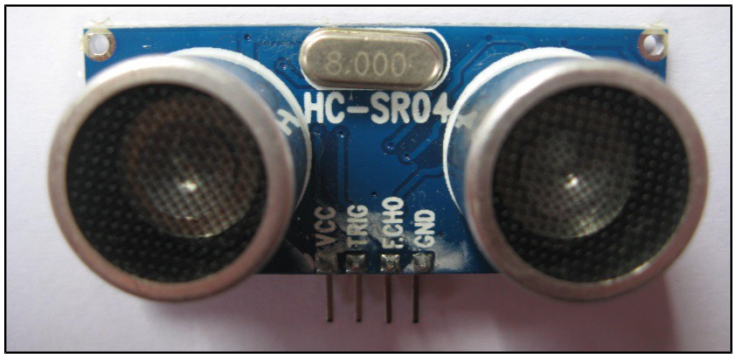


Osagaiak

Proiektu honetan osagai hauek behar dira:

* Arduino plaka
* HC-SR04 ultrasoinu-sentsorea
* 9 V-ko pila

HC-SR04 ULTRASOINU SENTSOREA



HC-SR04 ultrasoinu-sentsoreak lau pin ditu: VCC, TRIGGER (desarra-seinalea), ECHO (oihartzuna) eta GND. Hala ere, hiru pindun ultrasoinu-sentsoreak ere erabili daitezke, eta, beraz, desarra-seinalea eta oihartzuna pin beretik kontrolatzen dira. Kontuan izan pin kopuruaren arabera programa aldatu egiten dela. Hiru pindun sentsorea kontrolatzen duen programa lortzeko, joan Arduino IDE programara, eta jo atal hauetara, hurrenez hurren: *File*, *Examples*, *Sensors* eta *Ping*.

Objektua zenbat zentimetrotara dagoen jakiteko formula hau erabiltzen da:

```
distantzia = iraupena / 29 / 2;
```

Kontuan hartu behar da bidalitako uhina zerbait jo ondoren itzuli egiten dela, eta, hori dela eta, sentsorearen eta objektuaren arteko distantzia jakiteko, **zati 2** egin behar dela.
Bestalde, kontuan hartu behar da soinuaren gutxi gorabeherako abiadura 340 m/s dela.

34.000 cm ————1.000.000 µs
1 cm ————X

X = (1.000.000/34.000) = 29,41 µs

Beraz, 1 cm egiteko, gutxi gorabehera 29 µs behar dituela joko da.

Programa

ultrasoinua.ino

```
/* Programa honek distantziak neurtzeko
balio du          */

int oih_han = 2;
int desarra_han = 4;


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


void loop(){
  long iraupen, distantzia;
  pinMode(desarra_han, OUTPUT);
  digitalWrite(desarra_han, LOW);
  delayMicroseconds(2);
  digitalWrite(desarra_han, HIGH);
```

Programaren azalpena

Programaren iruzkina.

Lehenengo, aldagaien izenak erabaki dira: desarra-seinalea 4 zenbakiko pin digitalean konektatuko da, eta oihartzun seinalea, 2koan.

> Oharra: programa honetan, pin digital jakin batzuk erabili dira, baina ez da zertan derrigorrean pin horiek erabili; nahi izanez gero, beste batzuk erabili daitezke.

Arduino plakaren eta konputagailuaren arteko serieko datu transferentzia 9.600 bit segundotan gauzatuko da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:
Beste aldagai bi, `iraupen` eta `distantzia`, `long` motakoak izango dira.
4 zenbakiko pina irteera izango dela erabaki da.
Desarra-seinalea 4 zenbakiko pinetik ultrasoinu-sentsorera

```
  delayMicroseconds(10);
  digitalWrite(desarra_han, LOW);

  pinMode(oih_han, INPUT);
  iraupen = pulseIn(oih_han, HIGH);
  distantzia = iraupen / 29 / 2;

  Serial.print(distantzia);
  Serial.print(" cm-ra dago");
  Serial.println();
  delay(1000);
}
```

bidaliko da. Pultsuaren LOW egoerak 2 µs iraungo du, eta HIGH egoerak, 10 µs. Gero, LOW egoeran jarriko da berriro.

2 zenbakiko pina sarrera izango dela erabaki da.
Bidalitako pultsuaren eta oihartzun seinalearen arteko iraupena neurtzeko, `pulseIn` funtzioa erabili da.
Distantzia formula bidez kalkulatu da.

Emaitza bistaratzeko, `distantzia` aldagaia sartu da `Serial.print` funtzioan; gero, `" cm-ra dago"` esaldi zatia, emaitza zer unitate den zehazteko; ondoren, kurtsorea hurrengo lerroaren hasieran kokatzeko `Serial.println;` funtzioa txertatu da, eta, azkenik, azaltzen dena irakurtzeko astia izateko, 1 segundo itxarongo dela ezarri da.

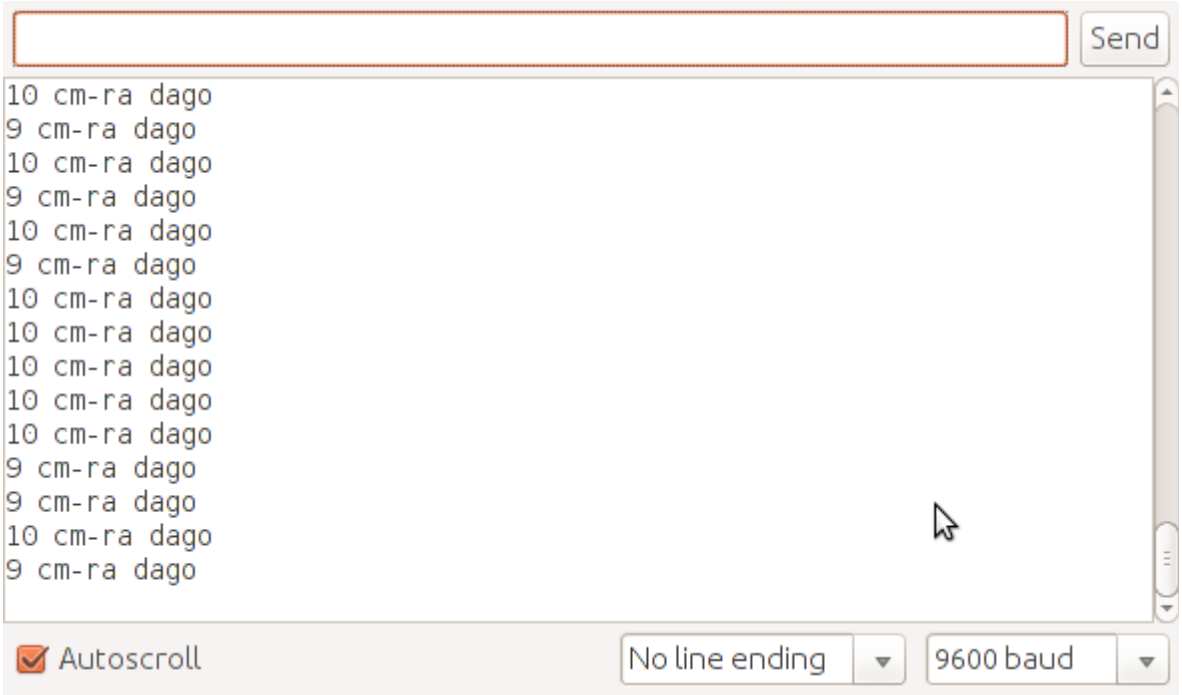
Oharra: Programa bitarraren tamaina 3.210 byte da..

DISTANTZIEN BALIOAK KONPUTAGAILUAN BISTARATZEA

Proiektu honetan neurtutako distantzien balioak konputagailuaren pantailan bistaratuko dira.

1. Horretarako, sakatu *Serial Monitor* ikurra.

2. Beste leiho bat irekiko da, hain zuzen ere, neurtutako distantzien balioak bistaratuko dituena. Distantziak neurtzeko, jarri objektu bat sentsorearen aurrean, eta objektu hori segundoro zenbat zentimetrora dagoen bistaratuko da konputagailuaren pantailan. Objektu hori mugitzen bada, bistaratzen diren balioak aldatu egingo dira.

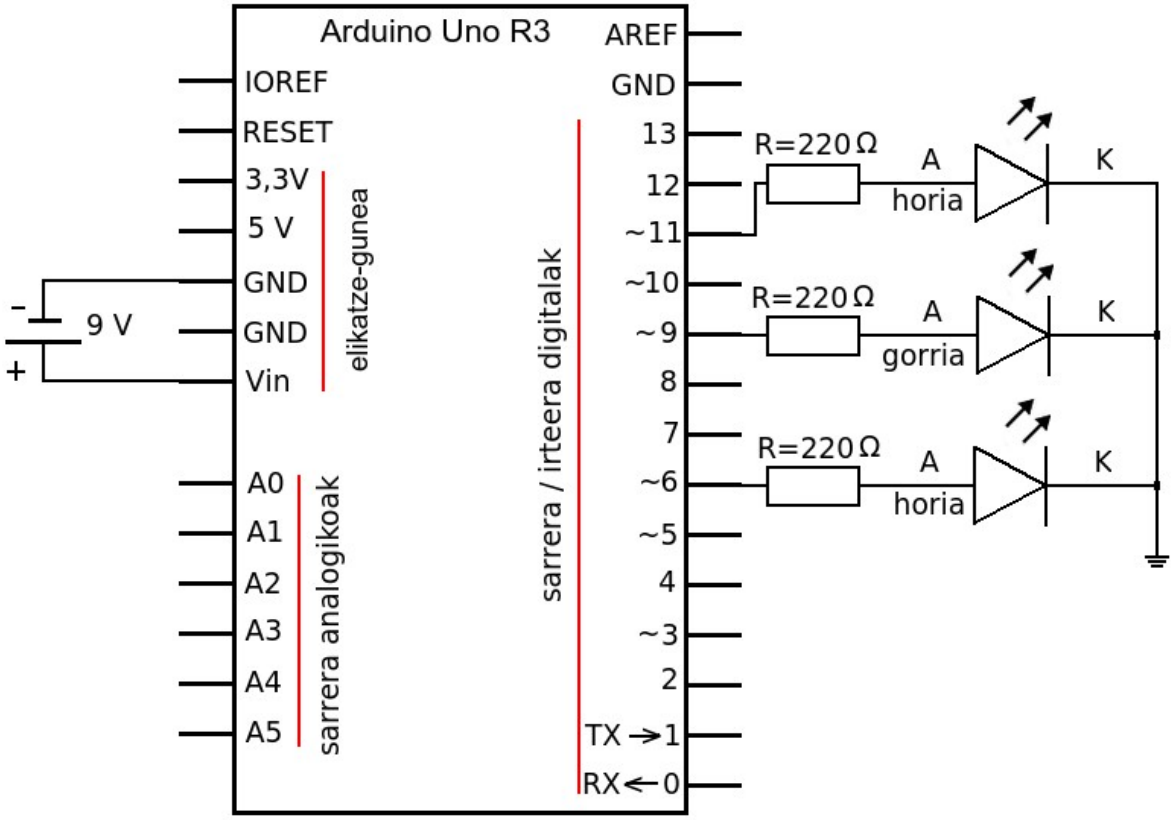


Distantzia zehatzago neurtu nahi izanez gero, sentsorearen inguruko tenperatura neurtu behar da, uhinen abiadura tenperaturen arabera aldatzen da eta. Jakina, distantzia hamartarrekin adierazi nahi bada, aldagaiak `float` motakoak izan behar dira.

10.PROIEKTUA: LED DIODOEN DISTIRA ALDATZEA

Proiektu honetan, lan-ziklo aldakorreko uhinak bidaltzen dira LED diodoetara, haien distira aldatzeko asmoz.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* 220 Ω-eko hiru erresistentzia
* LED hori bi
* LED gorri bat
* 9 V-ko pila

Lan proposamen batzuk

1- Beheko sua simulatzea.

2- Ibilgailuen gaineko argiak simulatzea.

Programa

distira.ino

```
/* Programa honek LED diodoen distira
aldatzen du                          */

int led_horia1 = 11;
int led_gorria = 9;
int led_horia2 = 6;

void setup()
{
pinMode(led_horia1, OUTPUT);
pinMode(led_gorria, OUTPUT);
pinMode(led_horia2, OUTPUT);
}

void loop()
{
analogWrite(led_horia1, random(256));
analogWrite(led_gorria, random(256));
analogWrite(led_horia2, random(256));

delay(random(100));
}
```

Programaren azalpena

Programaren iruzkina.

LED diodoak zein pinetan konektatuta dauden zehaztu da. 11, 9 eta 6 pin digitalak erabiltzea erabaki da. Kontuan hartu PWM ezaugarria daukaten pin digitalak bakarrik erabili daitezkeela.

Aipatutako pinek irteera legez funtzionatuko dute.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

`Random()` funtzioa erabiliz, lan-ziklo aldakorreko uhina bidaliko da hiru pin digital horietara.
`Random()` funtzioak pseudoausazko zenbakiak ematen ditu. Ematen duen zenbaki txikiena 0 da, eta handiena parentesi artean dagoen zenbakia baino bat txikiagoa izaten da.
Adibidez, `random(256)` funtzioak 0tik 255erako zenbakiak ematen ditu.
Jakina, zenbat eta zenbaki handiagoa izan, orduan eta distiratsuago egongo da LEDa.

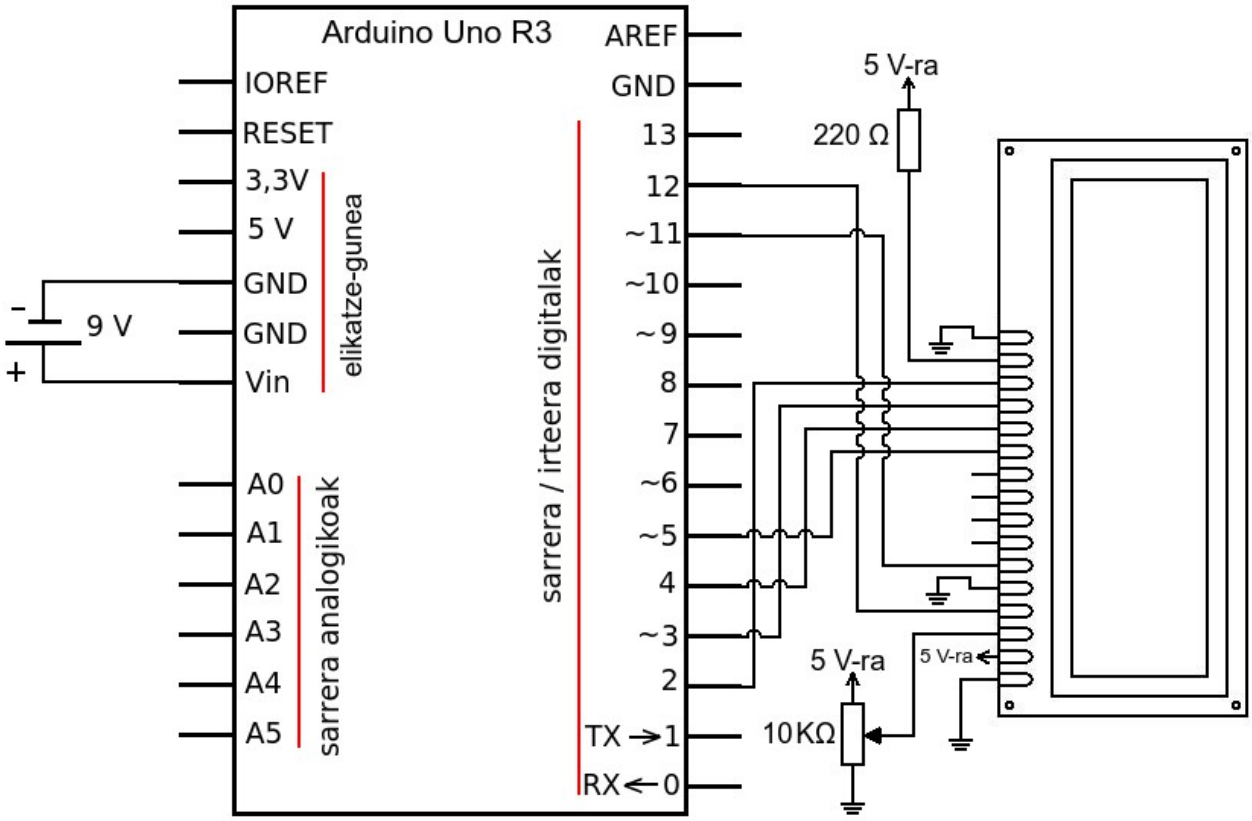
Itxarote denbora aldakorra egingo da, `random()` funtzioaren menpe dago eta.

Oharra: Programa bitarraren tamaina 1.754 byte da.

11. PROIEKTUA: TESTUAK BISTARATZEA LCD PANTAILAN

LCD pantailan autobusen ordutegiak bistaratzea da proiektu honen helburua.

Zirkuitua



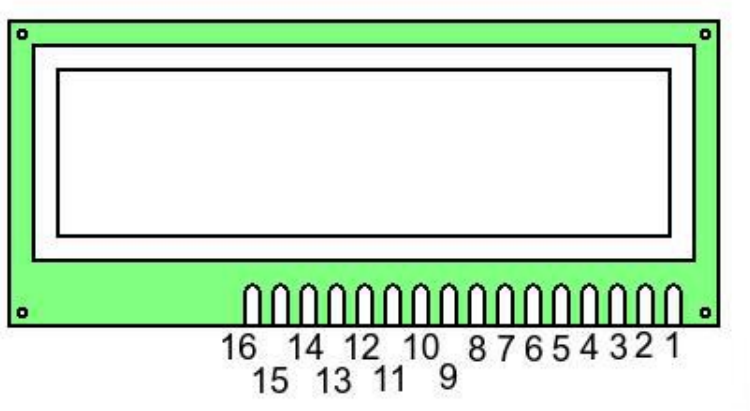
Osagaiak

Proiektu honetan erabili beharreko osagai elektronikoak honako hauek dira:

* Arduino plaka
* LCD pantaila
* 220 Ω-eko erresistentzia
* 10 KΩ-eko potentziometroa
* 9 V-ko pila

LCD PANTAILA

Erabiltzen den LCD pantailak Hitachi HD44780 zirkuitu integratu kontrolatzailearekin bateragarria izan behar du.

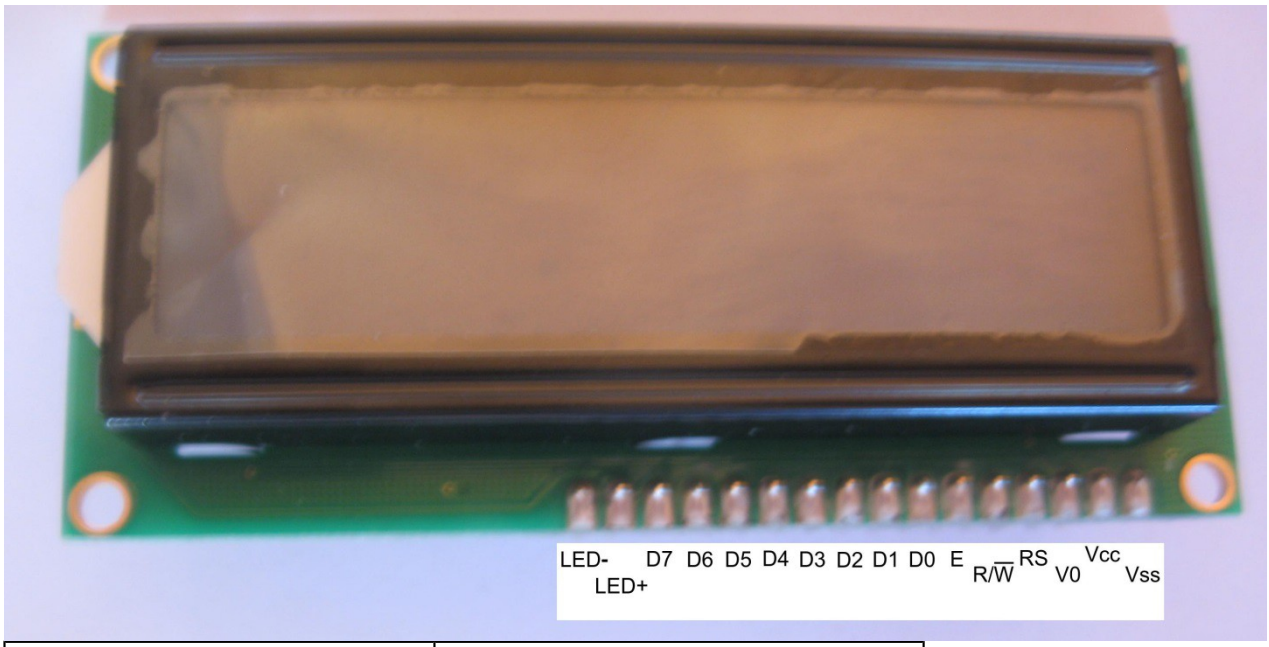


LCD PANTAILA KONTROLATZEKO FUNTZIOAK

LCD pantaila kontrolatzeko, hainbat funtzio daude. Informazio gehiago behar izanez gero, joan Arduino IDEra, aukeratu *Libraries;* ondoren, aukeratu `LiquidCrystal` eta aukeratu komeni den funtzioa irekitako orrian.

LCD pantailara datu baten 8 bitak batera edo bitan zatituta bidali daitezke. Proiektu honetan, 4 bit batera bidaltzea erabaki da, horrela, Arduino plakaren 4 pin digital gutxiago erabiliko dira eta.

LCD-AREN PINAK



LCDaren pinak	Arduinoren zein pinetara
1 Vss	GND
2 Vcc	5 V
3 V0 (kontrastea)	
4 RS (register select)	12
5 R/$\overline{W}$ (read/$\overline{write}$)	GND
6 E (enable)	11
7 D0	
8 D1	
9 D2	
10 D3	
11 D4	5
12 D5	4
13 D6	3
14 D7	2
15 LED+	
16 LED-	GND

Programa

lcd.ino

```
/* Programa honek autobusen
    ordutegia bistaratzen du   */

#include <LiquidCrystal.h>



LiquidCrystal lcd(12, 11, 5, 4, 3, 2);




void setup() {

  lcd.begin(16, 2);
}

void loop() {
  lcd.setCursor(0,0);
  lcd.print("Bilbo--Alts--Iru");
  lcd.setCursor(0,1);
  lcd.print("0700--0845--0920");
  delay(4000);
  lcd.setCursor(0,0);
  lcd.print("Bilbo--Don--Lour");
  lcd.setCursor(0,1);
  lcd.print("0730--0840--1200");
  delay(4000);
  lcd.setCursor(0,0);
  lcd.print("Bilbo--Gas--Zara");
  lcd.setCursor(0,1);
  lcd.print("1445--1545--1845");
  delay(4000);

}
```

Programaren azalpena

Programaren iruzkina.

Programa-liburutegi bat (*library*) gehitu zaio idatzitako programari; horri esker, LCD pantaila oso erraz kontrolatuko da funtzio berezi batzuk erabilita.

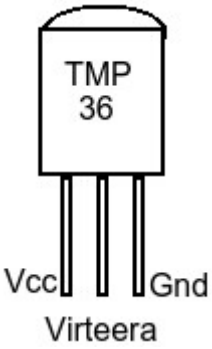
Zehaztu da Arduino plakak zein pinetatik bidaliko dituen kontrolerako seinaleak eta datuak: erregistroa aukeratzeko seinalea (RS) 12 zenbakiko pin digitaletik irtengo da, gaitzeko seinalea (*enable*) 11 zenbakiko pinetik eta lau bitak 5, 4, 3 eta 2 pinetatik.

Proiektuan erabiliko den LCD pantailak 16 zutabe eta errenkada bi dituela zehaztu da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

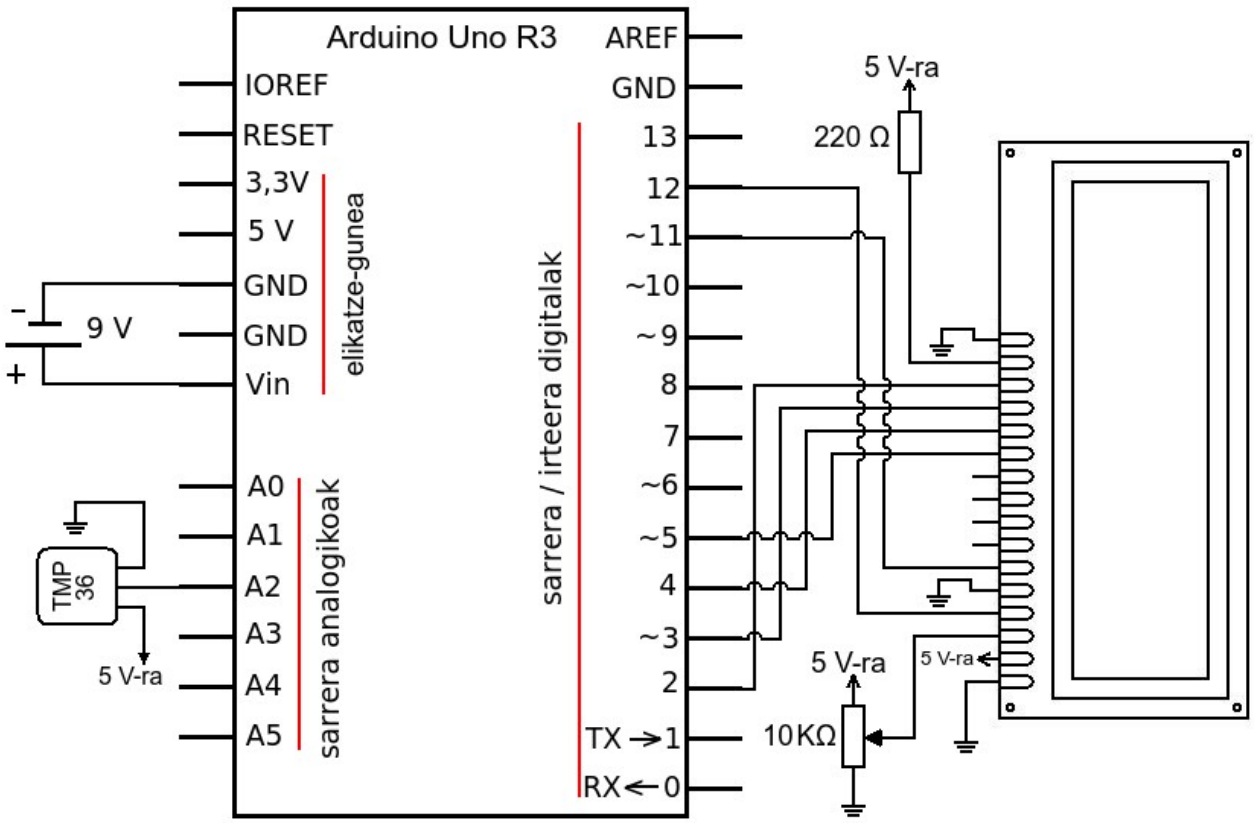
1) Kurtsorea 0 zutabean eta goiko lerroan kokatuko da.
2) Esaldi bat bistaratuko da
3) Kurtsorea 0 zutabean eta beheko lerroan kokatuko da.
4) Esaldi bat bistaratuko da.
5) Lau segundo itxarongo da.
6) Kurtsorea 0 zutabean eta goiko lerroan kokatu da.
7) Esaldi bat bistaratuko da.
8) Kurtsorea 0 zutabean eta beheko lerroan kokatu da.
9) Esaldi bat bistaratuko da.
10) Lau segundo itxarongo da.
11) Kurtsorea 0 zutabean eta goiko lerroan kokatuko da.
12) Esaldi bat bistaratuko da.
13) Kurtsorea 0 zutabean eta beheko lerroan kokatuko da.
14) Esaldi bat bistaratuko da.
15) Lau segundo itxarongo da.

Oharra: Programa bitarraren tamaina 2.434 byte da.



12. PROIEKTUA: GELAKO TENPERATURA LCD-AN ETA KONPUTAGAILUKO PANTAILAN BISTARATZEA

Proiektu honetan, tenperatura-sentsore bat erabiliz, inguruko tenperatura kalkulatu ondoren, LCD pantailan eta konputagailuan bistaratzen da.



Zirkuitua

Osagaiak

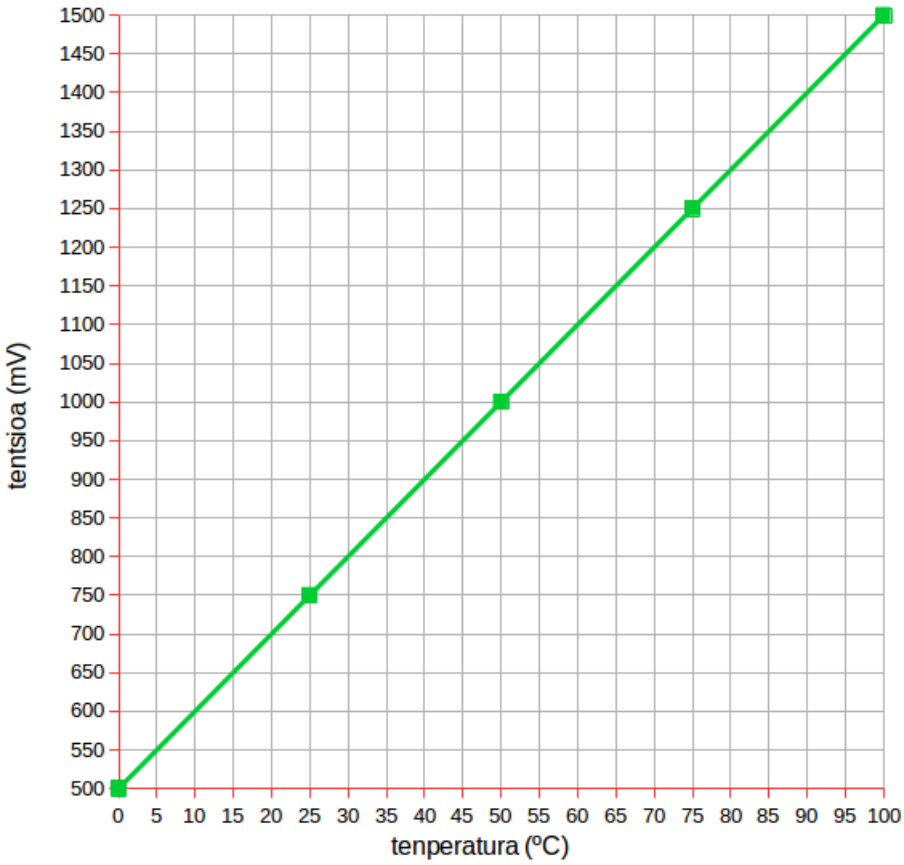
Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* LCDa
* TMP 36GZ tenperatura sentsorea
* 220 Ω-eko erresistentzia
* 10 KΩ-eko potentziometroa
* 9 V-ko pila

TMP 36GZ TENPERATURA SENTSOREA

TMP 36GZ tenperatura-sentsoreak hiru pin ditu: ezkerreko pina Vcc da, erdikoak tenperaturaren araberako tentsio elektrikoa ematen du eta eskuinekoa Gnd da.

TMP 36GZ tenperatura-sentsorea



Grafikoan ikusten denez, TMP 36GZ sentsoreak tenperaturaren araberako tentsio elektrikoak ematen ditu. Adibidez, sentsorea dagoen inguruko tenperatura 25 ºC bada, 750 mV ematen du.

Grafikoa lineala denez, funtzioa horrela adierazten da:

tentsioa = malda · tenperatura + ordenatua jatorrian

tentsioa = 1000/100 · tenperatura + 500

tentsioa = 10 · tenperatura + 500

Beraz, tenperatura = (tentsioa – 500) / 10

Tenperatura sentsoreak ematen duen tentsio elektrikoa Arduino plakaren 2 zenbakiko pin analogikora bidaltzen da. Ondoren, balio digital bihurtzen da, eta balioaren arabera jakingo da zer tentsio elektriko dagoen pinean.

Tentsioa mV-tan adierazteko, honako formula hau erabiltzen da: tentsioa = (balioa · 5000) / 1023

Formula horri buruz gehiago jakin nahi izanez gero, begiratu `analogRead()` funtzioaren azalpena.

Gero, tentsioaren balioa tenperaturaren formulan sartuta, sentsorearen inguruko tenperatura

kalkulatuko da.

Programa

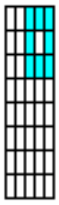
termometroa.ino

```
/*Programa honek tenperatuta neurtzeko
balio du.    */

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);
float tentsioa = 0;
float tenperatura = 0;
float sentsore_bal = 0;
byte gradua[8] = {
              B00111,
              B00101,
              B00111,
              B00000,
              B00000,
              B00000,
              B00000,
              B00000
              };
void setup(){
  Serial.begin(9600);
  lcd.begin(16, 2);
  lcd.createChar(1, gradua);
  }
void loop(){
  sentsore_bal = analogRead(2);
  tentsioa = (sentsore_bal*5000)/1023;
  tenperatura = (tentsioa-500)/10;

  Serial.print("tenperatura: ");
  Serial.print(tenperatura,1);
  Serial.print(char(176));
  Serial.println(" C da");
  Serial.println();
  delay(4000);

  lcd.setCursor(0,0);
  lcd.print("tenperatura");
  lcd.setCursor(0,1);
  lcd.print(tenperatura,1);
  lcd.write(1);
  lcd.print(" C da");
}
```



Programaren azalpena

Programaren iruzkina.

Idatzitako programari programa-biblioteka bat (*library*) gehitu zaio; horrela, LCD pantaila oso erraz kontrolatu da funtzio berezi batzuk erabilita.

Zehaztu da Arduino plakak zein pinetatik bidaliko dituen kontrolerako seinaleak eta datuak: erregistroa aukeratzeko seinalea (RS) 12 zenbakiko pin digitaletik irtengo da; gaitzeko seinalea (*enable*), 11 zenbakiko pinetik, eta lau bitak, 5, 4, 3 eta 2 pinetatik.

Aldagaiak `float` motakoak izan behar dira; izan ere, zenbaki frakzionarioak dira.

Ondoren, Celsius graduaren ikurra diseinatu da.

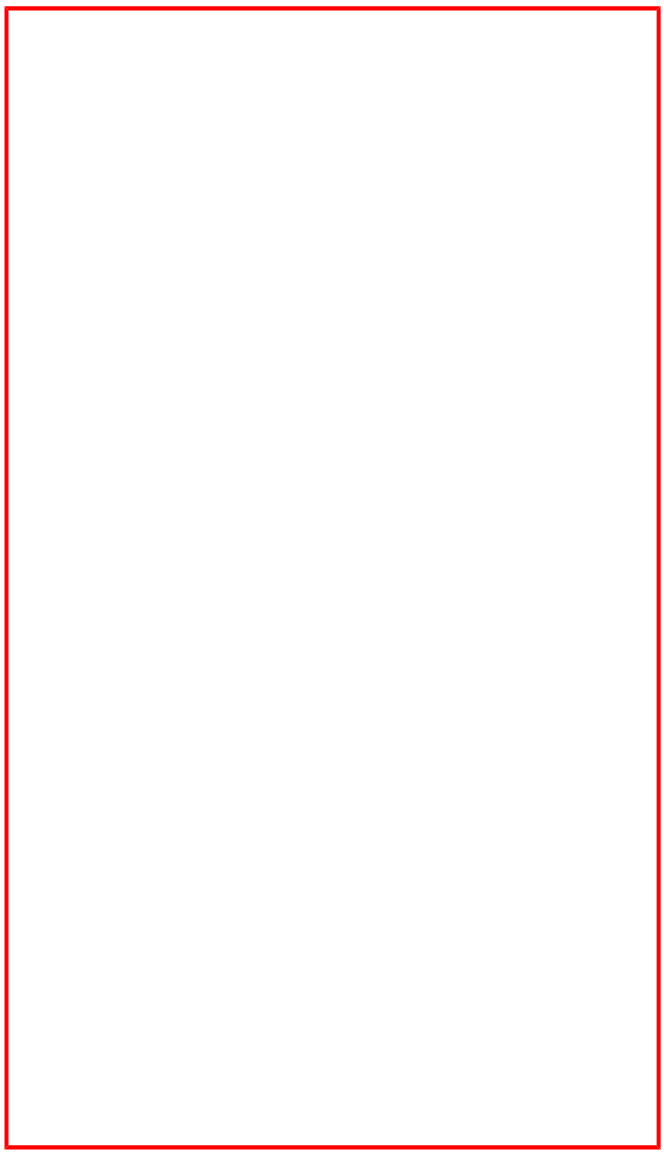
Arduino plakaren eta konputagailuaren arteko serieko datu transferentzia 9.600 bit segundotan gauzatuko da.

Proiektuan erabiliko den LCDak 16 zutabe eta errenkada bi dituela zehaztu da.

Diseinatutako Celsius graduari 1 deitu zaio.

Azkenik, `void loop()` funtzioan etengabe errepikatuko da prozesu hau:

1) 2 zenbakiko sarrera analogikoaren balioa irakurriko da.
2) Balio horren arabera, tentsioa kalkulatuko da.
3) Tentsioaren arabera, tenperatura kalkulatuko da.
4) Konputagailuaren pantailan `"tenperatura: "` bistaratuko da; ondoren, `tenperatura` aldagaia hamartar batekin bistaratuko da; gero, Celsius graduaren ikurra bistaratuko da; aurrerago, `" C da"` bistaratuko da eta kurtsorea azpiko lerroaren hasierara joango da; eta, azkenik, kurtsorea hurrengo lerroaren hasierara joango da.
5) 4 segundo itxarongo du.
6) LCDaren kurtsorea 0 zutabean eta goiko lerroan kokatuko da, gero `tenperatura` berba bistaratuko da; ondoren, kurtsorea 0 zutabean eta beheko lerroan kokatuko da; gero, `tenperatura` aldagaia hamartar batekin bistaratuko da; segidan, Celsius graduaren ikurra bistaratuko da, eta, amaieran, `" C da"` bistaratuko da.

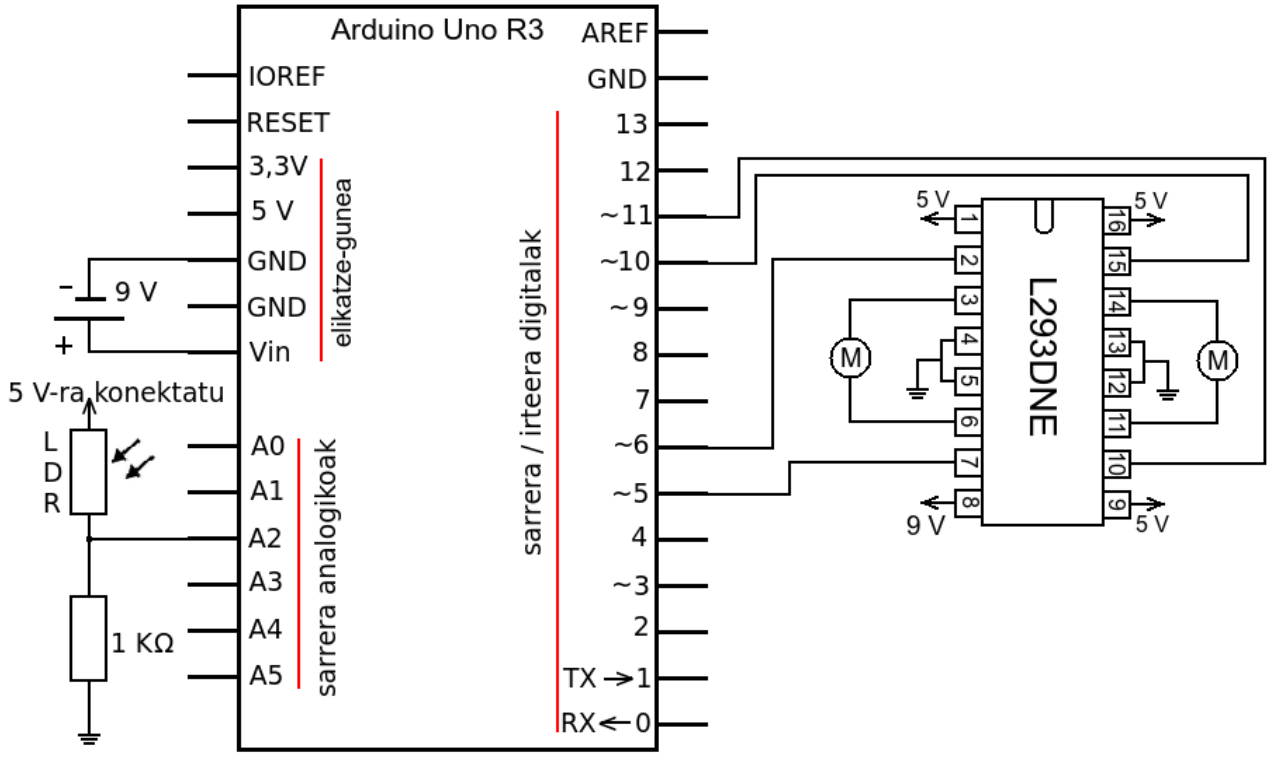


Oharra: Programa bitarraren tamaina 5.744 byte da.

13.PROIEKTUA: KORRONTE ZUZENEKO MOTORREN NORANZKOA ALDATZEA

Proiektu honetan, L293DNE zirkuitu integratua erabilita, korronte zuzeneko motor txiki biren noranzkoa kontrolatzen da.

Zirkuitua



Osagaiak

Proiektu honetan osagai hauek behar dira:

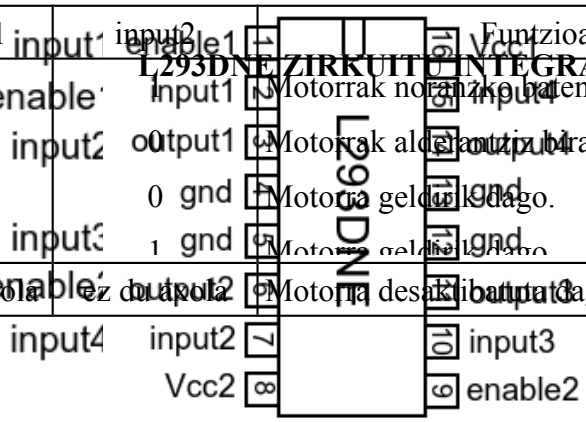
* Arduino plaka
* 1 KΩ-eko erresistentzia
* LDRa
* korronte zuzeneko motor 2
* L293DNE H-zubia edo baliokidea
* 9 V-ko pila

Proiektuaren aldaerak

Baldintza hauek beteko dituen ibilgailua diseinatzea:

- Gelako argitasun normalean ibilgailua gelditua egongo da.
- Argiak LDRa jotzen duenean ibilgailua aurrera joango da.
- LDRa ilunpean badago ibilgailua atzerantz joango da.

enable1	input1	input2	Funtzioa
1	0	1	Motorrak noranzko baten biratzen du.
1	1	0	Motorrak alderantziz biratzen du.
1	0	0	Motorra geldirik dago.
1	1	1	Motorra geldirik dago.
0	ez du axola	ez du axola	Motorra desaktibatuta dago.



L293DNE zirkuitu integratuaren eskema logikoa

L293DNE zirkuitu integratuaren funtzionamendua

Taulan argi eta garbi ikusten da korronte zuzeneko motor baten biraketaren noranzkoa aldatzeko, nahikoa dela elikadura alderantziz jartzea. Bestalde, taula horretan motorretako baten kontrola dago azalduta, eta, beste motorraren taula berdina denez, ez da jarri *enable2*, *input3* eta *input4*. Kontuan izan, L293DNE zirkuitu integratuak beste funtzionamendu era batzuk ere badituela; beraz, komeni da hari buruzko informazio teknikoa ere bilatu eta aztertzea.

Programa

noranzkoa.ino

```
/* Programa honek korronte
    zuzeneko motor biren
    noranzkoa kontrolatzen du          */

int ert_behe = 300;
int ert_goi = 600;
int sen=0;

void setup() {
  pinMode(5,OUTPUT);
  pinMode(6,OUTPUT);
  pinMode(10,OUTPUT);
  pinMode(11,OUTPUT);
}

void loop() {
  sen=analogRead(2);
  if (sen<=ert_behe)
 {digitalWrite(5,HIGH);
   digitalWrite(6,LOW);
   digitalWrite(10,HIGH);
   digitalWrite(11,LOW);
   delayMicroseconds(200);}

  if (sen>ert_behe && sen<=ert_goi)
 {digitalWrite(5,LOW);
   digitalWrite(6,LOW);
   digitalWrite(10,LOW);
   digitalWrite(11,LOW);
   delayMicroseconds(200);}

  if (sen>ert_goi)
 {digitalWrite(5,LOW);
   digitalWrite(6,HIGH);
   digitalWrite(10,LOW);
   digitalWrite(11,HIGH);
   delayMicroseconds(200);}

delay(10);
}
```

Programaren azalpena

Programaren iruzkina.

Hiru aldagaien izenak eta balioak zehaztu dira.
Hala ere, `ert_behe` eta `ert_goi` aldagaien balioak ibilgailua mugituko den gelako argitasunaren arabera doitu beharko dira.
Arduino plakaren 5, 6, 10 eta 11 zenbakiko pin digitalak irteera legez funtzionatuko dutela zehaztu da.

Azkenik, `void loop()` funtzioan, etengabe errepikatuko da prozesu hau:

1) Arduino plakaren 2 pin analogikoa irakurri eta `sen` aldagaian gordeko da.
2) Irakurritako balioa `ert_behe` aldagaiaren balioa baino txikiagoa edo berdina bada, motor biak martxan jarriko dira (atzerantz mugitzeko eran muntatuta egon beharko dira).
3) Programak 200 mikrosegundo itxarongo du.

4) Irakurritako balioa `ert_behe` aldagaiaren balioa baino handiagoa bada eta `ert_goi` aldagaiaren balioa baino txikiagoa edo berdina bada, motor biak geldituko dira.
5) 200 mikrosegundo itxarongo du programak.

6) Irakurritako balioa `ert_goi` aldagaiaren balioa baino handiagoa bada, motor biak martxan jarriko dira (aurrerantz mugitzeko eran muntatuta egon beharko dira).
7) 200 mikrosegundo itxarongo da.

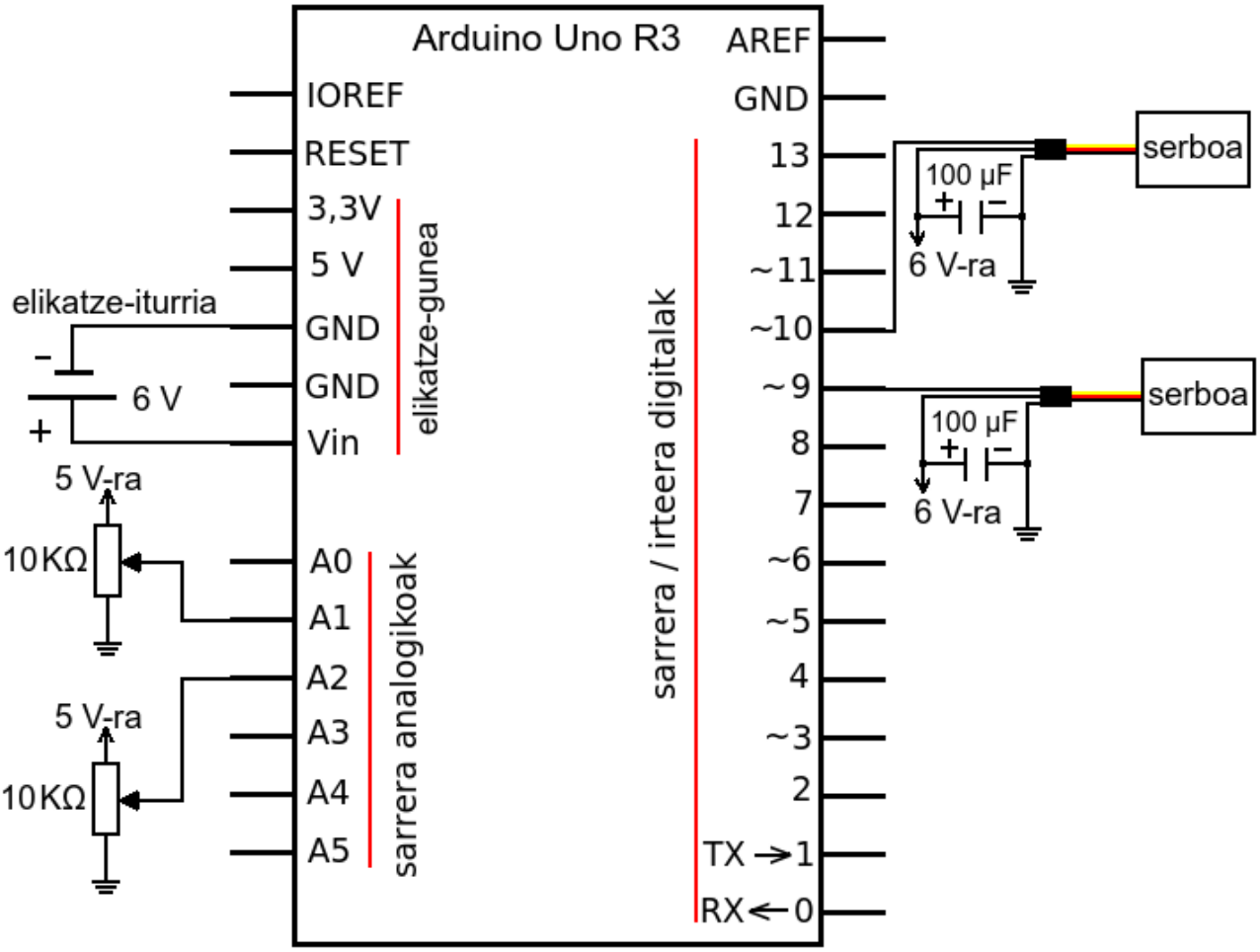
8) 10 milisegundo itxarongo da.

Oharra: Programa bitarraren tamaina 1.328 byte da.

14.PROIEKTUA: POTENTZIOMETRO BIDEZ SERBOMOTOREAK KONTROLATZEA

Proiektu honetan, serbomotore bi erabiltzen dira robotaren biraketa mugimenduak egiteko.

Zirkuitua



Osagaiak

Proiektu honetan osagai elektroniko hauek behar dira:

* Arduino plaka
* 10 KΩ-eko potentziometro bi
* serbomotore bi
* 100 µF-ko kondentsadore bi
* 6 V-ko elikatze-iturria

Hitec HS-311 SERBOMOTOREA

* Kable horia PWM ezaugarria duen irteera digitalera konektatu behar da.
* Kable gorria elikatze-iturriaren 6 V-ra konektatu behar da.
* Kable beltza elikatze-iturriaren GNDra konektatu behar da.

`map()` FUNTZIOA

`map()` funtzioari buruzko azalpena nahi izanez gero, begiratu funtzioen azalpena.

Programa

serbo2.ino

```
/*Programa honek serbomotore bi
kontrolatzeko balio du        */

#include <Servo.h>

Servo serboa_oinarria;
Servo serboa_besoa;

int pot1;
int pot2;
int angelua1;
int angelua2;

void setup(){
  serboa_oinarria.attach(9);
  serboa_besoa.attach(10);
  }


void loop(){

  pot1 = analogRead(1);
  pot2 = analogRead(2);


  angelua1 = map(pot1, 0, 1023, 0, 179);
  angelua2 = map(pot2, 0, 1023, 0, 179);

  serboa_oinarria.write(angelua1);
  serboa_besoa.write(angelua2);
  delay(15);
}
```

Programaren azalpena

Programaren iruzkina.

`Servo` izeneko programa-liburutegia gehitu zaio idatzitako programari; horrela, serbomotoreak oso erraz kontrolatuko dira funtzio berezi batzuk erabilita.
Serbomotore bakoitzari bere izena jarri zaio: robotaren oinarria biratuko duenari `serboa_oinarria`, eta besoa mugituko duenari, `serboa_besoa`.
Potentziometroen izenak `pot1` eta `pot2` izango dira.
Oinarriaren biraketa-angeluari `angelua1` deitu zaio, eta besoaren biraketa-angeluari, `angelua2`.

Oinarria biratuko duen serbomotorea 9 zenbakiko pin digitalean konektatu da, eta besoa biratuko duen serbomotorea, 10 zenbakiko pinean.

Azkenik, void loop() funtzioan, etengabe errepikatuko da prozesu hau:

1) 1 zenbakiko pin analogikoa irakurri, eta haren balioa `pot1` aldagaian gordeko da.
2) 2 zenbakiko pin analogikoa irakurri, eta haren balioa `pot2` aldagaian gordeko da.
3) Biraketa-angeluak lortzeko, `map` funtzioa erabiliko da.
4) Lortutako angeluak serbomotoreei emango zaizkie biraketak egin ditzaten, baina, biraketak egiteko serbomotoreek astia behar dutenez, 15 ms-ko denbora eman zaie.
Oharra: Programa bitarraren tamaina 2.740 byte da..