VERSIJA
1.0

MOKOMASIS RINKINYS

ELEKTRONIKA

IŠMANIOSIOS GRANDINĖS SU ARDUINO

ANODAS
electronics

www.anodas.lt

Medžiagą parengti padėjo:

Edvinas Mačiulis

Mindaugas Dagys

Marius Narvilas

Ieva Marija Dautartaitė

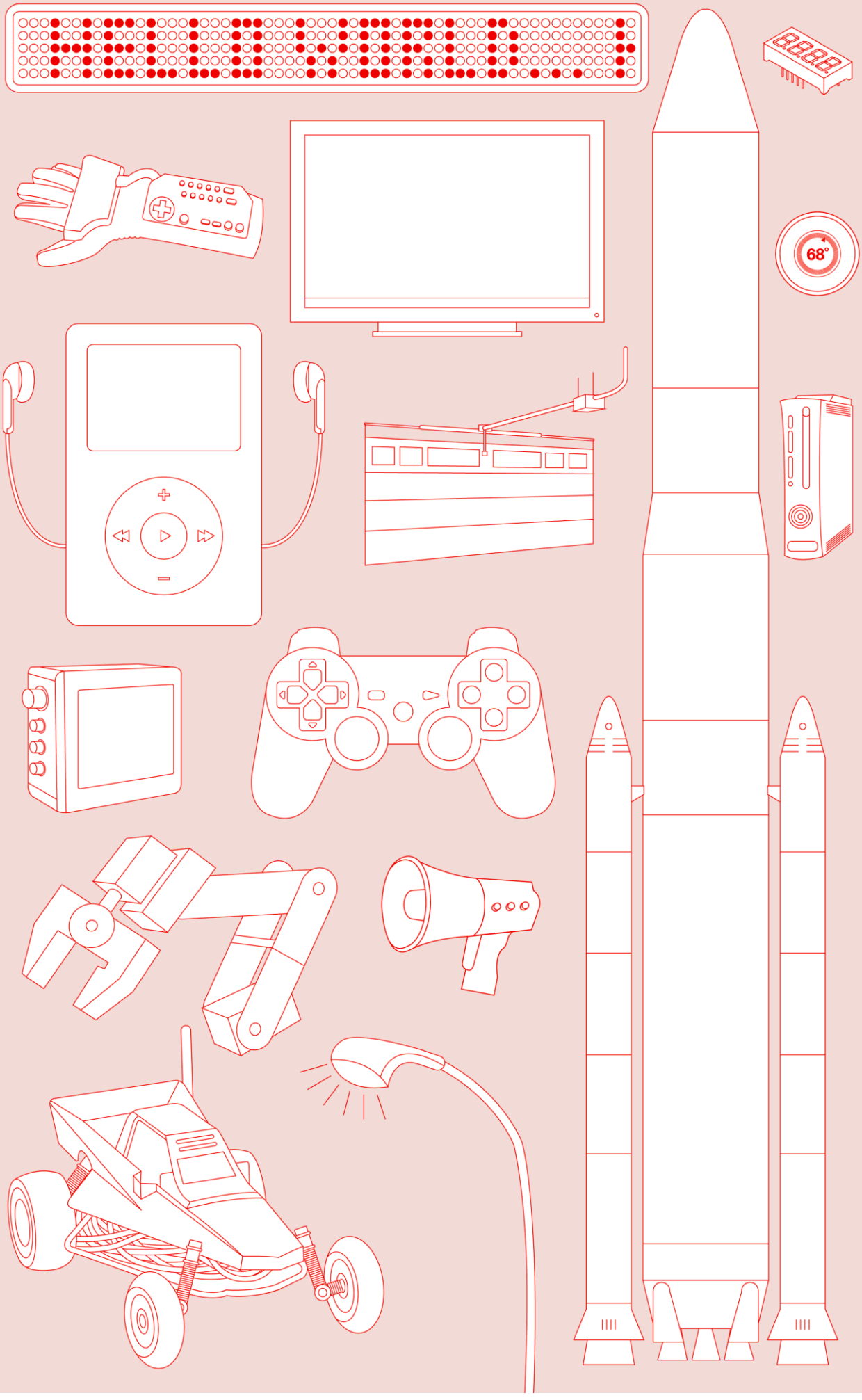
Artūr Kadzevič

Programinis kodas suderintas su Arduino IDE 1.6.11 versija

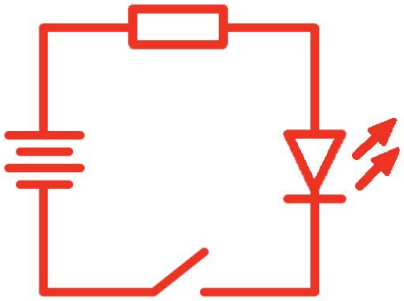
http://arduino.cc/en/Main/Software

Atnaujinta. v1.0 – 2016.09.06

Turinys:


8888
68°

Apie elektrines grandines



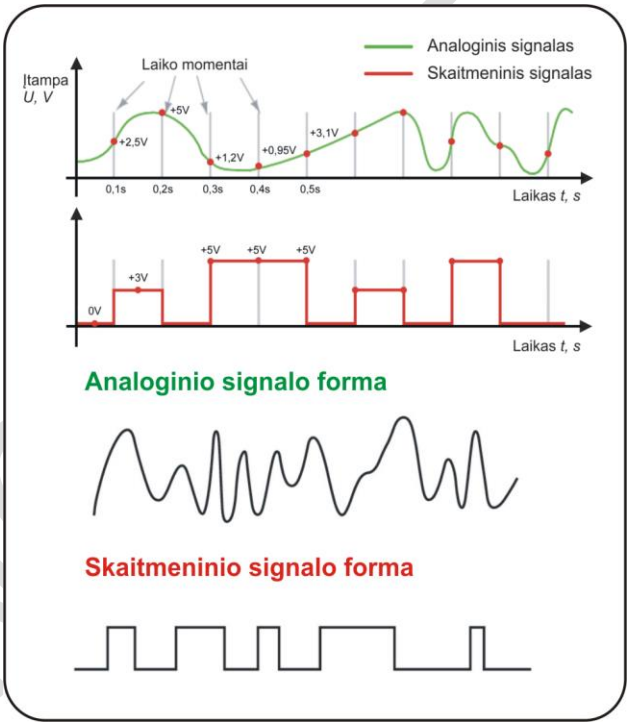
Kasdieną mus supa įvairiausi elektroniniai įrenginiai, prietaisai, kurie apdoroja, valdo ir perduoda labai daug informacijos. Visų jų pagrindas elektrinės grandinės.

Elektrinėse grandinėse visa informacija (komandos, skaičiai, garsai, vaizdai) perduodama **elektriniais signalais** – elektrinės įtampos (U) pokyčiais, impulsais.

Pagal elektrinio signalo apdorojimo pobūdį grandinės yra skirstomos į **analogines**, **skaitmenines**.

Analoginis signalas, skirtingu laiko momentu, tarp aukštosios ir žemosios įtampos lygmenų turi be galo daug reikšmių.

Skaitmeninis signalas turi tik dvi reikšmes, kurias atitinka du elektrinės įtampos lygmenys – aukštasis ir žemasis (žr. 1 pav.).

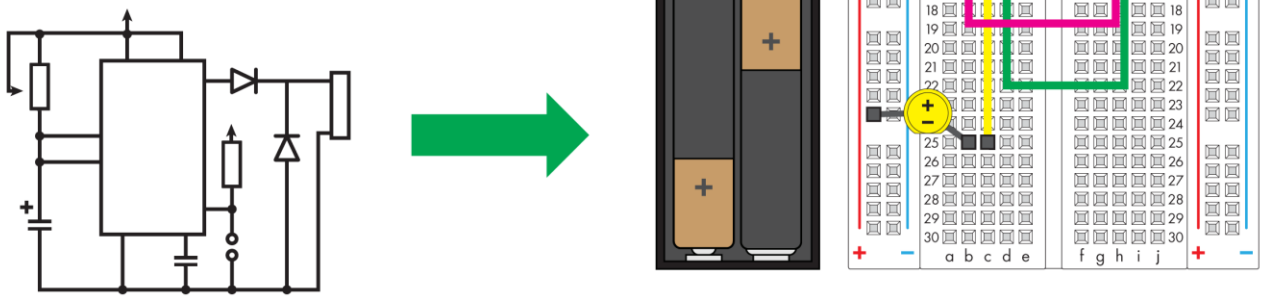


1 pav. Skaitmeninis ir analoginis signalas

Elektrinių grandinių prototipavimas

Geriausias būdas išsiaiškinti kaip veikia įvairūs elektroniniai prietaisai, jų grandinės, nustatyti ir suvokti jas sudarančių elementų veikimo, tarpusavio sąveikos dėsningumus – jas surinkti ant **maketo plokštės** (žr. 2 pav.). Toks aiškinimosi būdas vadinamas elektrinių grandinių prototipavimu ar tiesiog – maketavimu ar modeliavimu.

Surenkant grandinę ant maketo plokštės, nereikia lituoti elementų, dalių kontaktai, jungiamieji laidai tiesiog įstatomi į specialius kontaktinius lizdus (žr. *Kaip sudaryta maketo plokštė*). Modeliuodami elektrinę grandinę, galime eksperimentuoti su įvairiais jos elementais, funkciniais parametrais, savybėmis.



2 pav. Elektrinės grandinės surinkimas ant maketo plokštės

Elektronikos elementai ir prietaisai

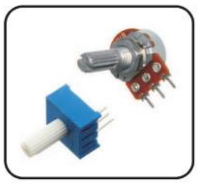
Elektrinės grandinės sudarytos iš įvairių **puslaidininkinių elementų** (diodų, tranzistorių) bei integrinių grandynų (mikroschemų). Jų pagrindas **puslaidininkinės medžiagos**, tokios kaip silicis, germanis, kurių elektrinis laidumas kinta kintant temperatūrai bei priemaišų kiekiui. Todėl **puslaidininkiniai elementai** naudojami signalų kūrimui ir apdorojimui, nes gali reguliuoti srovę, ją padidindami, praleisdami ar nepraleisdami. **Pasyviniai grandinių elementai** - rezistoriai, kondensatoriai, induktyvumai, transformatoriai ir kt.

Rezistorius

Tam tikros varžos prietaisas, naudojamas srovės apribojimui pagal Omo dėsnį **I=U/R**, kur **I** – srovės stipris, **U** – įtampa ir **R** – elektrinė varža. Elektrinės varžos matavimo vienetas – *omas* (Ω). Vartojami ir didesni varžos vienetai - *kiloomas* (kΩ), *megaomas* (MΩ) ir kt.
1 kΩ = 1000 Ω, 1 MΩ = 1000 000 Ω.
Rezistoriaus varžos dydis nurodomas ant korpuso spalviniu kodu (žiūrėti žemiau).

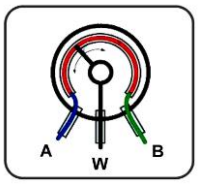
Jungiant: Neturi prijungimo ypatybių.

Potenciometras



Keičiamos varžos prietaisas, turintis tris išvadus (du fiksuotus ir tarp jų slankiojantį perkeliamą). Perkeliant slankiojantį kontaktą, keičiasi varžų santykis tarp vidurinės (W) ir kraštinių kojelių (A, B).

Jungiant: Būtina atsižvelgti, kad potenciometro vidurinė kojelė (W) yra sujungta su slankiojančiu kontaktu.



Diodas

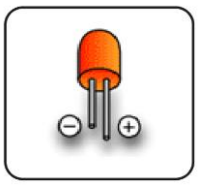
Puslaidininkis prietaisas. Dažniausiai pritaikoma jų savybė praleisti elektros srovę tik viena kryptimi. Turi dvi kojeles – *anodą* (+) ir *katodą* (-).

Jungiant: Būtina atsižvelgti į diodo išvadų kojeles. Turi būti prijungtas taip, kad elektros srovė tekėtų iš anodo į katodą. Katodas yra pažymėtas ant diodo korpuso juodu ar pilku žiedeliu.

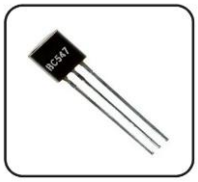
Šiesos diodas (LED)

Puslaidininkis prietaisas, spinduliuojantis šviesą, kai per jį teka silpna elektros srovė. Turi dvi kojeles – *anodą* (+) ir *katodą* (-).

Jungiant: Būtina atsižvelgti į šviesos diodo išvadų kojeles. Turi būti prijungtas taip, kad elektros srovė tekėtų iš anodo į katodą. Katodas yra trumpesnis išvadas, taip pat – išvadas esantis šiek tiek plokštesnėje diodo pusėje.

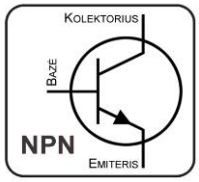


Tranzistorius

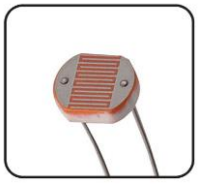


Puslaidininkinis prietaisas paprastai naudojamas elektriniams signalams sustiprinti ar nukreipti. Tranzistorius turi tris kojeles: *emiterį* (E), *bazę* (B) ir *kolektorių* (K). Tranzistoriai būna dviejų tipų: NPN (elektros srovė teka iš kolektoriaus į emiterį) ir PNP (elektros srovė teka iš emiterio i kolektorių).

Jungiant: Būtina atsižvelgti į tranzistoriaus tipą ir kojelių išdėstymą.



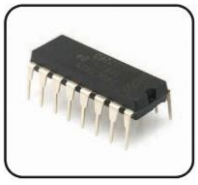
Fotorezistorius



Fotorezistoriai veikia dėl fotolaidumo reiškinio. Neapšviesto fotorezistoriaus varža yra didelė. Veikiant šviesai varža sumažėja.

Jungiant: Neturi prijungimo ypatybių.

Integrinis grandynas



Tai mikrograndynas talpinantis viename korpuse keletą šimtų tūkstančių rezistorių, kondensatorių, tranzistorių ir turintis atitinkamą paskirtį (atlikti sudėtingesnėms ir tikslesnėms užduotims nei tą gali padaryti pavieniai tranzistoriai su pasyviaisiais elementais). Pagal elektrinio signalo apdorojimo tipą integriniai grandynai skirstomi į analoginius ir skaitmeninius (loginiai elementai, skaitikliai, mikrovaldikliai, mikroprocesoriai ir t. t.).

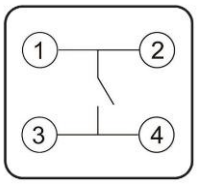
Jungiant: Būtina atsižvelgti į grandyno kontaktinių kojelių išdėstymo tvarką (kuri kojelė yra pirma) ir jų numeraciją.

Tęsinys...

Mygtukas

Mechaninis prietaisas, sujungiantis (kol yra nuspaustas) į jį ateinančius du elektrinius kontaktus į vieną.

Jungiant: Būtina atsižvelgti į mygtuko kontaktinių kojelių išdėstymo tvarką.



Servo variklis

Servo varikliai naudojami ten, kur reikalingas tikslus pasukimo kampas. Jų apsisu-kimo kampas yra nuo 00 iki 1800 (katrais daugiau, pvz. 3600), bet jie nesisuka pastoviai. Jų valdymas vyksta nuo 1 ms (00) iki 2 ms (1800) trukmės impulsais, kurie yra paverčiami sukamuoju judesiu, pasukančiu veleną atitinkamu kampu. Paprastai jie maitinami 4,8-6 V įtampa (raudonas ir juodas laidai), judėjimui reikalingas valdymo signalas (baltas laidas).

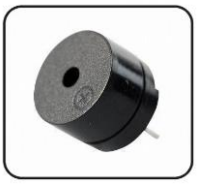
Jungiant: Būtina atsižvelgti į maitinimo ir valdymo laidų išdėstymą, kodinę spalvą.

Nuolatinės srovės (DC) variklis

Nuolatinės srovės (DC) variklis yra ektromechaninis prietaisas elektros energiją paverčiantis sukamuoju judesiu. Variklio sukimosi kryptis priklauso nuo per jį tekančios elektros srovės krypties.

Jungiant: Kadangi variklio sukimosi kryptis priklauso nuo per jį tekančios elektros srovės krypties, būtina atsižvelgti į matinimo šaltinio polių prijungimą.

Pjezo skambutis



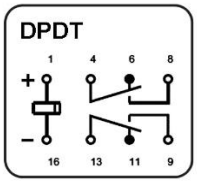
Elektromechaninis prietaisas, kuriuo tekančios elektros srovės impulsai sužadina jo garsinį toną.

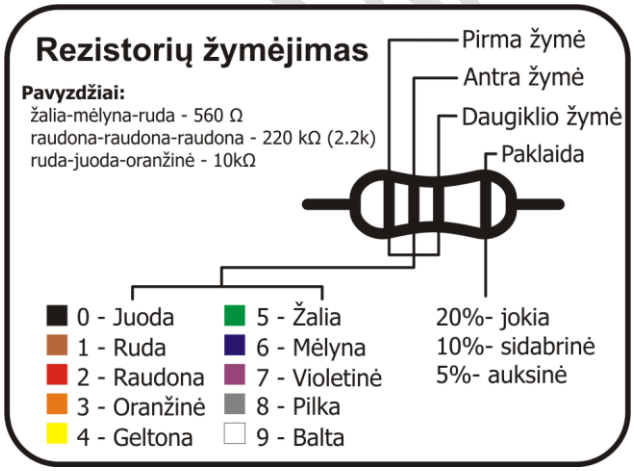
Jungiant: Būtina atsižvelgti, kad prietaisas yra poliarizuotas. Anodas dažniausiai žymimas skambučio viršuje, katodo kojelė yra trumpesnė.

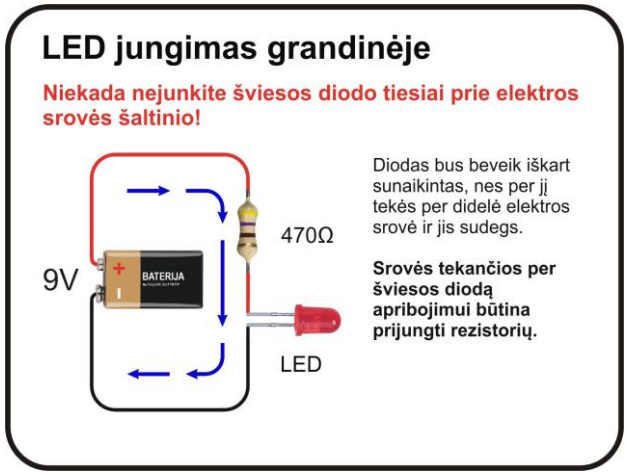
Rėlė

Prietaisas, kuriame viena grandine tekanti srovė sujungia ar atjungia kitą, nepriklausomą, relėje elektriškai nesusijusią elektros grandinę. Paprasčiausia elektromagnetinė relė turi elektromagnetą, kuriuo tekanti srovė perkelia į kitą padėtį vieną ar kelis judamus kontaktus. Šie kontaktai gali būti sujungiantys, atjungiantys arba perjungiantys. Valdančiai srovei išnykus, spyruoklė grąžina kontaktus į pradinę padėtį.

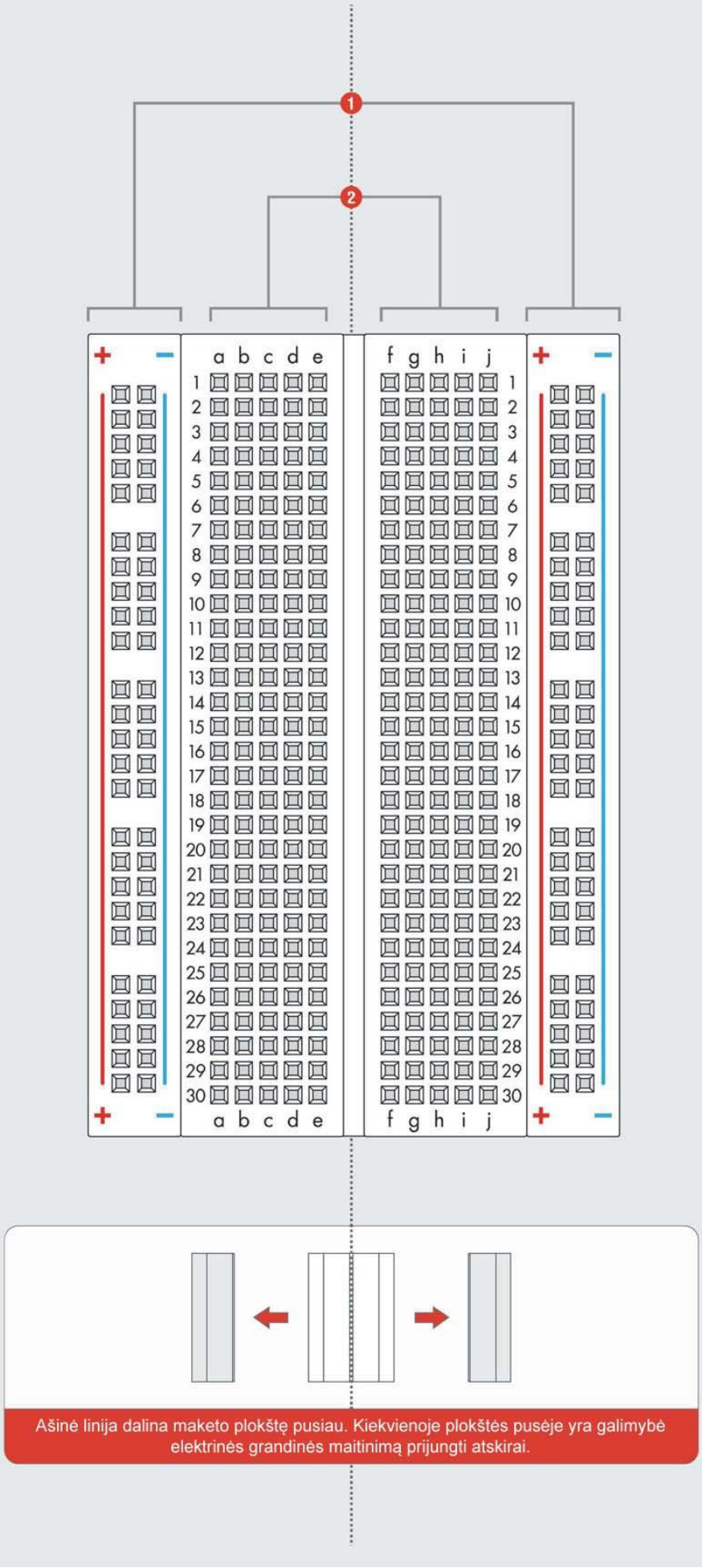
Jungiant: Būtina atsižvelgti į rėlės kontaktinių kojelių išdėstymo tvarką.







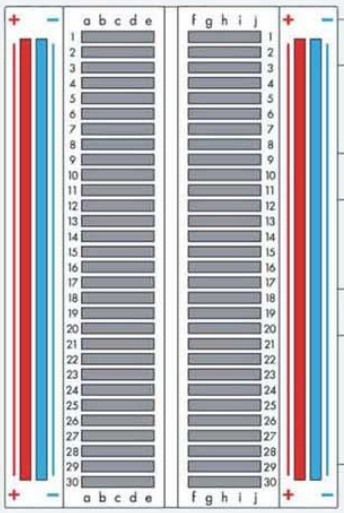
Maketo plokštė



1 **Vertikalios jungtys** (Maitinimo šaltinio (+) ir (-) / Įžeminimas)

2 **Horizontalios jungtys** (kontaktiniai lizdai a-e ir f-j)

Kaip viskas yra sujungta?



Maitinimo šaltinio (+)

Prijungus maitinimo šaltinio (+) jis bus prieinamas visoje vertikalioje juostoje

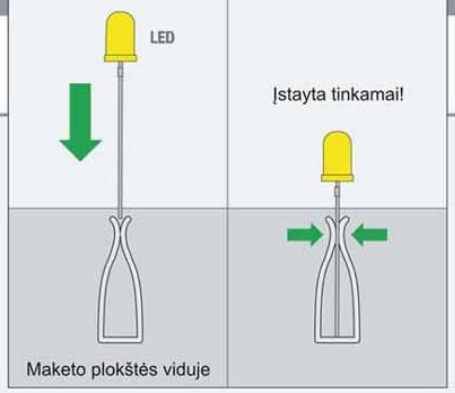
Matitinimo šaltinio (-) / Įžeminimas

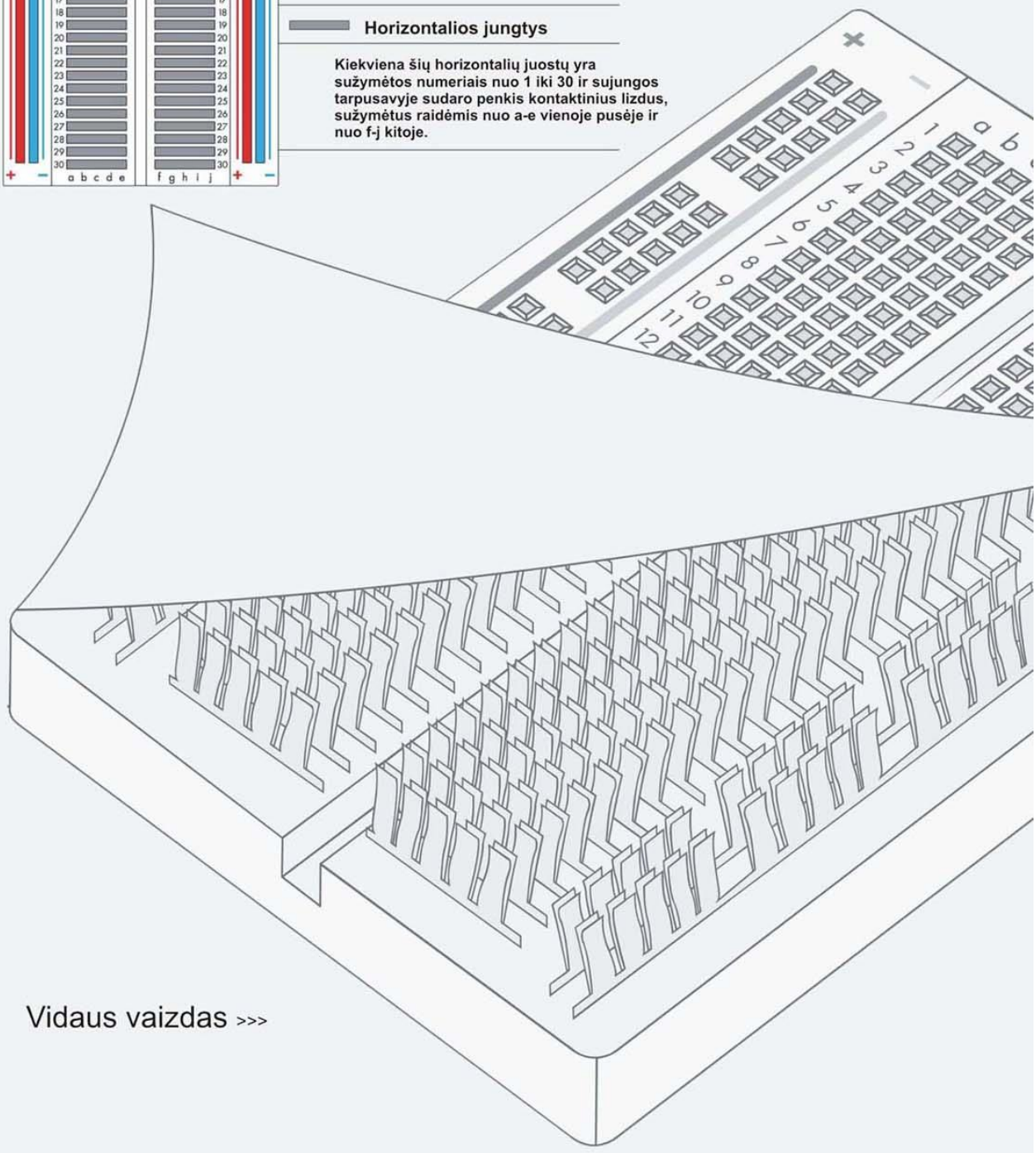
Prijungus matinimo šaltinio (-) jis bus prieinamas visoje vertikalioje juostoje

Horizontalios jungtys

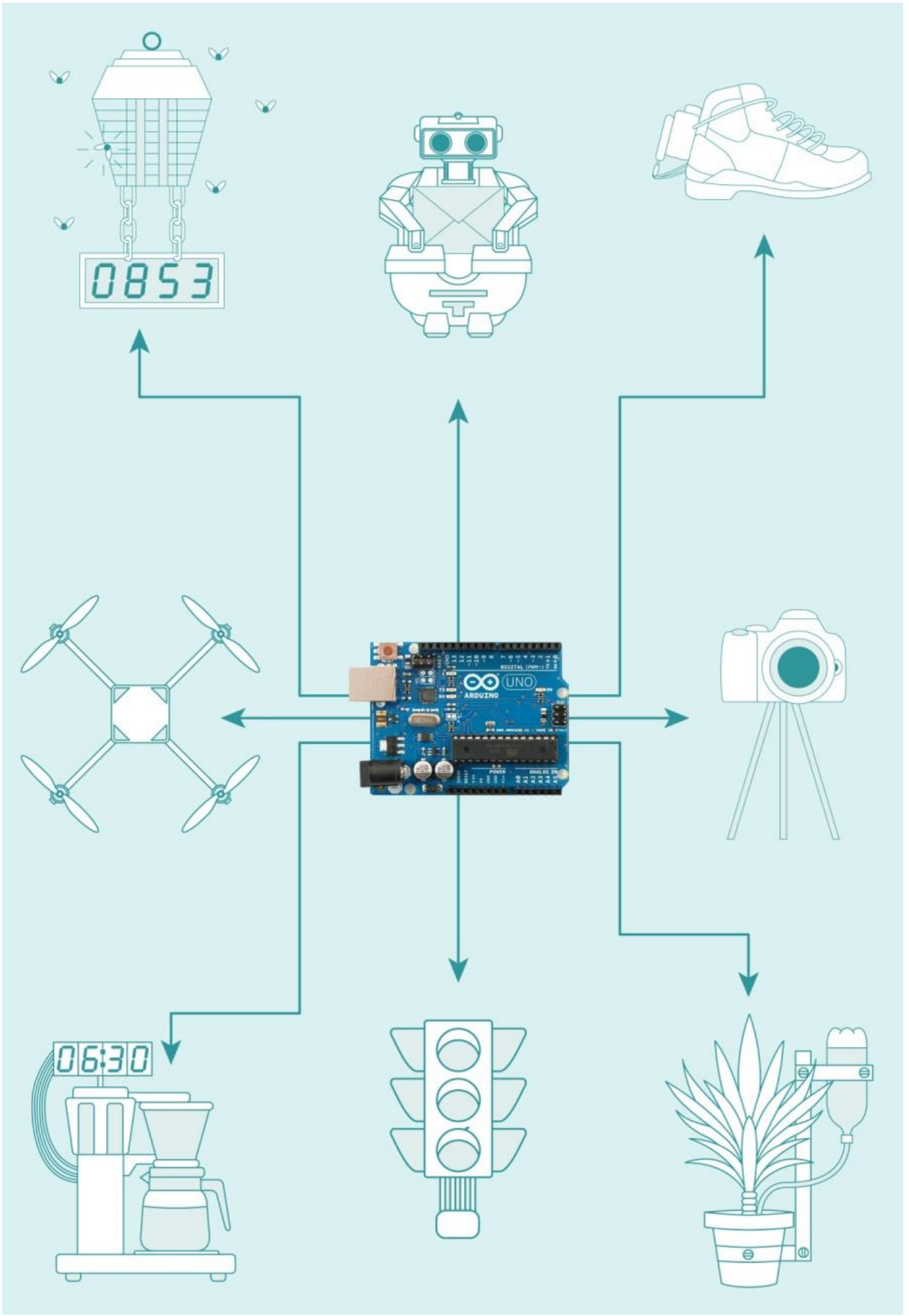
Kiekviena šių horizontalių juostų yra sužymėtos numeriais nuo 1 iki 30 ir sujungos tarpusavyje sudaro penkis kontaktinius lizdus, sužymėtus raidėmis nuo a-e vienoje pusėje ir nuo f-j kitoje.

Komponento įstaymas



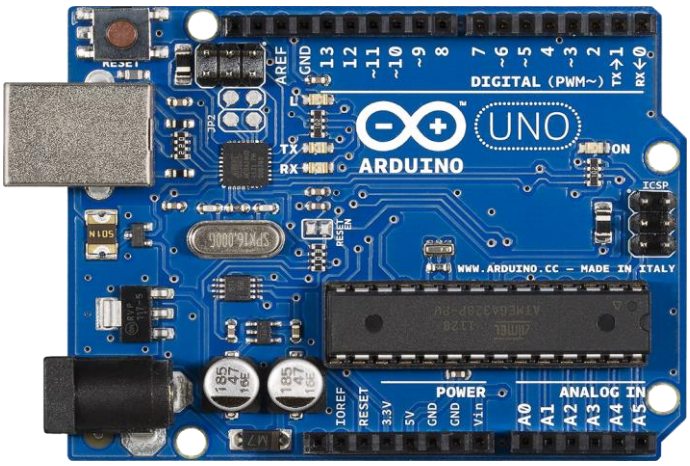


Vidaus vaizdas >>>

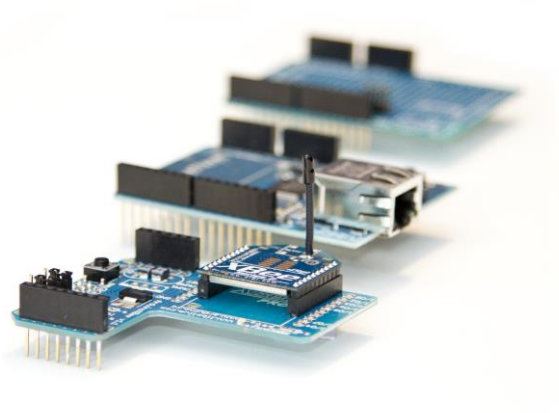

0853
UNO
ARDUINO
06:30

Kas yra Arduino?

Arduino – tai programuojamas elektroninis prietaisas, naudojantis **Atmel** firmos mikrovaldiklius, kurie pasižymi savo itin didelėmis galimybėmis**.** Tai atviro kodo platforma, kodėl ji ir tapo populiari visame pasaulyje. Šis instrumentas suteikia kuriamam projektui didesnes galimybes "jausti" ir kontroliuoti fizinį pasaulį.



Arduino UNO



Arduino priedeliai (angl.k. *shield*)

Arduino naudoja C ir C++ programavimo kalbų "mišinį". Išmokti šią programavimo kalbą nėra sudėtinga. Jos aprašymą rasite:

http://arduino.cc/en/Reference/HomePage

Trumpą arduino programavimo kalbos aprašymą rasite **4 priede**.

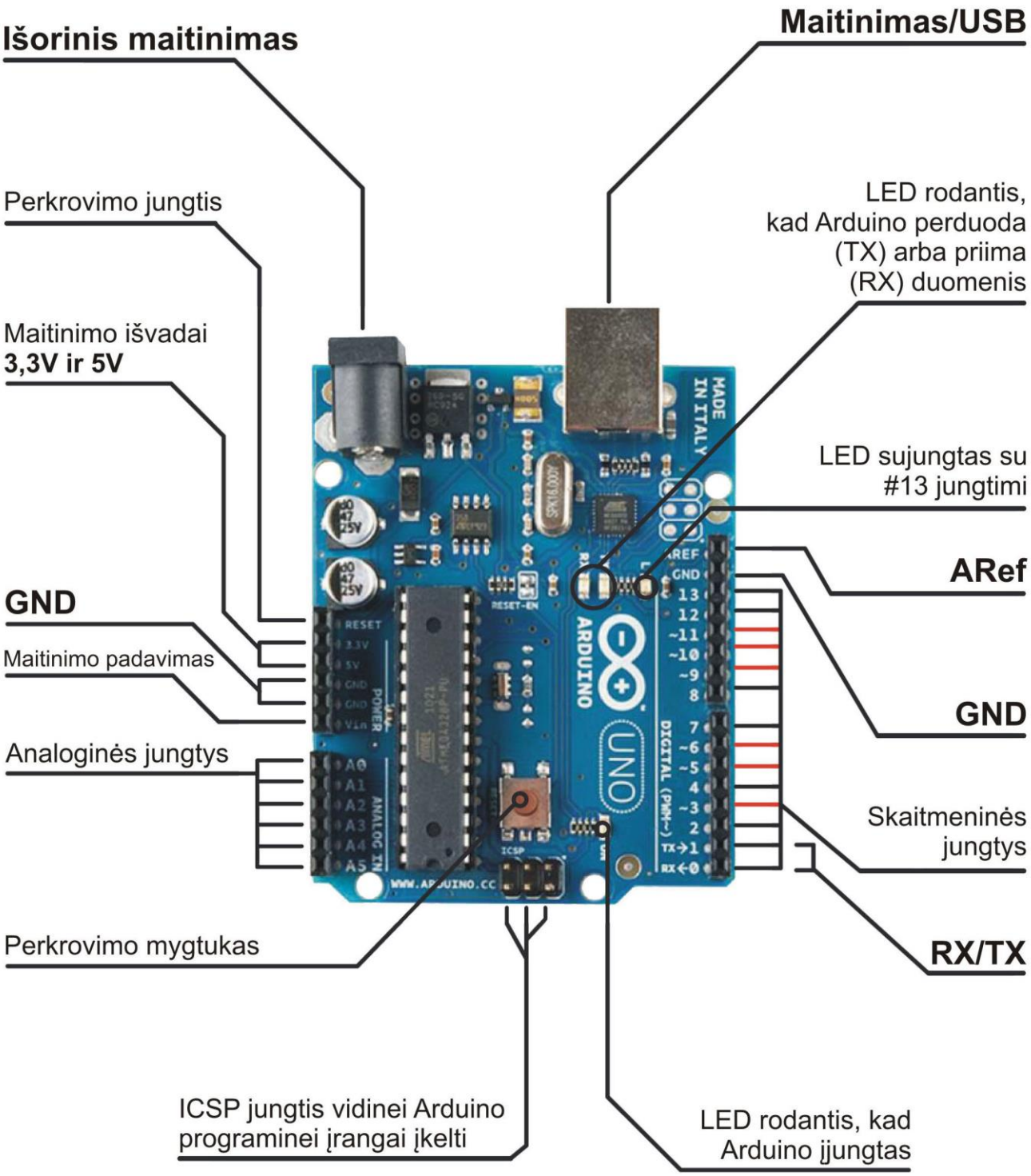
Dar vienas geras **Arduino** privalumas – tai, kad jam yra sukurti papildomi priedeliai (angl.k ***Shield***), kurie suteikia galimybę **Arduino** prijungti prie interneto, WiFi tinklo, Bluetooth ryšio. Taip pat gali suteikti galimybę naudoti SD atminties korteles ir dar daug visokių kitų galimybių.
Su **Arduino** žaisti galima kaip su LEGO – dėlioji detales ir iš karto matai rezultatą. Na, bet aišku privalai laikytis tam tikrų taisyklių, kad nesudegintum mikrovaldiklio. Dėl šios priežasties "nepatyrusiems" siūlyčiau pradėti nuo **Arduino UNO**.

Arduino UNO ypatingas tuo, kad jis nėra toks brangus kaip **Arduino MEGA** ar kita **Arduino** šeimos platforma, taip pat jame yra galimybė pakeisti pagrindinį integrinį grandyną (mikrovaldiklį).
Daugiau informacijos apie **Arduino** galite rasti oficialioje svetainėje:

http://arduino.cc/

Arduino tai puiki priemonė praplėsti moksleivių ir studentų žinias inžinerijos srityje, nes jo pagalba išbandoma, ir elektronika, ir mechanika, ir programavimas vienu metu.

Arduino UNO išvadai



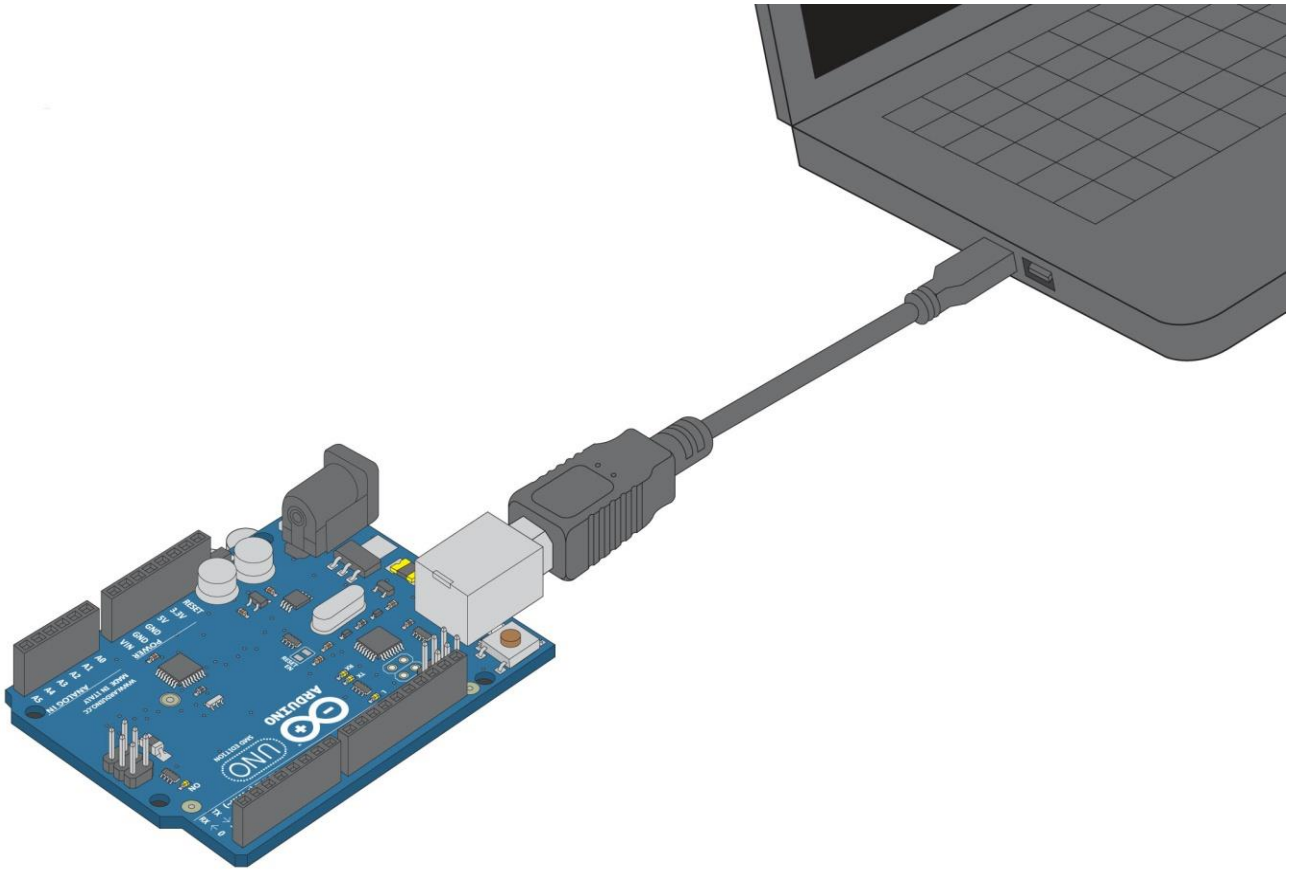
Arduino Uno

Arduino IDE diegimas į kompiuterį

Pirmas žingsnis. Tam, kad pradėti darbą su Arduino kompiuteryje kuris naudoja *Microsoft Windows* operacinę sistemą reikia įdiegti tvarkykles ir programavimo aplinką (IDE). Atsisiųskite į savo kompiuterį programinę įrangą iš

http://arduino.cc/en/Main/Software

ir ją įdiekite į kompiuterį. Dabar paimkit USB laidą, kurį gavote kartu su Arduino ir įjunkite vieną galą į kompiterį, o kitą į Arduino taip kaip parodyta paveikslėlyje.



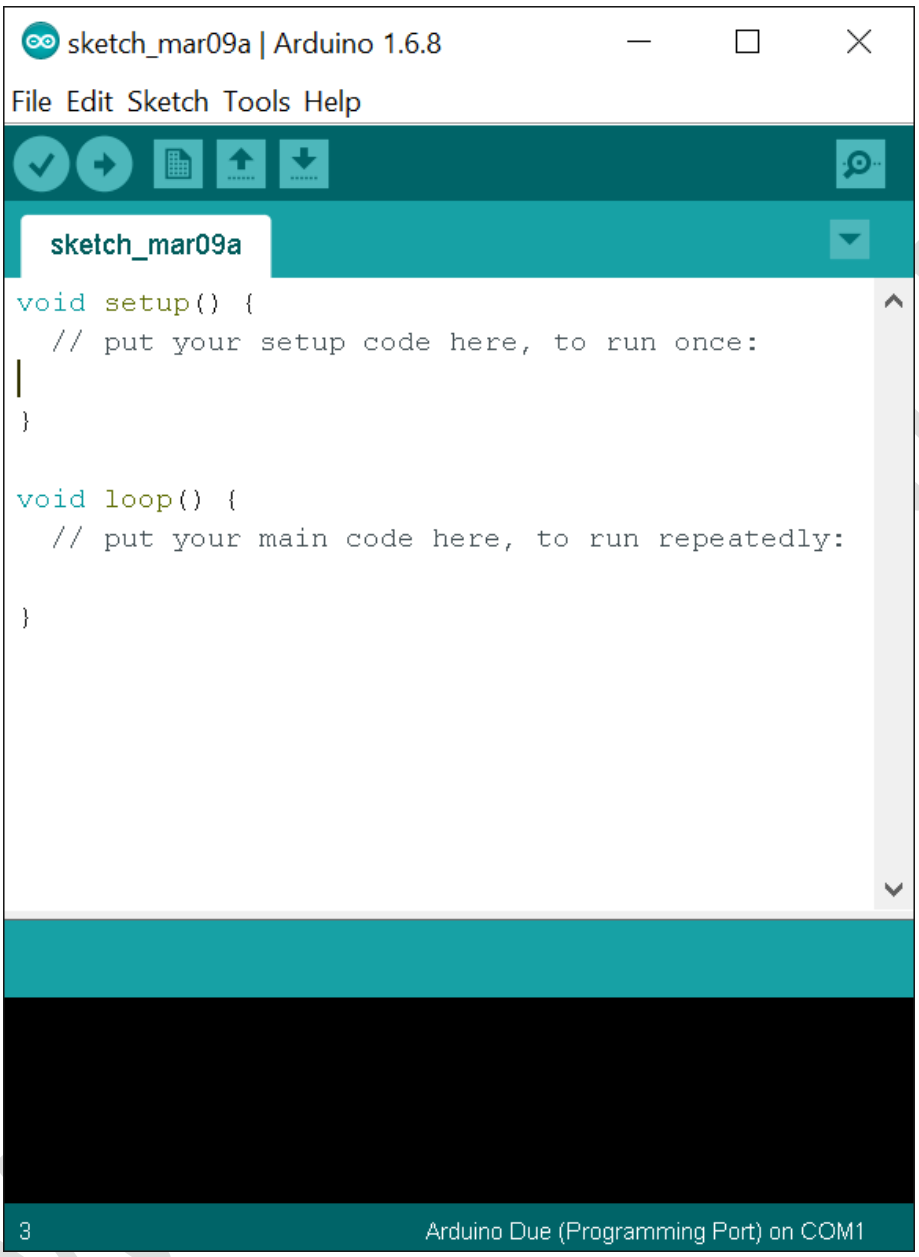
Ant Jūsų Arduino įsižiebs lemputės iš kurių viena pradės nuolatos mirksėti. Jei taip atsitiko vadinasi Jūsų Arduino veikia, bet su kompiuteriu dar nebendrauja. Kompiuteris aptiks naują USB įrenginį kuriam pasiūlys diegti tvarkykles, tačiau atomatinė paieška neduos rezultato. Nustatyte, kad tvarkyklės sistema ieškotų kataloge c:\Arduino-1.0.4\drivers (kiekviename kompiuteryje vieta ir pavadinimai gali šiek tiek skirtis).
Kompiteris aptiks tvarkykles ir jas įdiegs. Norėdami patikrinti ar viskas tvarkoje spustelėkite dešinį pelės klavišą ant piktogramos „My Computer“ ir pasirinkite „Properties“, o po to „Devices“. Ties USB įrenginiais turi atsirasti užrašas kad prijungtas Arduino. Jei ties USB įrenginiu yra šauktukas tai pakartokite tvarkyklės diegimą. Angliška detali instrukcija yra čia:

http://arduino.cc/en/Guide/Windows

Jei klaidų nebuvo tai Jūsų kompiuteris pasiruošęs darbui su Arduino.

Arduino programavimo aplinka (IDE)

Diegdami tvarkykles mes jau įdiegėme programavimo aplinką. Dabar ją reikai paleisti. Paleiskite failą **c:\arduino-1.6.8\arduino.exe** ekrane pamatysite langą:



Tai Arduino programavimo aplinka (*IDE – Integrated Development Environment*).

Nueikite su pele į „Tools“, „Board“ ir pasirinkite „Arduino Uno“ (arba tokią plokštę, kokią esate prijungę).

Puiku dar liko nustayti COM jungties numerį (COM Port): „Tools“, „Serial Port“ ir nustatykite tokį, kuris buvo nurodytas prie „Devices“ kai diegėte tvarkyklę.

Jei nustatysite neteisingą numerį, Jūsų Arduino tiesiog nepriims įkeliamų programų, eikite atgal į „Tools“, „Serial port“ ir nustatykite kitą.

Kaip patikrinti ar teisinga COM jungtis? Betikrindami išvien išmoksime ir bazinio programavimo. Eikite į „File“, „Examples“, „Basics“ ir pasirinkite „Blink“.

{GRND-01}

Pradžia –
Mirksintis šviesos diodas

Ką darysime

LED (šviesos diodai) yra naudojami visuose protinguose dalykuose, rūšyse, todėl mes įtraukiame juos į ši rinkinį. Pradėsime nuo ko nors paprasto - vieną jų įjungdami ir išjungdami pakartotinai, sukursim mirksintį efektą. Pradėdami pasiimkite žemiau išvardintas dalis, pin išdėstymo lapą ir bandomąją lentelę ir viską sujunkite. Kai grandinė yra surinkta, jums reikia įkelti programą. Norėdami tai padaryti, prijunkite „Arduino“ plokštę į USB jungtį. Tada pasirinkite tinkamą jungtį įrankiuose > tam skirta (serijinė) jungtis> (jungtis skirta „Arduino“ programai). Po to įkelkite programą: kelti failą> kelti I/O jungties lentelė (ctrl+U). Galiausiai mėgaukitės šlove galėdami valdyti žibintus.

Jei turite problem su įkėlimu pilną problem tvarkymo vadovą rasite čia: http://ardx.org/TRBL

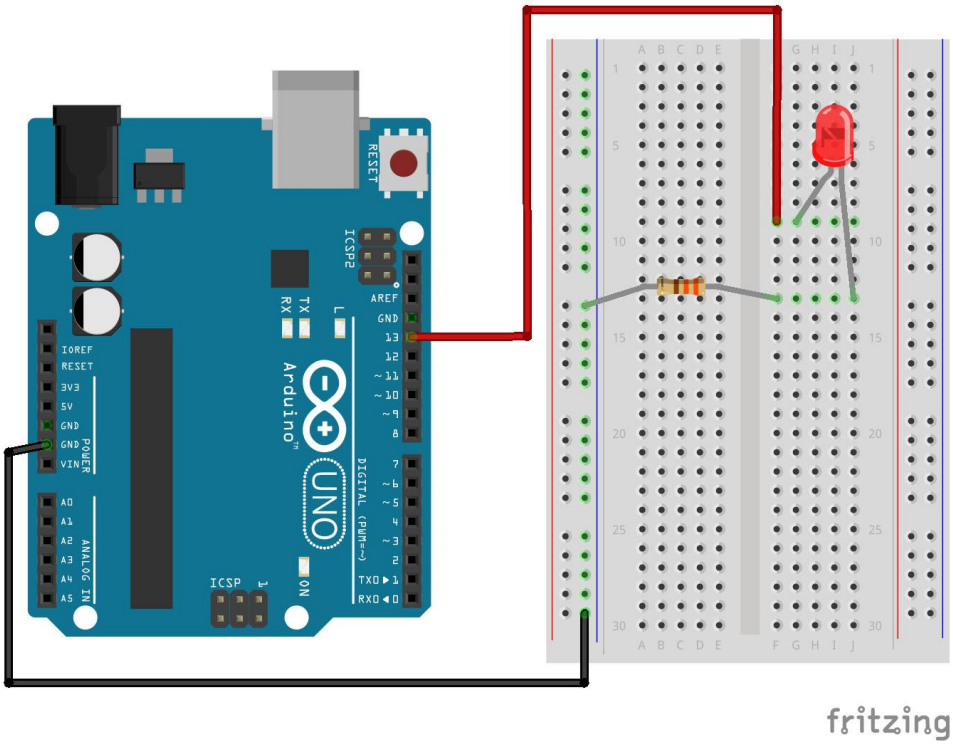
Grandinės dalys

330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x1

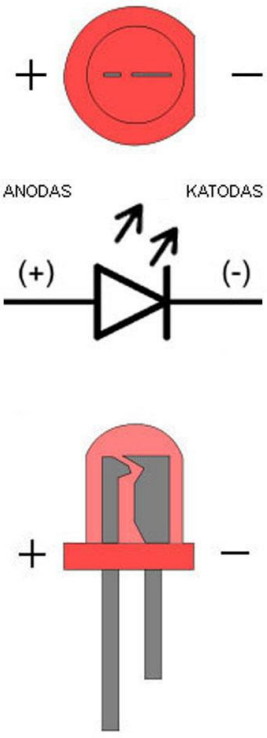
Jungiamieji laidai
x2

5 mm šviesos diodas
x1

Surinkimo grandinė



Šviesos diodas:



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-01: Pradžia - Mirksintis šviesos diodas             |
 *     ------------------------------------------------------------
 *
 * Mirksėjimas
 * Pakartotinai kas sekundę įjungia ir išjungia šviesos diodą.
 * Grandinė:
 * Šviesos diodas, prijungtas iš 13 skaitmeninio kontakto į įžeminimą.
 * Pastaba: daugumoje Arduino plokščių šviesos diodas jau turėtų būti prijungtas prie 13
 * kontakto, tad papildomo jungti nebereikia.
 *
*/

int ledPin =  13;    // LED'as prijungtas prie 13 skaitmeninio kontakto

// Sąrankos nustatymai() paleidžiami vieną kartą, kai montavimas prasideda
 void setup()   {
  // inicijuojame šviesos diodo (LED) kaištį(pin), kaip išvestį (OUTPUT):
 pinMode(ledPin, OUTPUT);
}

// ciklas() vis kartojasi ir kartojasi,
// kol Arduino turi energijos

void loop()
{
  digitalWrite(ledPin, HIGH);   // įjungti LED'ą
  delay(1000);                  // lukterėti sekundę
  digitalWrite(ledPin, LOW);    // išjungti LED'ą
  delay(10);                  // lukterėti sekundę
 }
```

Neveikia? (Trys dalykai bandymui)

Šviesos diodas nešviečia?	**Neisikelia programa**	**Dar jokio pasisiekimo?**
LED'ai veikia tik viena kryptimi. Pabandykite išimti ir apsukti LED atvirkščiai (nesijaudinkite, blogai įdėtas diodas nebus pažeistas).	Kartais nutinka ir taip. Dažniausia priežastis - supainiotas įvado numeris. Jį galima pakeisti per tools>serial port>	Sugadinta plokštė - menki juokai. Susisiekite su gamintojais arba savo tiekėju.

Padaryti geriau?

Kontakto keitimas:
LED'as prijungtas prie 13 kontakto, bet galima naudoti bet kurį kitą „Arduino“ kontaktą. Keisdami kontaktą išimkite jungiamąjį laidą iš 13 kontakto ir prijunkite į kitą pasirinktą kontaktą (nuo 0 iki 13). Galima naudoti ir analogines jungtis (0-5). Analoginis 0 žymimas kaip 14 ir t.t.

Tuomet pakeičiama kodo eilutė:

`int ledPin = 13;` Keičiame į : `-> int ledPin = newpin;`

Tada įkeliame kodą: (ctrl-v)

Keičiame diodo mirksėjimo laiką: Nepatinka vienos sekundės įsijungimas-išsijungimas?

Pakeiskite kodo eilutes:

```
digitalWrite(ledPin, HIGH);
delay(time on); //(seconds * 1000)
digitalWrite(ledPin, LOW);
delay(time off); //(seconds * 1000)
```

Šviesumo valdymas

Šalia skaitmeninio (on/off) valdymo „Arduino“ gali valdyti kontaktus analoginiu būdu (pvz. šviesumas). (daugiau apie tai - kituose projektuose). Pabandykime:

Prijunkime LEDą prie 9 kontakto: (pakeiskime ir laidą)

`ledPin = 13;` Keičiame į : `-> int ledPin = 9;`

pakeiskime kodą skliaustuose { }po loop() šia eilute:

```
analogWrite(ledPin, new number);
```

(new number) = bet koks skaičius tarp 0 iki 255. 0 = išjungta, 255 = įjungta, tarpinės reikšmės = skirtingas šviesumas

Gęsimas:

Naudosime dar vieną pridėtą pavyzdinę programą. Eikime į meniu:

File > Examples > Analog > Fading

Tuomet įkelkite į „Arduino“ plokštę ir stebėkite, kaip LED'o šviestukas palengva įsižiebia ir pamažu užgęsta.

{GRND-02}

Sukiojame –
Potenciometrai

Ką darysime

Kartu su skaitmeniniais kontaktais, Arduino taip pat turi 6 kontaktus kurie gali būti panaudoti analoginei įvesčiai. Šios įvestys paima įtampą (nuo 0 iki 5 voltų) ir konvertuoja ja į skatmeninį numerį nuo 0 (0 voltų) iki 1023 (5 voltų) (10 bitų gebos). Labai naudingas prietaisas, skirtas išnaudoti šias įvestis yra potenciometras (dar vadinamas kintamo dydžio rezistoriumi). Kai jis prijungiamas prie 5 voltų per išorinius kontaktus, vidurinysis kontaktas nuskaito vertę nuo 0 iki 5 voltų, priklausomai nuo kampo, kuriuo jis yra pasuktas (pavyzdžiui, 2,5 voltų viduryje). Mes taipogi galime naudoti grąžinamąsias vertes kaip kintamąjį mūsų programoje.

Grandinės dalys

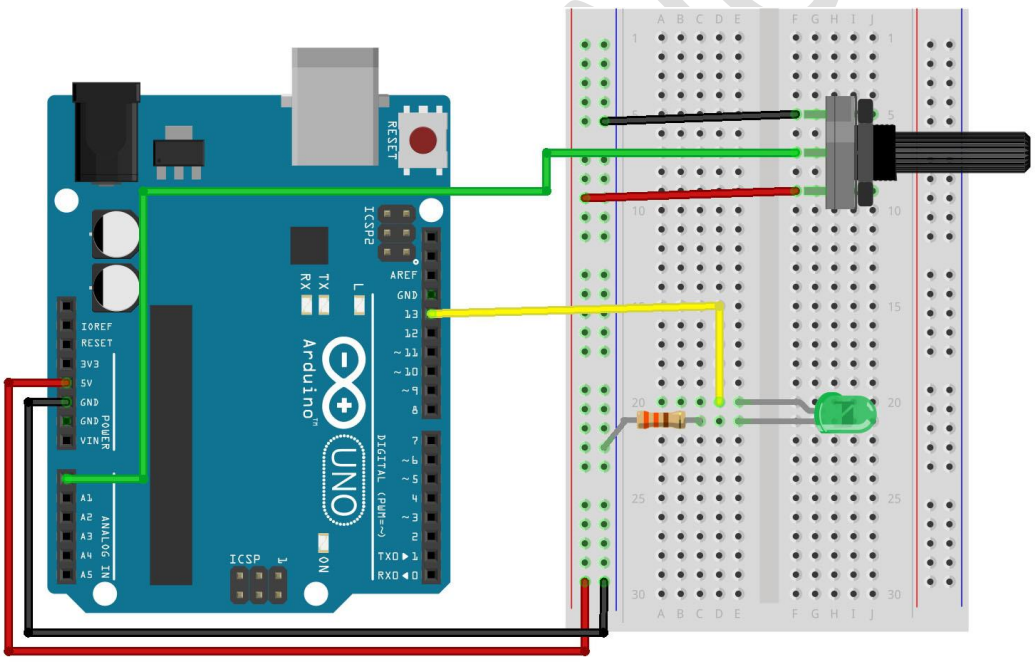
330 Ω rezistorius
(oranžinis-oranžinis-rudas)
x1

10 kΩ potenciometras
x1

5mm viesos diodas
x1

Laidai
X6

Surinkta grandinė



Potenciometras:

Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-02: Sukiojame - Potenciometrai                      |
 *     ------------------------------------------------------------
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 3: Driving and RGB LED
 *
 * Potenciometru kontroliuojame šviesos diodo (LED) mirksėjimo dažnį. Pasukime rankenėlę
 * norėdami, kad LED mirksėtų lėčiau arba greičiau.
 *
 * Analoginė įvestis
 * Parodo analoginį įėjimą skaitant analoginį daviklį iš analoginio pin0 kaisčio ir
 * įjungiant ir išjungiant šviesą spinduliuojantį šviesos diodą (LED), prijungtą prie
 * skaitmeninio pin13 kaisčio.
 * Kiek laiko LED bus įjungtas ir išjungtas priklauso nuo jo
 * vertės, gautos iš  analogRead ()
 *
 * Potenciometrą prijungiame prie analoginės įvesties pin0 kaisčio.
 * Ašis į analoginį pin potenciometrą
 * Viena pusė pin (arba vienas) su korpusu
 * Kitos pusės pin 5 V
 * LED anodą (ilgesnė kojelė), prijungiame prie skaitmeninio pin13 kaisčio
 * LED katodą (trumpesnė kojelė), prijungiame prie žemės GND kaisčio
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 */

int sensorPin = 0;    // priskiriame potenciometr įvesties kaištį pin0
int ledPin = 13;      // priskiriame šviesos diodui kaištį pin13
int sensorValue = 0;  // potenciometras yra nustatomas kaip įtampos daliklis, kad jį
sukiojant, viduriniojo kaiščio pin0 įtampa svyruos nuo 0V iki 5V. Mes priskyrėme potenciometro
vidurinį kaištį Arduino analoginiam įėjimui 0.

void setup() // ši funkcija paleidžiama kartą, kai pradedamas kodo nuskaitymas.
{
  // inicijuojame šviesos diodo (LED) kaištį(pin), kaip išvestį (OUTPUT):
  pinMode(ledPin, OUTPUT);
}

void loop() // ši funkcija kartojama kol setup() pasibaigia:
{
  // nuskaitoma potenciometro viduriniojo kaiščio vertė:
  sensorValue = analogRead(sensorPin);
  // įjungiamas šviesos diodas ledPin
  digitalWrite(ledPin, HIGH);
  // programa sustabdoma kelioms milisekundėmis:
  delay(sensorValue);
  // išjungiamas šviesos diodas ledPin :
  digitalWrite(ledPin, LOW);
  // programa sustabdoma kelioms milisekundėmis:
  delay(sensorValue);
}
```

Neveikia? (Trys dalykai bandymui)

Dalinai veikia	**Neveikia**	**Vis dar neveikia**
Tikėtina, kad tai yra dėl prasto potenciometro kojelių sujungimo. Tai gali būti pataisyta pastumiant potenciometrą žemyn.	Įsitikinkite, kad jūs netyčia nesujungėte potenciometro kojelės prie skaitmeninio pin 2, o ne su analoginiu pin 2. (eilė jungčių po maitinimo jungtimis)	Jūs galite pabandyti apsukti schemą ir ją perrinkti.Tai kartais padeda.

Padaryti geriau?

Perjungimas:
Kartais jūs norite perjungti išvestį (output), kai vertė viršija tam tikrą ribą. Norėdami tai padaryti su potenciometru keisti `loop ()` kodą.

```
void loop() {
  int threshold = 512;
  if(analogRead(potPin) > threshold){ digitalWrite(ledPin, HIGH);}
  else{ digitalWrite(ledPin, LOW);}
}
```

Tai sukels LED įjungti, kai vertė viršija 512 (apie pusiaukelėje), galite reguliuoti jautrumą keičiant ribinę vertę.

Blukimas:
Pakontroliuokime LED šviesumą tiesiai iš potenciometro. Norėdami tai padaryti, mums pirmiausia reikia pakeisti diodo jungtį (pin). Perkelti vielą iš kaiščio (pin) 13 į jungtį (pin) 9 ir pakeisti vieną kodo eilutę.

```
int ledPin = 13; ----> int ledPin = 9;
```

Tada pakeičiame `loop` kodą į

```
void loop() {
  int value = analogRead(potPin) / 4;
  analogWrite(ledPin, value);
}
```

Įkelkite kodą ir stebėkite, kaip bluks diodas reguliuojant potenciometrą. (Pastaba: priežastis, kodėl mes daliname reikšmę iš 4 yra todėl, analogread funkcija grąžina reikšmė nuo 0 iki 1024 (10 bitų) ir analogread užima vertę nuo 0 iki 255 (8 bitai)

Servo varikliuko valdymas:
Tai yra tvarkingas pavyzdys, kuris apima kelias sujungtas grandines. Sujunkite servo mechanizmą taip, kaip jūs sujungėte CIRC-04, tada atverkite pavyzdinę programėlę Knob (File > Examples > Library-Servo > Knob), ir pakeiskite 1 kodo eilutę.

```
int potpin = 0; ----> int potpin = 2;
```

Kodą įkelkite į ARDUINO ir stebėkite, kaip Servo varikliuko velenas sukasi, kai jūs pasukate potenciometrą.

{GRND-03}

Spalvota šviesa –
RGB šviesos diodai

Ką darysime

Pradėjus su pirmuoju projektu CIRC01, džiaugiamės gavę mirksinčią lemputę. Bet tai jau praeita, taip? Tu nori oranžinės, žydros! Mūsų laimei, yra būdas matyti skirtas šviesas iš vieno LEDo, nekeičiant jo skirtingo atspalvio lemputėmis. Tam mes naudojame RGB LEDą. RGB LED nėra atskiras LEDas, iš tikrųjų tai trys, vienas šalia kito esantys LEDai: vienas raudonas, vienas žalias ir vienas mėlynas. Kai juos įjungiame, šios šviesos maišosi – taip gaunami kitas spalvas. Šviesa, kurią gauni, priklauso nuo tam tikros raudono, žalio ir mėlyno LEDų intensyvumo. Intensyvumą reguliuojame su Pulse Width Modulation (PWM), kurį jau naudojome prieš tai, LED ryškumui ir motoriniam greičiui reguliuoti.

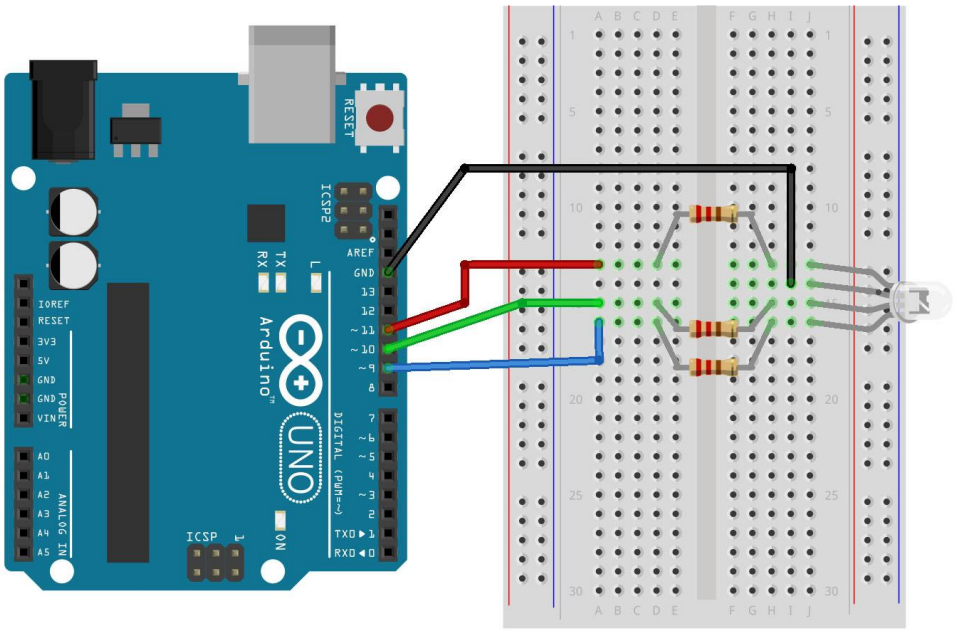
Grandinės dalys

330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x3

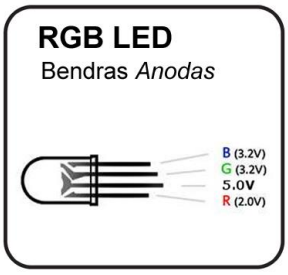
Jungtis (laidas)
X4

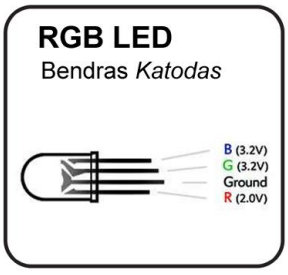
5 mm RGB šviesos diodas
x1

Surinkta grandinė



RGB šviesos diodas:





Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-03: Spalvota šviesa - RGB šviesos diodai            |
 *     ------------------------------------------------------------
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 3: Driving and RGB LED
 *
 * RGB tipo šviesos diodas (RGB LED)skleis besikeičiančias spalvas.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 */

// RGB šviesos diodo kontaktai prijungti prie PWM kaiščių (pin)
const int RED_LED_PIN = 11;
const int GREEN_LED_PIN = 10;
const int BLUE_LED_PIN = 9;

// Naudojamas konkretaus LEDo spalvos intensyvumo lygio saugojimui
int redIntensity = 0;
int greenIntensity = 0;
int blueIntensity = 0;

// šia funkcija nustatome švietimo laiko tarpą (ciklą) milisekundėmis
const int DISPLAY_TIME = 100;


void setup() {
  // nustatymai nereikalaujami.
}

void loop() {
  // nustatome spalvų ciklą iš raudonos į žalią
  // (Šiame cikle mes keičiam 100% raudoną, 0% žalią į 0% raudoną, 100% žalią)
  for (greenIntensity = 0; greenIntensity <= 255; greenIntensity+=5) {
        redIntensity = 255-greenIntensity;
        analogWrite(GREEN_LED_PIN, greenIntensity);
        analogWrite(RED_LED_PIN, redIntensity);
        delay(DISPLAY_TIME);
  }

  // nustatome spalvų ciklą iš žalios į mėlyną
  // (Šiame cikle mes keičiam 100% žalią, 0% mėlyną į 0% žalią, 100% mėlyn¹)
  for (blueIntensity = 0; blueIntensity <= 255; blueIntensity+=5) {
        greenIntensity = 255-blueIntensity;
        analogWrite(BLUE_LED_PIN, blueIntensity);
        analogWrite(GREEN_LED_PIN, greenIntensity);
        delay(DISPLAY_TIME);
  }

  // nustatome spalvų ciklą iš mėlynos į raudoną
  // (Šiame cikle mes keičiam 100% mėlyną, 0% raudoną į 0% mėlyną, 100% raudoną)

  for (redIntensity = 0; redIntensity <= 255; redIntensity+=5) {
        blueIntensity = 255-redIntensity;
        analogWrite(RED_LED_PIN, redIntensity);
        analogWrite(BLUE_LED_PIN, blueIntensity);
        delay(DISPLAY_TIME);
  }
}
```

Neveikia? (Trys dalykai bandymui)

LED nešviečia arba šviečia neteisingomis spalvomis	Matome raudoną	Ieškome daugiau?

Dėl keturių LEDo pinų, kurie yra taip arti vienas kito, kartais visai nesunku tai sumaišyti. Pabandyk dar kartą patikrinti ar kiekvienas pinas yra ten, kur ir turėtų būti.	RGB LEDo raudonas diodas gali būti šiek tiek ryškesnis nei kiti du. Norint labiau subalansuoti spalvas, naudokite aukštesnį ohm rezistorių arba reguliuokite raudonos vertę kode.
	`analogWrite(RED_LED_PIN, redIntensity);`
	to
	`analogWrite(RED_LED_PIN, redIntensity/3);`
		Jeigu ieškai daugiau, ką nuveikti, kodėl gi neišbandžius visų papildomų pasiūlymų *https://learn.sparkfun.com/tutorials/tags/arduino*

Padaryti geriau?

Atsisiųsti kodą: http://ardx.org/RGBMB

{GRND-04}

Pramogaujame –
Daugiau šviesos diodų

Ką darysime

Mes privertėme vieną LED mirksėti, now laikas padidinti riziką. Sujunkime 8. Taip pat turėsime galimybę išplėsti Arduino truputį sukurdami įvairius šviesos tęsinius. Ši grandinė taip pat puikiai tinka išbandyti savo programas ir pajusti kaip Arduino veikia. Taip pat prie LED kontroliavimo mes pradedame žiūrėti į kelis paprastus programavimo metodus, kad išlaikyti programas mažas.

`for() loops` – kai norima paleisti koda kelis kartus.
`arrays[]` – naudojama norint valdyti kintamuosius(grupė kintamųjų).

Grandinės dalys

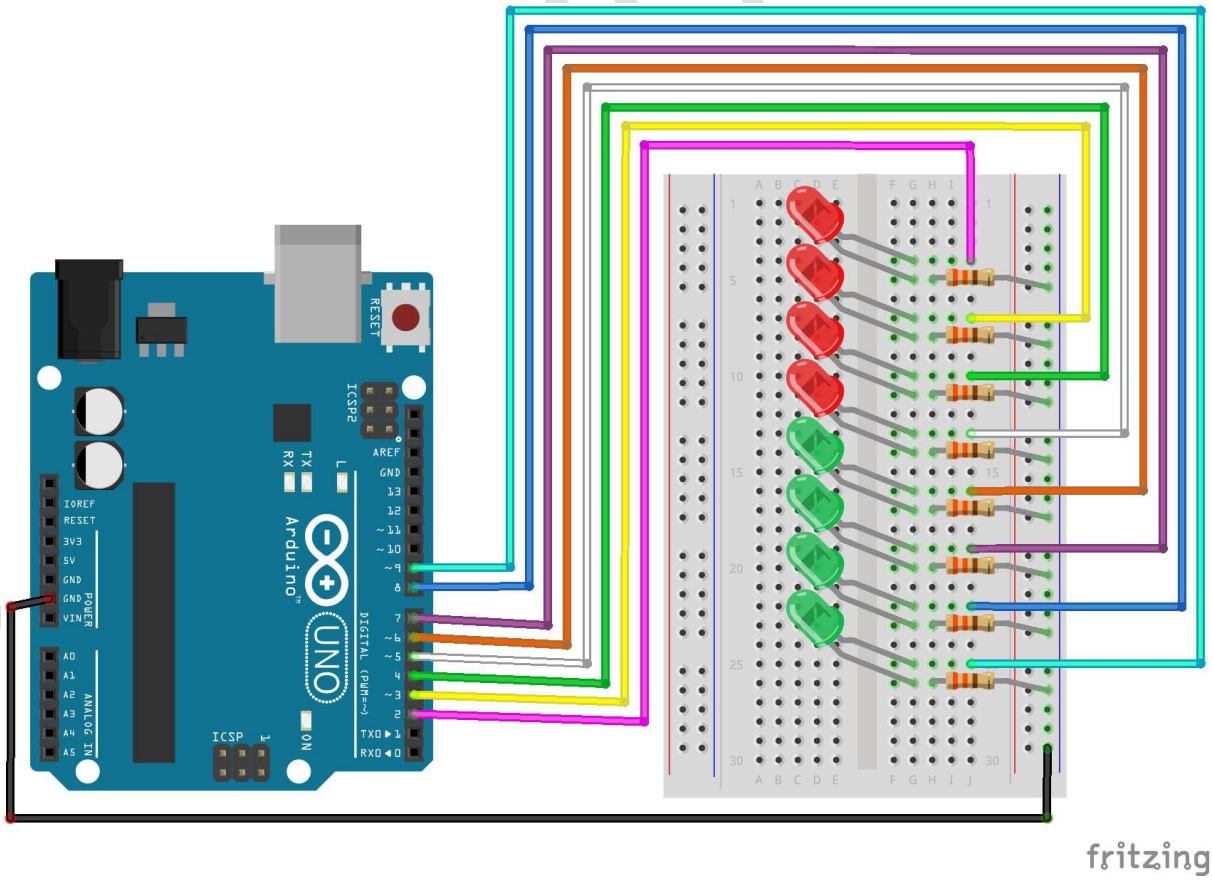
330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x8

Jungiamieji laidai
x9

5mm raudoni šviesos diodai
x4

5mm žali šviesos diodai
x4

Surinkta grandinė



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-04: Pramogaujame - Daugiau šviesos diodų            |
 *     ------------------------------------------------------------
 *
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 02: 8 LED Fun - Multiple LEDs
 *
 * Sukursime keletą paprastų LED animacijų
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 * Daugiau apie šią grandinę žiūrėkite http://tinyurl.com/d2hrud
 *
 */

  // inicijuojami LED Pin kintamieji
int ledPins[] = {2,3,4,5,6,7,8,9};
  // Masyvas, kuriame laikomi PIN nr. prie kurių prijungti LEDai
  // pvz. LED 8 prijungtas prie PIN 2, LED 1 prie PIN 3 ir t.t.
  // jei norime kreiptis į masyvą rašome ledPins[0], kas duotų ezultatą "2",
  // o ledPins[7] reikštų "9"

 /*
 * setup() - ši funkcija paleidžiama kartą kai Arduino yra įjungiamas
 */
void setup()
{

  // Padarome, kad kiekvienas PIN prijungtas prie LED būtų OUTPUT rėžime,
  // junginėsime HIGH arba LOW
  for(int i = 0; i < 8; i++){         //ciklas, kuris bus pakartotas 8 kartus
      pinMode(ledPins[i],OUTPUT);     //nustatome kiekvieną PIN į OUTPUT rėžimą
  }                                   //alternatyvus kodas žemiau

  /* komentuotas kodas nebus paleistas
   * šios eilutės padarytų tą patį, ką aukščiau parašytas ciklas.
  pinMode(ledPins[0],OUTPUT);
  pinMode(ledPins[1],OUTPUT);
  pinMode(ledPins[2],OUTPUT);
  pinMode(ledPins[3],OUTPUT);
  pinMode(ledPins[4],OUTPUT);
  pinMode(ledPins[5],OUTPUT);
  pinMode(ledPins[6],OUTPUT);
  pinMode(ledPins[7],OUTPUT);
  (užkomentuoto kodo pabaiga)*/
}
/*
 * loop() - ši funkcija pasileis po setup() ir kartosis
 */
void loop()                   // kartos vėl ir vėl
{
  oneAfterAnotherNoLoop();    // kiekvieną LED po vieną įjungs ir išjungs
}
/*
 * Įjungs vieną LED tada palauks kažkiek laiko (delayTime), po to įjungs kitą LED
 * ir t.t. kol visi LED bus įjungti. Tada tokia pat tvarka išjunginės vieną po kito.
 * Tai padaroma be ciklo funkcijos, todėl programoje yra daug eilučių
 */
void oneAfterAnotherNoLoop(){
  int delayTime = 100;       // pauzė (milisekundėmis) tarp LED
                             // jeigu norite pakeisti greitį, padidinkite arba pamažinkite šį
                             // skaičių.
  digitalWrite(ledPins[0], HIGH);  // Įjungia LED 0 (prijungtas prie PIN 2)
  delay(delayTime);                // laukia delayTime milisekundžių
  digitalWrite(ledPins[1], HIGH);  // Įjungia LED 1 (prijungtas prie PIN 3)
  delay(delayTime);                // laukia delayTime milisekundžių
  digitalWrite(ledPins[2], HIGH);  // Įjungia LED 2 (prijungtas prie PIN 4)
```

```
  delay(delayTime);                 // Laukia delayTime milisekundžių
  digitalWrite(ledPins[3], HIGH);   // Įjungia LED 3 (prijungtas prie PIN 5)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[4], HIGH);   // Įjungia LED 4 (prijungtas prie PIN 6)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[5], HIGH);   // Įjungia LED 5 (prijungtas prie PIN 7)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[6], HIGH);   // Įjungia LED 6 (prijungtas prie PIN 8)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[7], HIGH);   // Įjungia LED 7 (prijungtas prie PIN 9)
  delay(delayTime);                 // laukia delayTime milisekundžių

//Išjungia kiekvieną LED
  digitalWrite(ledPins[7], LOW);    // Išjungia LED 7 (prijungtas prie PIN 9)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[6], LOW);    // Išjungia LED 6 (prijungtas prie PIN 8)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[5], LOW);    // Išjungia LED 5 (prijungtas prie PIN 7)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[4], LOW);    // Išjungia LED 4 (prijungtas prie PIN 6)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[3], LOW);    // Išjungia LED 3 (prijungtas prie PIN 5)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[2], LOW);    // Išjungia LED 2 (prijungtas prie PIN 4)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[1], LOW);    // Išjungia LED 1 (prijungtas prie PIN 3)
  delay(delayTime);                 // laukia delayTime milisekundžių
  digitalWrite(ledPins[0], LOW);    // Išjungia LED 0 (prijungtas prie PIN 2)
  delay(delayTime);                 // laukia delayTime milisekundžių
}
```

Neveikia? (Trys dalykai bandymui)

Kai kurie LED neužsidega

Lengva įstatyti LED ne ta puse. Patikrinkite LED kurie neveikia ir užtikrinkite, kad jie idėti reikiama puse.

Laidų klaida

Jungiant 8 laidus lengva susipainioti. Dar karta patikrinkite ar pirmas LED yra ikistas į 2 ir kiekvienas kaištis taip pat.

Pradėti iš naujo

Lengva ne ten idėti laida kur reikia. Viska ištraukti ir pradėti iš naujo dažniausiai yra lengviausias būdas pataisyti klaidą.

Padaryti geriau?

Pakeisti `Loop`:

`loop()` yra keturios eilutės. Paskutinės 3 prasideda su '//' tai reiškia, kad su eilute yra elgiamasi kaip su komentaru (neveikti). Pakeisti programą, kad veiktu eilutė `void loop()` kodu:

```
//oneAfterAnotherNoLoop();
oneAfterAnotherLoop();
//oneOnAtATime();
//inAndOut();
```

Į loop() įrašome eilutes :

```
oneAfterAnotherLoop();
//oneOnAtATime();
//inAndOut();
```

Programos pabaigoje įrašome funkcijas:

```
/*
 * oneAfterAnotherLoop() - Daro tą patį, ką oneAfterAnotherNoLoop() tik kad
 * su žymiai mažiau rašymo, nes yra naudojamas ciklas
 */
void oneAfterAnotherLoop(){
  int delayTime = 100; //pauzė (milisekundėmis) tarp LED
                       //jeigu norite pakeisti greitį, padidinkite arba pamažinkite
```

```
šį skaičių.
//Turn Each LED on one after another
  for(int i = 0; i <= 7; i++){
    digitalWrite(ledPins[i], HIGH);  //Įjungia LED nr.i ir kartosis. Prie i
    delay(delayTime);                //bus vis pridedama po vieną ir kartosis
  }                                  //8 kartus. Pirmą kartą i=0, o paskutinį i=8

                                     //Išjungia LED vieną po kito
  for(int i = 7; i >= 0; i--){       //Tas pats kas aukščiau, tik pradeda ne nuo 0,
                                     //o nuo 7 ir i mažėja po vienetą.
    digitalWrite(ledPins[i], LOW);   //Išjungia LED #i kiekvieną pakartojimą.
    delay(delayTime);                //Kiekvieną pakartojimą iš i atimamas vienetas.
  }                                  //Pirmą kartą i=7, paskutinį - i=0

}

/*
 * oneOnAtATime() - Uždega vieną LED, tada kitą ir išjungia visus kitus.
 */
void oneOnAtATime(){
  int delayTime = 100;   //pauzė (milisekundėmis) tarp LED
                         //jeigu norite pakeisti greitį, padidinkite arba pamažinkite
šį skaičių.

  for(int i = 0; i <= 7; i++){
    int offLED = i - 1;  //Calculate which LED was turned on last time through
    if(i == 0) {         //for i = 1 to 7 this is i minus 1 (i.e. if i = 2 we will
      offLED = 7;        //turn on LED 2 and off LED 1)
    }                    //Kai i = 0 nenorime išjungti LED -1(nes neegzistuoja)
// vietoj to išjungiam LED 7

    digitalWrite(ledPins[i], HIGH);     //įjungia LED nr.i
    digitalWrite(ledPins[offLED], LOW); //išjungia paskutinį kartą įjungtą LED
    delay(delayTime);
  }
}
```

cc

```
/*
 * inAndOut() - Įjungia du vidurinius LED, tada du jiems iš šonų ir t.t
 */
void inAndOut(){
  int delayTime = 100; //pauzė (milisekundėmis) tarp LED
                       //jeigu norite pakeisti greitį, padidinkite arba pamažinkite
šį skaičių.
  for(int i = 0; i <= 3; i++){
    int offLED = i - 1;  //Suskaičiuojame kuris LED buvo įjungtas
                         //paskutinį kartą per ciklą
    if(i == 0) {         //for i = 1 to 7 this is i minus 1 (i.e. jei i = 2 įjungsime
      offLED = 3;        //LED 2 ir išjungsime LED 1)
    }                    //Kai i = 0 nenorime išjungti LED -1(nes neegzistuoja)
                         //vietoj to išjungiam LED 0
    int onLED1 = 3 - i;       // pirmas LED nuo kurio prasidės pvz. LED 3 kai i = 0
ir LED
                     //
                              //0 kai i = 3
    int onLED2 = 4 + i;       // pirmas LED nuo kurio prasidės pvz. LED 4 kai i = 0
ir LED
                     //
                              //7 kai i = 3
    int offLED1 = 3 - offLED; //Išjungia LED, kurį įjungėme praeitą kartą
    int offLED2 = 4 + offLED; //Išjungia LED, kurį įjungėme praeitą kartą

    digitalWrite(ledPins[onLED1], HIGH);
    digitalWrite(ledPins[onLED2], HIGH);
    digitalWrite(ledPins[offLED1], LOW);
    digitalWrite(ledPins[offLED2], LOW);
```

```
    delay(delayTime);
  }
  for(int i = 3; i >= 0; i--){
    int offLED = i + 1;       //Suskaičiuojame kuris LED buvo įjungtas
                              //paskutinį kartą per ciklą
    if(i == 3) {            //for i = 1 to 7 this is i minus 1 (pvz. if i = 2 we will
      offLED = 0;             //turn on LED 2 ir išjungsime LED 1)
    }                         //vietoj to išjungiam LED 0
    int onLED1 = 3 - i;       //pirmas LED nuo kurio prasidės pvz. LED 4 kai i = 0 ir
LED
                              //0 kai i = 3
    int onLED2 = 4 + i;       //pirmas LED nuo kurio prasidės pvz. LED 4 kai i = 0 ir
LED
                              //7 kai i = 3
    int offLED1 = 3 - offLED; //Išjungia LED, kurį įjungėme praeitą kartą
    int offLED2 = 4 + offLED; //Išjungia LED, kurį įjungėme praeitą kartą

    digitalWrite(ledPins[onLED1], HIGH);
    digitalWrite(ledPins[onLED2], HIGH);
    digitalWrite(ledPins[offLED1], LOW);
    digitalWrite(ledPins[offLED2], LOW);
    delay(delayTime);
  }
}
```

Įkėlę programą pastebime, kad niekas nepasikeitė. Galite pažiūrėti į dvi funkcijas, kiekveina kažką daro, bet naudoja skirtingus priėjimus.

Kitos animacijos:
Pavargote nuo tos pačios animacijos? Tada pabandykite kitas dvi animacijas. Nukomentuokite jų eilutes ir įkelkite program į lentelę ir megaukitės nauja animacija. (ištrinkite pasvyruosius brukšnius prieš 3 ir 4 eilutes)

Savo paties sukurtos animacijos bandymas:
Pažvelkite į įtrauktus kodus ir pradėkite keisti dalykus. Pagrindinis tikslas yra panaudoti LED darbe, naudokite:

```
digitalWrite(pinNumber, HIGH);
```

Tada jį išjunkite naudodami;

```
digitalWrite(pinNumber, LOW);
```

Nebijokite viską keisti, nesvarbu ką parašysite, nieko nesugadinsite.

{GRND-05}

Spaudžiame mygtukus – Mygtukai

Ką darysime

Iki šiol mes buvome susitelkę į išvestis, laikas išmokyti „Arduino“ klausyti mūsų. Pradėsime nuo paprasto paspaudžiamojo mygtuko. Jo prijungimas yra gana paprastas. Grandinėje tėra vienas papildomas komponentas – rezistorius, kuris gali atrodyti esąs ne savo vietoje. Jis naudojamas tam, kad „Arduino“ nejaučia ar mygtukas yra paspaustas, ar ne. Vietoje jautimo, Arduino nuskaito kaiščio (pin) įtampą ir nusprendžia ar ji aukšta (HIGH), ar žema (LOW). Kai mygtuką nuspaudžiame, Arduino jį nuskaitys kaip LOW. Tačiau, kada mygtukas yra nenuspaustas, kaiščio (pin) įtampa tarsi „plūduriuoja“ , kas sukelia atsitiktinių nuskaitymo klaidų. Kad „Arduino“ patikimai nuskaitytų kaiščio (pin) įtampą kaip aukštą (HIGH), mygtukui esant nenuspaustam, mes prijungiame rezistorių. (Pastaba: pirmajame pavyzdyje naudojamas tiktais vienas mygtukas iš dviejų).

Grandinės dalys

330 Ω rezistorius
(oranžinis-oranžinis-rudas)
x1

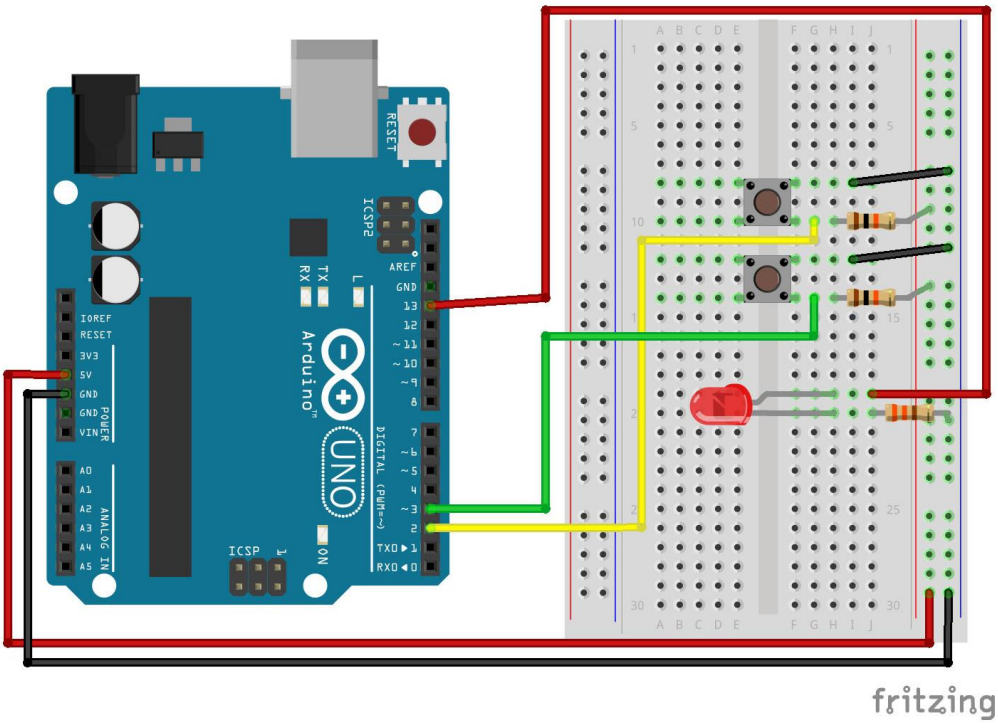
10 kΩ rezistorius
(rudas-juodas-oranžinis)
x2

Jungiamieji laidai
X7

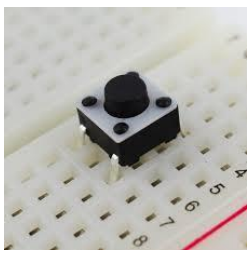
5mm šviesos diodas
x1

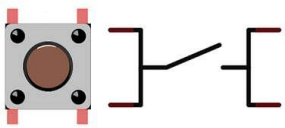
Mygtukai
x2

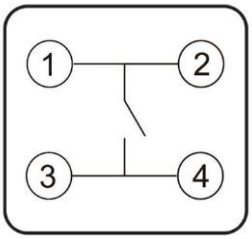
Surinkta grandinė



Mygtukas:







Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:      |
 *     |  GRND-05: Spaudžiame mygtukus - Mygtukai        |
 *     ------------------------------------------------------------
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 5: Push Buttons
 *
 * Įjungsime ir išjungsime šviesos diodą (LED), prijungtą pin13, kai nuspausime
 * mygtuką prijungtą pin7.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 */

  // konstantos nekis, jos naudojamos pakeisti kaiščių (pin) „rodmenis" :
const int button1Pin = 2;  // pirmo mygtuko kaištis (pin)
const int button2Pin = 3;  // antro mygtuko kaištis (pin)
const int ledPin =  13;    // LED kaištis (pin)

  // kintamieji pasikeis:
int buttonState = 0;       // kintamasis spaudžiamajo mygtuko nuskaitymui ir būsenai.

void setup() {
  // inicijuoja šviesos diodo (LED) kaištį(pin), kaip išvestį (OUTPUT):
  pinMode(ledPin, OUTPUT);
  // inicijuoja šviesos diodo (LED) kaištį(pin) kaip įvestį (INPUT):
  pinMode(buttonPin, INPUT);
}

void loop(){
  // nuskaito spaudžiamojo mygtuko vertės būklę:
  buttonState = digitalRead(buttonPin);

  // patinkrina ar spaudžiamas mygtukas yra nuspaustas.
  // jeigu jis nuspaustas, spaudžiamo mygtuko būsena yra aukšta(HIGH):
  if (buttonState == HIGH) {
  // įjungia šviesos diodą (LED):
  digitalWrite(ledPin, HIGH);
  }
  else {
  // kitu atveju išjungia šviesos diodą (LED):
    digitalWrite(ledPin, LOW);
  }
}
```

Mygtukas

Spaudinėdami mygtuką prijungtą prie 7 kaiščio (pin 7), įjungiame ir išjungiame šviečiantį LED'ą, prijungtą prie 13 skaitmeninio kaiščio (pin13).

Grandinė:
* Šviesos diodas (LED) prijungtas iš 13 kaiščio (pin13) į neigiamą šaltinio polių
* Mygtukas – kurį spausime, pridėtas prie antrojo kaiščio(pin2) iš (+5V)
* 10K rezistorius pridėtas prie 2 kaiščio(pin2) ir eina iš neigiamo šaltinio poliaus(įžeminimo)

Pastaba: daugelyje „Arduino" schemų jau yra ledas (LED) ant plokštės prijungtas į 13 kaištį (pin_13).

http://www.arduino.cc/en/Tutorial/Button

Neveikia? (Trys dalykai bandymui)

Šviesa neįsijungia
Spaudžiamasis mygtukas yra keturkampis ir dėl šios priežasties yra lengva jį įstatyti ne ta puse. Perstatykite spaudžiamąjį mygtuką 90 laipsnų kampu ir pažiūrėkite ar grandinė veikia.

Šviesa negęsta (neblanksta)

Kvailoka klaida, kurią visi kartais padarome, kada pereiname nuo šviesos įjungimo iki jos blankimo, nepamirškite perdėti ledo(LED) laidą iš 13 kaiščio(pin 13) į 9 kaištį (pin 9).

Underwhelmed?

Nesijaudinkite, šios grandinės yra sujungtos paprastai, kad būtų lengva žaisti su komponentais, bet kai jūs sumetate juos kartu, dangus yra riba.

Padaryti geriau?

Įjungimo mygtukas išjungimo mygtukas:
Sunkesnis bet irigi įdomus pavyzdys, vienas mygtukas įjungs LED'ą, kitas – išjungs. Pakeiskite kodą į :

```
int ledPin = 13;                  // priskiria kaištį (pin) ledui (LED)
int inputPin1 = 3;                // mygtukas 1
int inputPin2 = 2;                // mygtukas 2

void setup() {
  pinMode(ledPin, OUTPUT);        // priskiria LED kaip išvadą (output)
  pinMode(inputPin1, INPUT);      // padaro 1 mygtuką kaip įvadą (input)
  pinMode(inputPin2, INPUT);      // padaro 2 mygtuką kaip įvadą (input)
}

void loop(){
  if (digitalRead(inputPin1) == LOW) {
       digitalWrite(ledPin, LOW);  // išjungia ledą (LED)
  }
  else if (digitalRead(inputPin2) == LOW) {
       digitalWrite(ledPin, HIGH); // įjungia ledą (LED)
  }
}
```

Įkelkite kodą į savo programos lauką ir pradėkite perjunginėti LED'ą.

Blukimas stipryn ir silpnyn:
Panaudokime mygtukus, kad galėtume reguliuoti analoginį kodą, Norėdami tai padaryti, jums reikės pakeisti laidą, iš pin 13 LED į pin 9, taip pat pakeisti kodą:

```
int ledPin = 13; ----> int ledPin = 9;
```

Toliau keičiame – `loop()` procedūrą. `// loop` – procedūros pavadinimas.

```
int value = 0;
void loop(){
  if (digitalRead(inputPin1) == LOW) { value--; }
  else if (digitalRead(inputPin2) == LOW) { value++; }
  value = constrain(value, 0, 255);
  analogWrite(ledPin, value);
  delay(10);
}
```

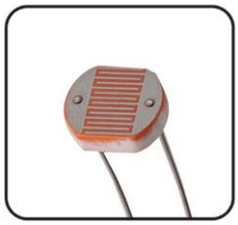
Blukimo grečio keitimas:
Jeigu norėtumėte pakeisti LED'o šviesos blukimo greitį iš didesnio į mažesnį arba atvirkščiai, tėra viena eilutė kode, kurią keičiame;

```
delay(10); ----> delay(naujas #);
```

Kad padarytumėte šviesos blukimą greitesniu, keiskite numerį į dar mažesnį, lėtumas reikalauja didesnio numerio.

{Pro-06}

Šviesa –
Fotorezistoriai



Ką darysime

Nors potenciometras gali būti naudingas žmogaus kontroliuojamiems eksperimentams, ką mes naudosime, kai norėsime padaryti aplinkos įtakojamą eksperimentą? Mes naudojame tiksliai tuos pačius principus, bet vietoj potenciometro, mes naudsime fotorezistorių. Arduino negali tiesiogiai pajusti pasipriešinimo (jis jaučia įtampos pokyčius), todėl mes įsteigti įtampos daliklį. Tiksli įtampa daviklio pin apskaičiuojama, bet mūsų tikslas (tik jutimo santykinis šviesos), mes galime eksperimentuoti su vertėmis, ir pamatyti, kas tinka mums. Žemos vertės bus tada, kai jutiklis yra gerai apšviestas, nors galimos didelės vertės, kai jis yra tamsoje.

Grandinės dalys

330 Ω rezistorius
(oranžinis-oranžinis-rudas)
x1

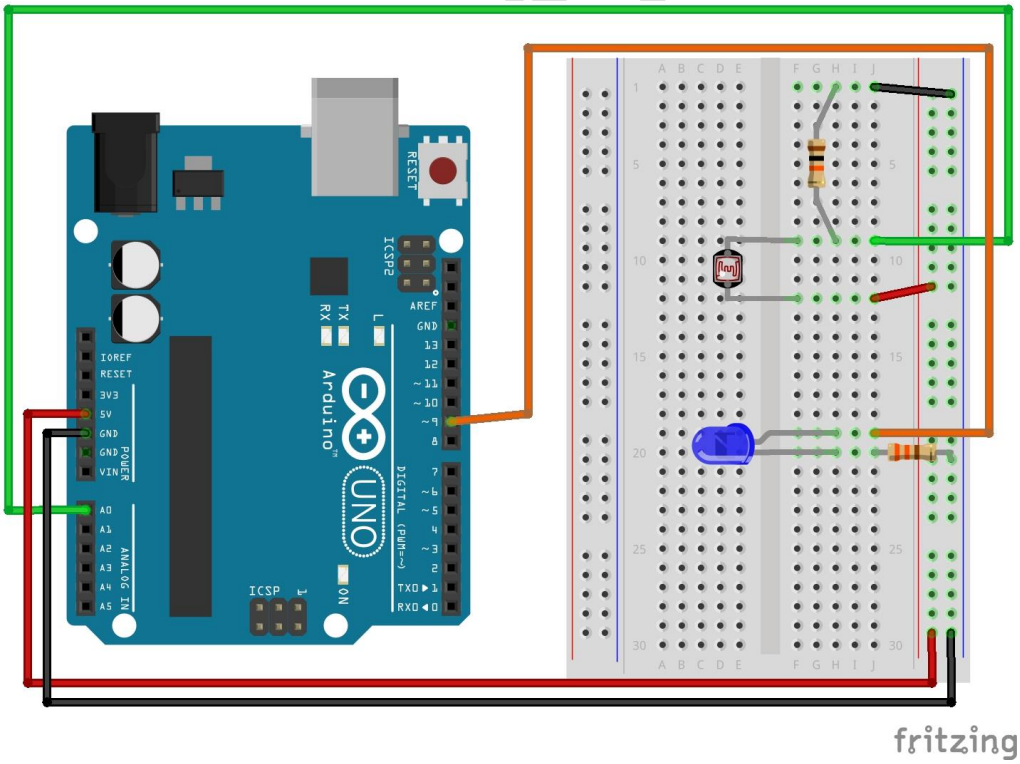
10 kΩ rezistorius
(rudas-juodas-oranžinis)
x1

Jungiamieji laidai
x6

Fotorezistorius
x1

5mm šviesos diodas
x1

Surinkta grandinė



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:      |
 *     |  GRND-06: Šviesa - Fotorezistoriai              |
 *     ------------------------------------------------------------
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 6: Reading a Photo Resistor
 *
 * Naudosime fotorezistorių (šviesos jutiklį), kad galėtumėte kontroliuoti LED šviesumą.
 *
 * Aparatūros jungtys:
 *
 * Fotorezistorius:
 * Vieną fotorezistoriaus pusę sujunkite su 5V (voltais).
 * Sujunkite kitą fotorezistoriaus pusę su ANALOG pin 0.
 * Sujunkite 10000 omų rezistorių tarp ANALOG pin 0 and GND (žemė).
 *
 * Tai sukuria įtampos dalintoją su fotorezistoriumi ir vienu iš rezistorių. Įtampos
 * dalintojo išėjimo įtampa keisis su aspšviestumu.
 *
 * LED:
 * Sujunkite teigiamąją LED pusę (ilgesnioji kojelė) su skaitmeniniu (Digital) pin 9
 * Norint pakeisti šviesumą, ši jungtis (pin) turi būti sujungta su pulso
 * pločio moduliatoriumi (PWM), kuris yra nurodytas simboliu "~" arba "PWM" per Arduino.
 *
 * Sujunkite neigiamąją LED pusę(trumpesnioji kojelė) su 330 omų rezistoriumi. *
 *
 * Kitą rezistoriaus pusę sujunkite su žeme (GND).
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
*/

// Kaip įprasta, mes sukursime pavadinimus jungtims (pins), kurias mes naudojame.
// Dėl to bus lengviau naudotis žemiau pateiktu kodu.

const int sensorPin = 0;
const int ledPin = 9;

// Mes taip pat įvesime bendrų kintamųjų šviesos lygiui.

int lightLevel, high = 0, low = 1023;


void setup()
{
  // Padarysime kad LED kaištis būtų išėjimo.
  // (Neturime daryti nieko ypatingo, norėdami naudotis analogine įėjimo jungtimi).

  pinMode(ledPin, OUTPUT);
}


void loop()
{
  // Lygiai taip, kaip mes padarėme praeityje, mes naudosime analoginio kodo (analogRead)()
  // funkciją išmatuoti įtampą, ateinančią iš fotorezistorių
  // porų. Šis skaičius gali kisti nuo 0 (0 Volts) iki
  // 1023 (5 Volts), bet ši grandinė turės mažesnį tarpą
  // tarp tamsos ir šviesos.

  lightLevel = analogRead(sensorPin);

  // We now want to use this number to control the brightness of
  // the LED. But we have a problem: the analogRead() function
  // returns values between 0 and 1023, and the analogWrite()
  // function wants values from 0 to 255.

  // We can solve this by using two handy functions called map()
  // and constrain(). Map will change one range of values into
```

```
  // another range. If we tell map() our "from" range is 0-1023,
  // and our "to" range is 0-255, map() will squeeze the larger
  // range into the smaller. (It can do this for any two ranges.)

  // lightLevel = map(lightLevel, 0, 1023, 0, 255);

  // Because map() could still return numbers outside the "to"
  // range, (if they're outside the "from" range), we'll also use
  // a function called constrain() that will "clip" numbers into
  // a given range. If the number is above the range, it will reset
  // it to be the highest number in the range. If the number is
  // below the range, it will reset it to the lowest number.
  // If the number is within the range, it will stay the same.

  // lightLevel = constrain(lightLevel, 0, 255);

  // Here's one last thing to think about. The circuit we made
  // won't have a range all the way from 0 to 5 Volts. It will
  // be a smaller range, such as 300 (dark) to 800 (light).
  // If we just pass this number directly to map(), the LED will
  // change brightness, but it will never be completely off or
  // completely on.

  // You can fix this two ways, each of which we'll show
  // in the functions below. Uncomment ONE of them to
  // try it out:

  manualTune();  // manually change the range from light to dark

  //autoTune();  // have the Arduino do the work for us!

  // The above functions will alter lightLevel to be cover the
  // range from full-on to full-off. Now we can adjust the
  // brightness of the LED:

  analogWrite(ledPin, lightLevel);

  // The above statement will brighten the LED along with the
  // light level. To do the opposite, replace "lightLevel" in the
  // above analogWrite() statement with "255-lightLevel".
  // Now you've created a night-light!
}


void manualTune()
{
  // As we mentioned above, the light-sensing circuit we built
  // won't have a range all the way from 0 to 1023. It will likely
  // be more like 300 (dark) to 800 (light). If you run this sketch
  // as-is, the LED won't fully turn off, even in the dark.

  // You can accommodate the reduced range by manually
  // tweaking the "from" range numbers in the map() function.
  // Here we're using the full range of 0 to 1023.
  // Try manually changing this to a smaller range (300 to 800
  // is a good guess), and try it out again. If the LED doesn't
  // go completely out, make the low number larger. If the LED
  // is always too bright, make the high number smaller.

  // Remember you're JUST changing the 0, 1023 in the line below!

  lightLevel = map(lightLevel, 0, 1023, 0, 255);
  lightLevel = constrain(lightLevel, 0, 255);

  // Now we'll return to the main loop(), and send lightLevel
  // to the LED.
}


void autoTune()
{
  // As we mentioned above, the light-sensing circuit we built
  // won't have a range all the way from 0 to 1023. It will likely
  // be more like 300 (dark) to 800 (light).

  // In the manualTune() function above, you need to repeatedly
```

```
  // change the values and try the program again until it works.
  // But why should you have to do that work? You've got a
  // computer in your hands that can figure things out for itself!

  // In this function, the Arduino will keep track of the highest
  // and lowest values that we're reading from analogRead().

  // If you look at the top of the sketch, you'll see that we've
  // initialized "low" to be 1023. We'll save anything we read
  // that's lower than that:

  if (lightLevel < low)
  {
    low = lightLevel;
  }

  // We also initialized "high" to be 0. We'll save anything
  // we read that's higher than that:

  if (lightLevel > high)
  {
    high = lightLevel;
  }

  // Once we have the highest and lowest values, we can stick them
  // directly into the map() function. No manual tweaking needed!

  // One trick we'll do is to add a small offset to low and high,
  // to ensure that the LED is fully-off and fully-on at the limits
  // (otherwise it might flicker a little bit).

  lightLevel = map(lightLevel, low+30, high-30, 0, 255);
  lightLevel = constrain(lightLevel, 0, 255);

  // Now we'll return to the main loop(), and send lightLevel
  // to the LED.
}
```

Neveikia? (Trys dalykai bandymui)

Šviesos diodas nešviečia	Jis nereguoja į šviesos kiekio pakitimus	Vis dar neveikia?
Tai yra klaida, mes ir toliau dėti laiko ir laiko vėl, jei tik jie galėtų LED, kuris veikia į abi puses. Ištraukite jį ir suteikti jai Tvist.	Atsižvelgiant į tai, kad laidai ant nuotraukos-rezistorius tarpai yra ne standartas, tai lengva nudėti ją. Dar kartą patikrinkite savo į tinkamą vietą.	Jums gali būti patalpoje, yra per šviesus arba tamsus. Pabandykite tekinimo šviesas įjungti arba išjungti norėdami pamatyti, jei tai padeda. Arba, jei turite žibintuvėlis netoliese duoti, kad pabandyti.

Padaryti geriau?

Pakeiskite atsaką:
Gal jums prireiks pakeisti atsaką. Galime lengvai tai padaryti pakeisdami:

```
analogWrite(ledPin, lightLevel);
          ----> analogWrite(ledPin, 255 - lightLevel);
```

Upload and watch the response change:

Naktinė lempa:
Rather than controlling the brightness of the LED in response to light, lets instead turn it on or off based on a threshold value. Change the `loop()` code with.

```
void loop(){
  int threshold = 300;
  if(analogRead(lightPin) > threshold){
    digitalWrite(ledPin, HIGH);
```

```
  }else{
    digitalWrite(ledPin, LOW);
  }
}
```

Šviesa kontroliuoja servo variklį:

Lets use our newly found light sensing skills to control a servo (and at the same time engage in a little bit of Arduino code hacking). Wire up a servo connected to pin 9 (like in CIRC-04). Then open the Knob example program (the same one we used in CIRC-08) File > Examples > Library-Servo > Knob. Upload the code to your board and watch as it works unmodified.

Using the full range of your servo:

You'll notice that the servo will only operate over a limited portion of its range. This is because with the voltage dividing circuit we use the voltage on analog pin 0 will not range from 0 to 5 volts but instead between two lesser values (these values will change based on your setup). To fix this play with the val = map(val, 0, 1023, 0, 179); line. For hints on what to do visit http://arduino.cc/en/Reference/Map .

{GRND-07}

Temperatūros matavimas –
Temperatūros jutiklis TMP35

Ką darysime

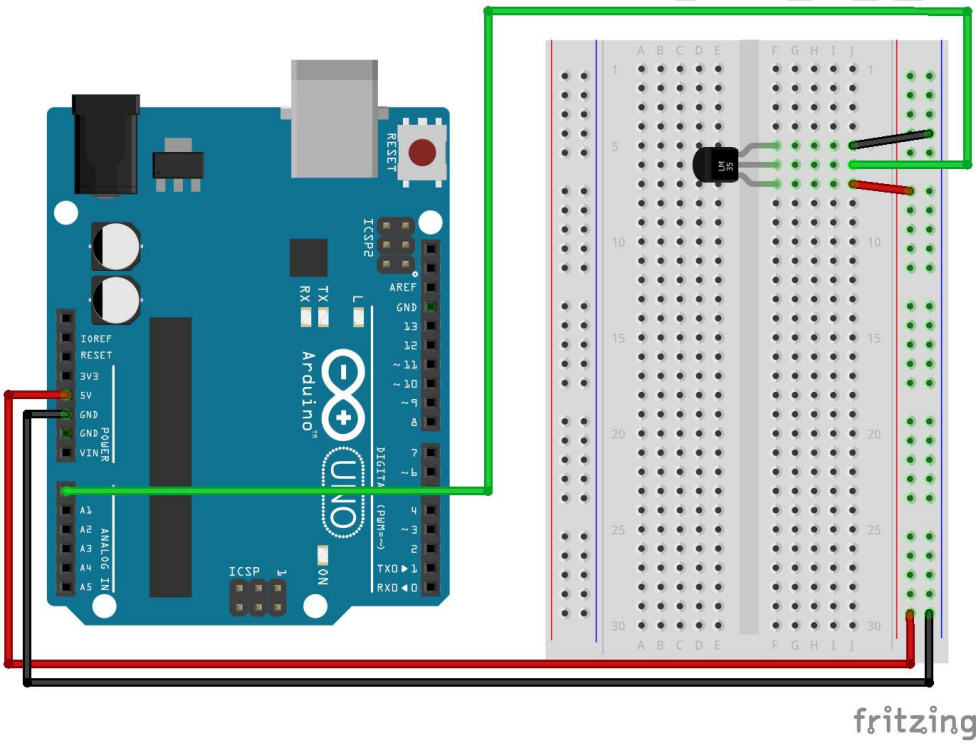
Arduino su matuosime temperatūra! Pasinaudosime jutikliu TMP35, integriniu grandynu, panašiu į tranzistorius. Jis turi tris kontaktus: įžeminimo, signalo ir +5 voltų prijungimo. Jis signalo kontakte teikia 10 mV vienam laipsniui (kad galėtume matuoti temperatūras žemiau nulio, yra numatyta 500 mV atsvara, pvz. 25^0 C = 750 mV, 0^0 C = 500mV). Signalui versti iš skaitmeninės vertės į laipsnius, naudosime keletą Arduino matematinių sugebėjimų. Norėdami pamatyti, naudosime vieną iš IDE ypatybių - derinimo langą. Išvesime duomenų vertę per serijinę jungtį, kad ją rodytų ekrane. **Pastaba:** ši grandinė naudoja Arduino IDE serijinį monitorių. Kad jį atvertume, pirma įkeliame programą, tada spaudžiame mygtuką panašų į kvadratą su antena.
TMP35 charakteristikos: http://ardx.org/datasheet/IC-TMP36.pdf

Grandinės dalys

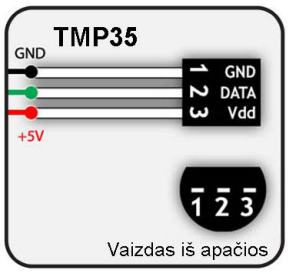
Temperatūros jutiklis TMP35
x1

Jungiamieji laidai
x5

Surinkta grandinė



Temperatūros jutiklis TMP35:



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                      |
 *     |  GRND-07: Temperatūros matavimas - Temperatūros jutiklis TM35   |
 *     ------------------------------------------------------------------
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 7: Reading a Temperature Sensor
 *
 * Naudodami „serijinį langą“ išvesime dabartinę temperatūrą į IDE derinimo langą
 *
 * Arduino IDE turi funkciją, kuri padeda derinant programos kodą ar valdant Arduino
 * kompiuterio klaviatūra. „Serijinis langas“ (Serial Monitor) yra atskiras iššokantis langas,
 * kuris veikia kaip atskiras terminalas, priimantis ir išsiunčiantis serijinius duomenis.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 */

//TMP36 Pin'ų Kintamieji
int temperaturePin = 0; //priskiria analoginį kaištį pin0 jutikliui TMP36
                        //keitimo santykis yra 10 mV / vienam laipsniui
                        //(500 mV atsvara) kad galėtų būti rodoma neigiama temperatūra.

/*
 * setup() - ši funkcija įvykdoma kartą, kai įjungiamas Arduino
 * inicijuojama serijinė jungtis su kompiuteriu
 */
void setup()
{
  Serial.begin(9600);  //inicijuojama serijinė jungtis su kompiuteriu
                       //kad matytume rezultatą, įjugiame „serijinį langą“ - spaudžiame
                       //IDE mygtuką, su pavaizduotu kvadratu su antena.
}

void loop()            //vykdyti visą laiką
{
 float temperature = getVoltage(temperaturePin); //iš temperatūros jutiklio nuskaito
                                                 //įtampos reikšmes
 temperature = (temperature - .5) * 100;         //keičia gautas reikšmes (10 mv
                                                 //laipsnui) su 500 mV atsvara
                                                 //į laipsnius ((įtampa - 500mV) kart 100)
 Serial.println(temperature);                    //išveda rezultatą
 delay(1000);                                    //laukia sekundę
}
/*
 * getVoltage() - sugrįžta prie įtampos reikšmės nuskaitymo iš jutiklio analoginės jungties,
 */
float getVoltage(int pin){
 return (analogRead(pin) * .004882814); //keičia nuo 0 iki 1023 skaitmeninės vertės
                                        //nuo 0 iki 5 voltų (vienas nuskaitymas lygus
                                        //~ 5 milivoltų
}
```

Neveikia? (Trys dalykai bandymui)

Niekas nevyksta
Ši programa gali parodyti, kad veikia. Kad pamatytumėte rezultatus, privalote įjungti Arduino IDE serijinį langą. (instrukcijos praeitame lange)

Rodomos nesąmonės
Tai vyksta, nes serijinis monitorius gauna duomenis greičiu, kurio jis nesitikėjo. Kad tai pataisytumėte, spauskite nuleidžiamą langą rašantį "*** baud" ir pakeiskite jį į "9600 baud".

Temperatūra nesikeičia
Bandykite pirštais sušildyti sensorių arba atšaldyti šaltu oru.

Padaryti geriau?

Rodyti įtampą:
Reikia pakeisti tik vieną eilutę. Sensorius išveda 10mv per šimtalaipsnį, kad gautume įtampą, keičiame išvedamą rezultatą į `getVoltage()`. Ištriname eilutę

```
temperature = (temperature - .5) * 100;
```

Kad rodytų farenheitus:
Tai paprastas pakeitimas, pasitelkiant matematiką, kad pakeistume Celsijus į Farenheitus, naudojime formulę:

$$(F = C * 1.8) + 32)$$

Pridėkite eilutę

```
temperature = (((temperature - .5) * 100)*1.8) + 32;
before Serial.println(temperature);
```

Daugiau informacijos:
Padarykime, kad būtų rodoma daugiau duomenų. Pirma grąžinkime prie originalaus kodo tada pakeiskime:

```
Serial.println(temperature); ----> Serial.print(temperature);
Serial.println(" degrees centigrade");
```

Pakeitimas pirmoje eilutėje reiškia, kad sekantis išvedimas pasirodys toje pačioje eilutėje, tada pridedame informacinį tekstą ir naują eilutę.

Keičiame serijinį greitį:
Jei norite išvesti daug duomenų greitai. Dabar išvedame greičiu 9600 baud, bet įmanoma greičiau. Kad tai pakeistume, keičiame šią eilutę:

```
Serial.begin(9600); ----> Serial.begin(115200);
```

Keiskite 9600 baud į 115200 baud programavimo lange. Dabar duomenis gaunate 12 kartų greičiau.

{GRND-08}

Judesio kontrolė –
Servo mechanizmai

Ką darysime

Darant projektus, kartais reikalinga tiksli judesio kontrolė. Tam naudojami servo mechanizmai, arba tiesiog – "servas". Jie yra masiškai gaminami ir plačiai aptinkami modeliavimo reikmenų parduotuvėse. Kainuoja nuo kelių iki kelių šimtų eurų. Jų viduje aptiksime nedidelę pavarų dežę judesio sustiprinimui ir šiek tiek elektronikos valdymui. Standartinės mechanizmo posūkio padėys gali būti nuo 0 iki 180^0. Posūkio padėties kitimas yra valdomas per impulsą, pasikartojantį tam tikru laiko intervalu – tarp 1.25 millisekundės (0^0) ir 1.75 millisekundės (180^0). 1.5 milisekundės atitinka 90^0. Laikas skirtingų gamintojų servuose gali skirtis. Jei impulsas yra siunčiamas kas 25-50 milisekundžių, servo mechanizmas veiks sklandžiai. Vienas iš didžiausių Arduino privalumų, kad jis turi programinės įrangos biblioteką, kuri leidžia kontroliuoti du servo mechanizmus (prijungtas į 9 ir 10 jungtis) viena kodo eilute.

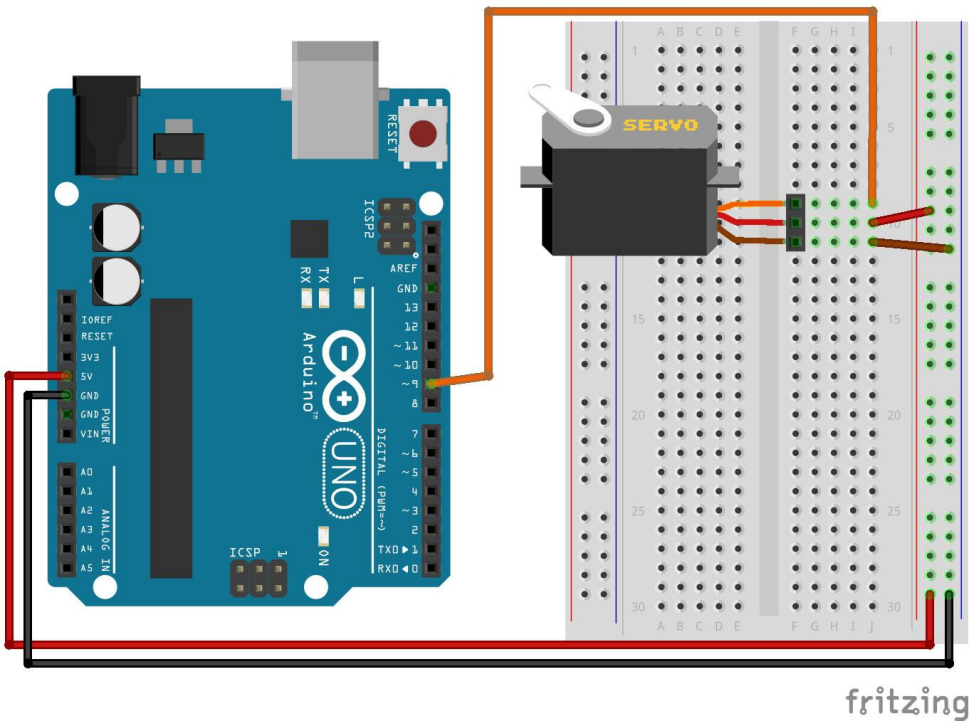
Grandinės dalys

Mini servo mechanizmas
x1

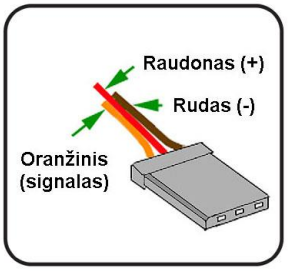
Jungiamieji laidai
x5

3 Jungtys
x1

Surinkta grandinė



Servo mechanizmas:



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-08: Judesio kontrolė - Servo mechanizmai            |
 *     ------------------------------------------------------------
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 04: A Single Servo - Servos
 *
 * Valdysime servo mechanizmą, sukdami jį pirmyn ir atgal, per visą savo judesių amplitudę
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 */

#include <Servo.h>     // įtraukima servo programinės įrangos biblioteka

Servo myservo;         // sukuria servo objektą servo mechanizmui kontroliavimui
                       // daugiausia aštuoni servo objektai gali būti sukurti

int pos = 0;           // inicijuojamas kintamasis servos pozicijai fiksuoti

void setup()
{
  myservo.attach(9);  // prijungia servo mechanizmą 9 jungtyje prie servo objekto
}


void loop()
{
  for(pos = 0; pos < 180; pos += 1)  // suka mechanizmą nuo 0 iki 180⁰
  {                                  // palengva, po 1⁰
    myservo.write(pos);              // liepia servui suktis į kintamojo "pos" poziciją
    delay(15);                       // laukia 15ms pasiekti pozicijai
  }
  for(pos = 180; pos>=1; pos-=1)     // suka nuo 180 iki 0 laipsnių
  {
    myservo.write(pos);              // liepia servui suktis į kintamojo "pos" poziciją
    delay(15);                       // laukia 15ms pasiekti pozicijai
  }
}
```

Neveikia? (Trys dalykai bandymui)

Nesisuka?	Vis dar neveikia?	Fits and Starts
Pažiurėk, ar servo mechanizmas įjungtas teisinga puse.	Patikrink ar prijungei į maitinimą.	Jei serva pradeda suktis, tačiau trūkčioja ir mirksi lemputė ant Arduino plokštės, reiškia energijos šaltinis yra per silpnas. Naudojant šviežią bateriją vietoj USB turėtų išspręsti problemą.

Padaryti geriau?

Kontrolė pasitelkus potenciometrą:
Eik čia: File > Library-Servo > Knob. Naudosime potenciometrą (žr. GRND-02) servo mechanizmui valdymui. Instrukcijas gali rasti čia: https://www.arduino.cc/en/Tutorial/Knob

Laiko kontroliavimas:
Nesunku servą valdyti naudojant Arduino biblioteką, tačiau pabandykime "suprogramuoti" kažką patys. Galime

valdyti impulsus tiesiogiai. Šis metodas tinks valdyti servo mechanizmus bet kuria iš 20 Arduino jungčių (prieš tai darant reikėtų paderinti programos kodą).

```
int servoPin = 9;

void setup(){
  pinMode(servoPin,OUTPUT);
}

void loop() {
  int pulseTime = 2100; //(pauzės laikas mikrosekundėmis (1500 90 laipsnių
                        // 900 0 laipsnių 2100 180 laipsnių)
  digitalWrite(servoPin, HIGH);
  delayMicroseconds(pulseTime);
  digitalWrite(servoPin, LOW);
  delay(25);
}
```

Puikios idėjos:
Kelios puikios idėjos, kaip galima panaudoti servo mechanizmus –

Kalėdinis smūgių skaičiuotuvas
http://ardx.org/XMAS

Atviro kodo robotinė ranka
http://ardx.org/RARM

Vaikščiojantis robotas
http://ardx.org/SEWA

{GRND-09}

Kuriame muziką –
Pjezo signalizatoriai

Ką darysime

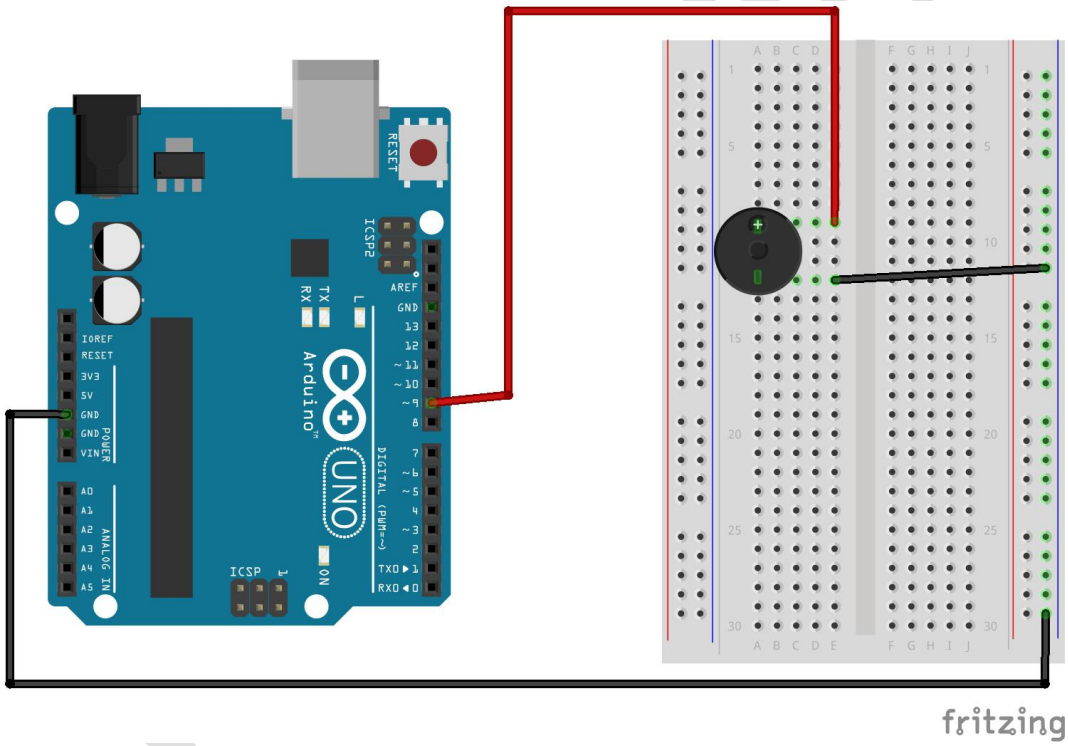
Šiuo metu mes jau mokame su Arduinu valdytį šviesas, judesį ir elektronus. Galime pereiti prie garso. Tačiau jis yra analoginis reiškinys. Kažin kaip su juo susidoros skaitmeninis Arduino? Mes pasitelksim jo neįtikėtiną greitį, kuris leis imituoti analogines sąvybes. Norėdami tai padaryti, mes prijungsime pjezo signalizatorių prie vieno iš Arduino skaitmeninių kontaktų. Pjezo signalizatorius padaro spragtelėjimą kiekvieną kartą, kai į jį patenka srovės impulsas. Jei mes duosime jam impulsą tinkamu dažnumu (pvz. 440 kartų per sekundę, norėdami sukurti natą A) šie paspaudimai bus paleisti kartu, siekiant išgauti natą. Ekspermentuokite su Pjero signalizatoriumi ir kurkite savo muziką su Arduino.

Grandinės dalys

Pjezo signalizatorius
x1

Laidai
x2

Surinkta grandinė



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-09: Kuriame muziką - Pjezo signalizatoriai          |
 *     ------------------------------------------------------------
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 06: Music - Piezo Elements
 *
 *
 * Šis pavyzdys naudoja piezo garsiakalbį groti melodijai. Jis siunčia stačiakampes bangas
 * nustatytu dažnumu pjezo signalizatoriui, kad jis generuotų atitinkamus tonus.
 *
 * Tono skaičiavimas yra padaromas remiantis sekančia matematine funkcija:
 *
 *       timeHigh = period / 2 = 1 / (2 * tono dažnumas)
 *
 * kur skirting tonai yra apibūdinami kaip lentelėje:
 *
 * nata (note)     dažnumas     periodas    timeHigh
 *      c           261 Hz        3830        1915
 *      d           294 Hz        3400        1700
 *      e           329 Hz        3038        1519
 *      f           349 Hz        2864        1432
 *      g           392 Hz        2550        1275
 *      a           440 Hz        2272        1136
 *      b           493 Hz        2028        1014
 *      C           523 Hz        1912        956
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 * Plačiau: http://www.arduino.cc/en/Tutorial/Melody
 */

int speakerPin = 9;  // priskirti pjezo signalizatoriaus kontaktą (pin) pin9

int length = 28;     // nustatyti melodijos natų skaičių
char notes[] = "ggecggecfaCaggfaafeggedfedcc "; // paliktas tarpas reiškia pauzę
float beats[] = { 1 , 1 , 1, 1, 1, 1, 1, 1, 1, 1, 1, 2.75 , 2.75, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 2.75 , 2.75 };
int tempo = 300;

void playTone(int tone, int duration) {
  for (long i = 0; i < duration * 1000L; i += tone * 2) {
    digitalWrite(speakerPin, HIGH);
    delayMicroseconds(tone);
    digitalWrite(speakerPin, LOW);
    delayMicroseconds(tone);
  }
}

void playNote(char note, int duration) {
  char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
  int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };

  // groti toną atitinkantį žymens pavadinimą
  for (int i = 0; i < 8; i++) {
    if (names[i] == note) {
      playTone(tones[i], duration);
    }
  }
}

void setup() {
  pinMode(speakerPin, OUTPUT);
}

void loop() {
  for (int i = 0; i < length; i++) {
    if (notes[i] == ' ') {
```

```
      delay(beats[i] * tempo); // rest
    } else {
      playNote(notes[i], beats[i] * tempo);
    }

    // pauzė tarp natų
    delay(tempo / 2);
  }
}
```

Neveikia? (Trys dalykai bandymui)

Nėra Garso	Negali galvoti kol groja melodija?	Pavargai nuo mirganti mirganti maža žvaigždutė?
Atsižvelgiant į tai, dydį ir formą piezo elementai lengvai praleidžia tinkamas skyles ant breadboard. Išbandykite, ar dvigubus patikrinimus jo vietoje.	Ištraukite piezo elementus kol galvojate, pakraukite savo programą ir vėl juos įkiškite.	Kodas yra parašytas, todėl tu gali lengvai pridėti savo dainų, patikrink žemiau esantį kodą pradėdamas.

Padaryti geriau?

Žaidimas su greičiu:
Kiekvieno užrašo laikas skaičiuojamas remiantis kintamuoju, kaip, pavyzdžiui, mes galime įgnybti kiekvieno pranešimo garsą ar laiką. Norėdami pakeisti melodiją turime keisti tik vienos eilutės greitį.

`int tempo = 300; ---> int tempo = (naujas #)`

Pakeisti didesnio numerio melodiją į lėtesnę ar mažesnio numerio melodiją į greitesnę.

Užrašų nustatymai:
Jei esate susirūpinę pastabomis čia yra naudingos informacijos. Pastabos buvo apskaičiuotos remiantis formule į komentaro bloką programos viršuje. Bet komponuoti atskirus užrašus galite tiesiog keisdami savo tonų vertes. [] masyvas aukštyn arba žemyn, kol jie skambės gerai. (kiekviena nata yra suderinta jo pavadinimo pavadinimuose [] (masyvas ty. c = 1915)

```
char names[] = { 'c', 'd', 'e', 'f', 'g', 'a', 'b', 'C' };
int tones[] = { 1915, 1700, 1519, 1432, 1275, 1136, 1014, 956 };
```

Sudarykite savo melodijas:
Programa yra iš anksto nustatyti groti "Žiba žiba maža žvaigždutė", tačiau, kaip ji yra užprogramuota, tai suteikia galimybę ją lengvai keisti. Kiekviena daina yra apibrėžiama vienu int ir dviem masyvų, int ilgis apibrėžia užrašų skaičių, pirmieji masyvo pastabos `[]` apibrėžia kiekvieną pastabą, antroji plaka `[]` apibrėžia, kiek laiko kiekvienas pastaba yra grojama. Kai kurie pavyzdžiai:

Melodija – *Twinkle Twinkle Little Star*

```
int length = 15;
char notes[] = "ccggaagffeeddc ";
int beats[] = { 1, 1, 1, 1, 1, 1, 2, 1, 1, 1, 1, 1, 1, 2, 4 };
```

Melodija – *Happy Birthday* (pirma linija)

```
int length = 13;
char notes[] = "ccdcfeccdcgf ";
int beats[] = {1,1,1,1,1,2,1,1,1,1,1,2,4};
```

{GRND-10}

Muzikinis sintezatorius – Pjezo signalizatoriai

Ką darysime

Sukursime muzikinį sintezatorių. Garso signalo toną valdysime dviem potenciometrais. Programos kodas apdoroja du parametrus, kurie yra nuskaitomi iš dviejų potenciometrų. Vienas potenciometras įtakoja pikio lygį, kitas – nuolat pasikartojančio ciklo trukmę.

Grandinės dalys

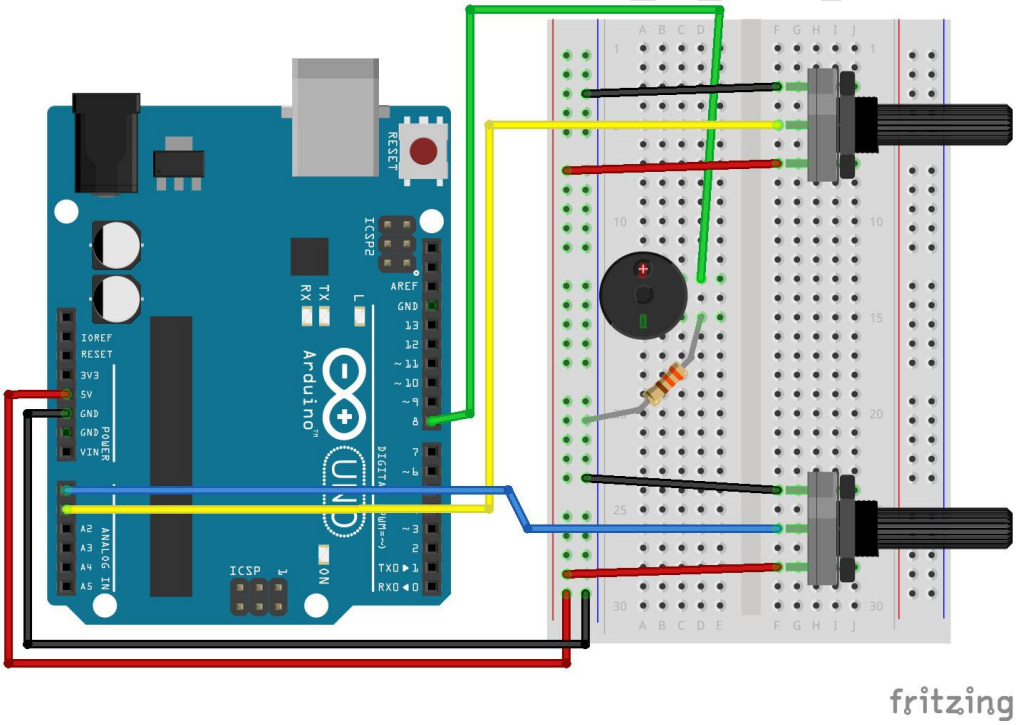
330 Ω rezistorius
(oranžinis-oranžinis-rudas)
x1

100 kΩ potenciometrai
x2

Pjezo signalizatorius
x1

Jungiamieji laidai
x10

Surinkta grandinė



Potenciometras:

Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ---------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                  |
 *     |  GRND-10: Muzikinis sintezatorius - Pjezo signalizatorius |
 *     ---------------------------------------------------------------
 *
 * Šaltinis: iš Fritzing Creator Kit: www.fritzing.org/creator-kit
 *
 * Naudosime pjezo signalizatorių generuoti įvairius garsus. Garso signalo toną valdysime
 * dviem potenciometrais. Vienas potenciometras įtakoja pikio lygį, kitas – nuolat
 * pasikartojančio ciklo trukmę.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 *
 * Žiūrėti: http://fritzing.org/projects/fritzing-creator-kit-14-synthesizer
 *
 */

int buzzerPin = 8;                      // piezo pin declaration
int potPin1 = 0;                        // pirmo potenciometro priskyrimas 1 pin
int potPin2 = 1;                        // antro potenciometro priskyrimas 2 pin

int toneHeight;                         // inicijuoti tono aukštį
int lfo;                                // inicijuoti dažnio greitį

/* lfo yra "žemo dažnio osciliatorius" – dažnio kitimas priklauso nuo laiko */

void setup() {
  pinMode(buzzerPin, OUTPUT);           // inicijuoti signalizatoriaus kontaktą kaip išvadą
}

void play(int myToneHeight) {           // priskirti grojimo būdą
    digitalWrite(buzzerPin, HIGH);      // įjungti signalizatorių
    delayMicroseconds(myToneHeight);    // sustojimo laikas įtakoja toneHeight
    digitalWrite(buzzerPin, LOW);       // išjungti signalizatorių
    delayMicroseconds(myToneHeight);    // sustojimo laikas įtakoja toneHeight
}

void loop() {
  toneHeight=analogRead(potPin1);       // nuskaityti 1 potenciometro toneheight reikšmę
  lfo=analogRead(potPin2);              // nuskaityti 2 potenciometro lfo reikšmę

  for (int i = (lfo/10); i > 0; i--) {  // toneheight augimas priklausomai nuo LFO reikšmės
    play(toneHeight);                   // vykdyti grojimo būdą
  }
  delayMicroseconds(lfo);               // padaryti pauzę
  for (int i = 0; i < lfo/10; i++) {    // toneheight kritimas priklausomai nuo LFO reikšmės
    play(toneHeight);                   // vykdyti grojimo būdą
  }
}
```

{GRND-11}

Sukame variklį –
DC variklio valdymas

Ką darysime

Arduino yra puikus prietaisas, norint tiesiogiai valdyti mažus elektrinius prietaisus pavyzdžiui, LED. Tačiau, kai susiduriama su didesniais daiktais (pavyzdžiui, žaisliniu varikliu ar skalbimo mašina), būtina panaudoti tranzistorių. Tranzistorius yra naudingas prietaisas. Jis keičia pratekančios srovės stiprumą, naudodamas mažus jos kiekius. Tranzistorius turi 3 kontaktus (emiterį, bazę ir kolektorių). Turėdami NPN tranzistoriaus tipą – jo kolektoriaus kontaktą prijungiate prie apkrovos, o emiterį – į žemę GND. Tada, kai maža srovė tekės iš bazės į emiterį, srovė tekės ir per tranzistorių – jūsų variklis suksis (tai atsitinka, kai mes nustatome Arduino į HIGH). Yra tūkstančiai skirtingų tipų tranzistorių, leidžiančių puikiai suderinti kiekvieną situacija.

Grandinės dalys

330 Ω rezistorius
(oranžinis- oranžinis - rudas)
x1

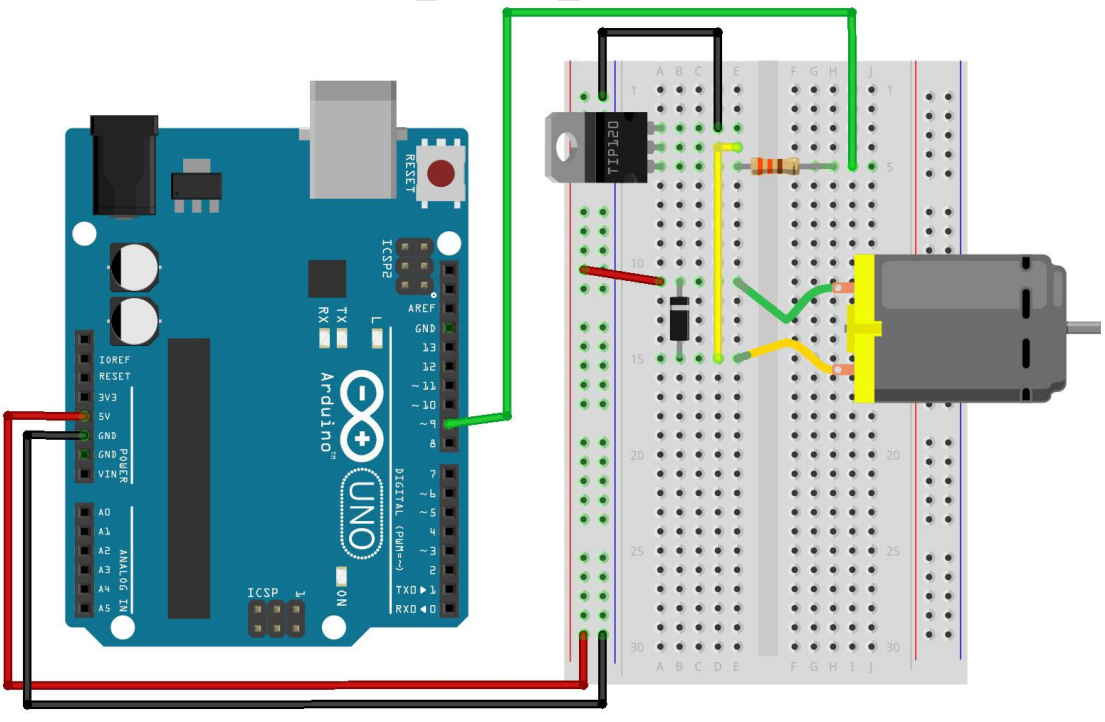
Žaislinis motoras
x1

Jungiamieji laidai
x8

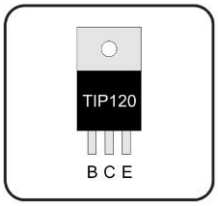
Diodas 1N4001
x1

Tranzistorius TIP120
x1

Surinkta grandinė



NPN tranzistorius TIP120:



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ----------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                    |
 *     |  GRND-11: Sukame variklį - Nuolatinės srovės variklio valdymas  |
 *     ----------------------------------------------------------------
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 03: Spin Motor Spin - Transistor &
 * Motor
 *
 * Arduino yra puikus, norint valdyti šviesos diodus, tačiau jeigu mes prijungtume kažką, kas
 * reikalauja daugiau galios, mes galime nesunkiai jį sudeginti. Didesnių galios prietaisų
 * valdymui mums reikės tranzistoriaus pagalbos. Šiame projekte naudosime tranzistorių mažo
 * nuolatinės srovės variklio valdymui.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 */

int motorPin = 9;  // inicijuojame kontaktą, į kurį įjungtas variklis
                   // (jeigu naudosite 9,10,11 arba 3 kištukus, jūs taip pat galėsite
kontroliuoti greitį)

/*
 * setup() – ši funkcija veikia vieną kartą, kai įjungiate Arduino.
 * Mes nustatėmė variklio kištuko išvestį į aukštą (kištukas +5V)arba žemą (kištukas -GROUND)
 * O ne įvestį (tikrinimą ar kištukas yra aukštos ,ar žemos įtampos)
 */
void setup()
{
 pinMode(motorPin, OUTPUT);
}


/*
 * loop() – ši funkcija prasidės, kai pasibaigs funkcija setup, po to ji kartosis.
 * mes suteikiamia vardą funkcijai pavadinimu motorOnThenOff()
 */

void loop()                        // vykdyti ciklą
 motorOnThenOff();
 //motorOnThenOffWithSpeed();
 //motorAcceleration();
}

/*
 * motorOnThenOff() – įjungia variklį, tada išjungia
 * Pastaba! Šis programos kodas yra identiškas, kurį naudojome mirksinčiam LED.
 */
void motorOnThenOff(){
  int onTime = 2500;  //laikas milisekundėmis, kurį variklis
įsijungs.    int offTime = 1000; //laikas milisekundėmis, kurį variklis išsijungs.

  digitalWrite(motorPin, HIGH);   // įjungia variklį
  delay(onTime);                  // palaukia onTime millisekundžių
  digitalWrite(motorPin, LOW);    // išjungia variklį
  delay(offTime);                 // palaukia offTime millisekundžių
}

/*
 * motorOnThenOffWithSpeed() - įjungia ir išjungia variklį, naudojamos ir greičio vertės
 * Pastaba! Šis kodas identiškas, kurį naudojome mirksinčiam LED.
 *
 */
void motorOnThenOffWithSpeed(){

  int onSpeed = 200;  // skaičius tarp 0 (sustabdytas) ir 255 (visas greitis)
  int onTime = 2500;  // milisekundžių skaičius varikliui įjungti

  int offSpeed = 50;  // skaičius tarp 0 (sustabdytas) ir 255 (visas greitis)
  int offTime = 1000; // milisekundžių skaičius varikliui išjungti
```

```
  analogWrite(motorPin, onSpeed);   // įjungia variklį
  delay(onTime);                    // palaukia onTime millisekundžių
  analogWrite(motorPin, offSpeed);  // išjungia variklį
  delay(offTime);                   // palaukia offTime millisekundžių
}

/*
 * motorAcceleration() - variklio greitis didėja iki didžiausio, tada atgal iki 0 reikšmės
void motorAcceleration(){
  int delayTime = 50; //millisekundės tarp kiekvieno greičio didėjimo žingsnio

  //Greitina motorą
  for(int i = 0; i < 256; i++){     // pereina per visus greičius nuo 0 iki 255
    analogWrite(motorPin, i);       // priskiria naują greitį
    delay(delayTime);               // palaukia delayTime millisekundžių
  }

  //Lėtina motorą
  for(int i = 255; i >= 0; i--){    // pereina per visus greičius nuo 255 iki 0
    analogWrite(motorPin, i);       // priskiria naują greitį
    delay(delayTime);               // palaukia delayTime millisekundžių
  }
}
```

Neveikia? (Trys dalykai bandymui)

Variklis nesisuka?	Vis dar neveikia?	Ir toliau nesiseka?
Jeigu naudojate savo šaltinio tranzistorių, patikrinkite ar jis suderinamas su kontaktų numeracija (dauguma yra atvirkščiai apsukti).	Jeigu tai jūsų variklis, patikrinkite ar jis veikia su 5 voltais. Įsitikinkite ar jam nereikia daugiau galios.	Kartais Arduino atsijungia nuo kompiuterio. Pamėginkite iš naujo prijungti USB kištuką.

Padaryti geriau?

Kontroliuojame greitį:
Anksčiau mes žaidėme su Arduino galimybe valdyti LED ryškumą! Dabar mes naudosime tą pačią funkciją variklio greičio valdymui. Arduino, tai daro naudodamas vadinamąją pulso pločio moduliaciją (PWM). Tai priklauso nuo Arduino galimybės veikti tikrai greitai. Užuot tiesiogiai kontroliuodamas įtampą ateinančią iš kištuko, Arduino perjunginės kištuką iš *on* į *off* labai greitai. Kompiuterių pasaulyje tai vyksta nuo 0 iki 5 voltų, daug kartų per sekundę, bet ir mūsų pasaulyje mes matome tai, kaip įtampa. Pavyzdžiui, jei Arduino PWM'ing 50% matome šviesos blyškius 50%, nes mūsų akys yra nepakankamai greitos, pamatyti jį mirksintį.Ta pati funkcija veikia su tranzistoriais. Netikite? Išbandykite patys!

```
// motorOnThenOff();
motorOnThenOffWithSpeed();
//motorAcceleration();
```

Įkelkite į programą. Jūs galite pakeisti greičius keičiant funkcijų `onSpeed` ir `offSpeed duomenis`.

Greitėjimas ir lėtėjimas:
Kodėl sustabdyti, kodėl nepagreitinti ir vėl sulėtinti motoro? Norint tai atlikti pakeiskite `loop()kodą`.

```
// motorOnThenOff();
// motorOnThenOffWithSpeed();
motorAcceleration();
```

Tada įkelkite programą ir stebėkite kaip Jūsų motoras lėtai greitės iki viso greičio, tada pradės lėtėti. Jeigu norite pakeisti greitėjimo greitį, pakeiskite duomenis delayTime (kuo didesnis, tuo ilgesnis greitėjimo laikas).

{GRND-12}

Didesnė apkrova –
Relės

Ką darysime

Mes apjungsime tai, ka mokėmės apie tranzistorius (Pro-11) valdant relę. Relė yra elektra valdomas mechaninis jungiklis. Mažoje plastikinėje dėžutėje yra elektromagnetas, kuris įelektrintas perjungia jungiklį (dažnai girdimas malonus spragsėjimo garsas). Jūs galite nusipirkti skirtingų dydžių relių, kurios skiriasi didžiu nuo labai mažų iki dydžiu sulyg šaldytuvas, kurios geba valdyti tam tikros įtampos srovę. Jie yra nepaprastai įdomus. Su visais iki šiol dirbtais komponentais kartais būna linksma sujungus šimtus jungiklių valdyti kažką įspūdingo. Prijungus Arduino galima valdyti visas reles iš karto.

Grandinės dalys

330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x3

Diodas 1N4001
x1

Žalias 5mm šviesos diodas
x1

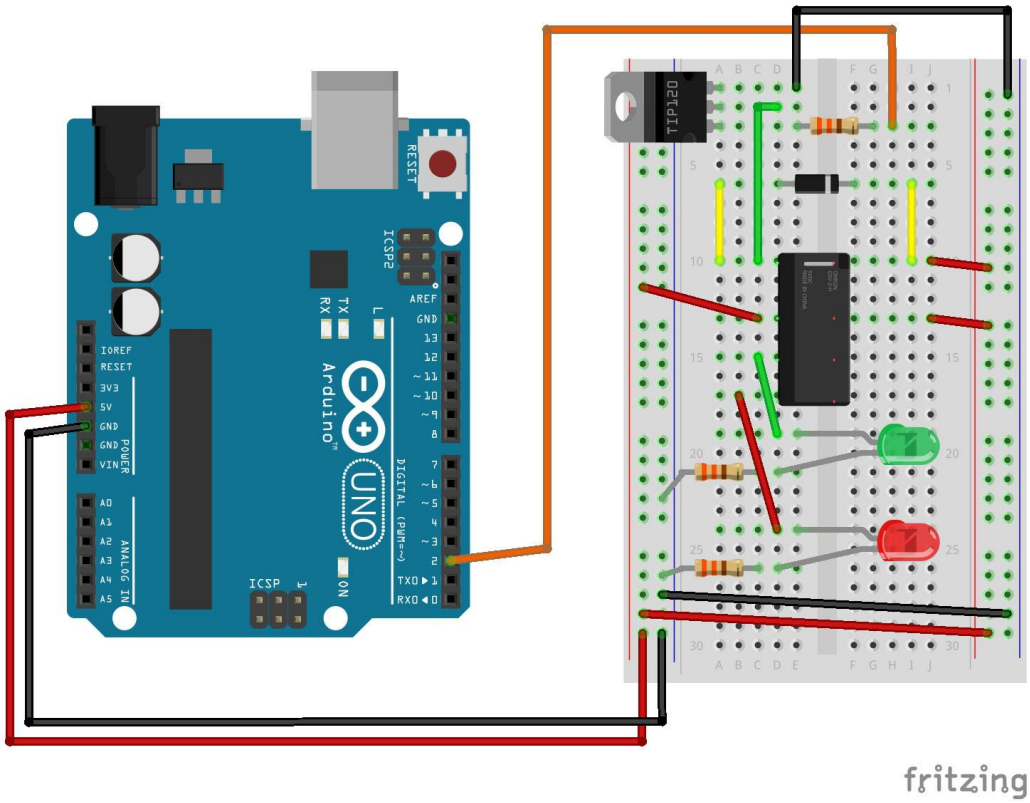
Jungiamieji laidai
X13

Relė DPDT tipo 5V
x1

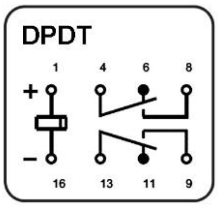
Tranzistorius TIP120
x1

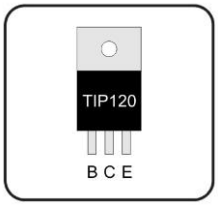
5mm skirtingų spalvų šviesos diodai
x2

Surinkta grandinė



Rėlė DPDT tipo:





Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:      |
 *     |  GRND-12: Didesnė apkrova – Rėlės               |
 *     ------------------------------------------------------
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 11: Larger Loads - Relays
 *
 * Pasikartojančiai įjungia LED vienai sekundei, po to laiko išjungtą vieną sekundę.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 */

int ledPin =  2;    // Relė prijungta į pin 2   <-----Pakeisti į pin 2

// setup() veikia kartą, kai vykdoma programa

void setup()   {
  // priskiria pin kaip išeigą:
  pinMode(ledPin, OUTPUT);
}

// loop() programos dalis kuri kartojasi - ciklas, kol Arduinu teka elektra

void loop()
{
  digitalWrite(ledPin, HIGH);   // įjungia šviesos diodą
  delay(1000);                  // palaukia sekundę
  digitalWrite(ledPin, LOW);    // išjungia šviesos diodą
  delay(1000);                  // palaukia sekundę
}
```

Neveikia? (Trys dalykai bandymui)

Niekas nevyksta

Pavyzdys naudoja pin 13, o mes rėlę prijungėm į pin 2. Įsitikinkite, kad jūssavo kode irgi tai pakeitėte.

Nesigirdi spragsėjimo garso

Tranzistorius arba jungiamoji grandinės dalis neveikia. Patikrinkite ar tranzistorius gerai sujungtas.

Nevisiškai veikia

Relė netaisyklingai sujungta arba nevisi laidai tinkamai sujungti. Reikia stipriau prispausti rėlę ar laidus.

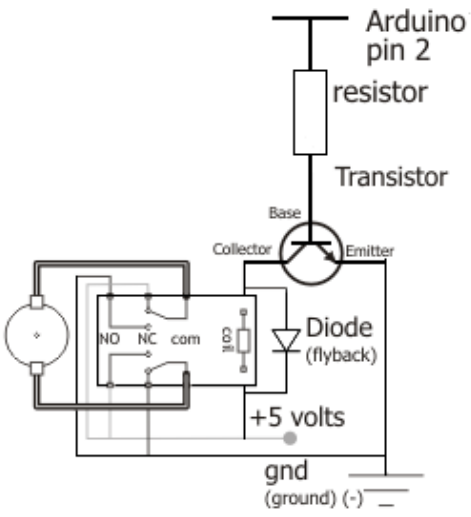
Padaryti geriau?

Valdyti variklį

GRND-11 mes kontroliavom variklį naudodami tranzistorių. Norit valdyti galingenį variklį, turėsime pasinaudoti rėle. Išimkit raudoną LED ir prijunkite jo vietoje motorą(nepamirškite prijungti 560 Omų varžą).

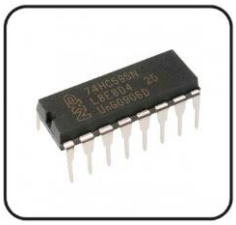
Valdyti variklio sukimosi kryptį

Kad galėtume kontroliuoti jo sukimąsi, mes turime apsukti elektros srovės tekėjimo kryptį. Tereikia apsukti laidus. Tam mums reikia sudaryti H-bridge grandinę. Tai gali atrodyti sudėtinga, bet iš tikrųjų užtrunka nedaug laiko. Pamėginkite.



{GRND-13}

Daugiau šviesos diodų –
Postūmio registras 74HC595



Ką darysime

Laikas pradėti dirbti su lustais arba integriniais grandynais kaip juos yra įprasta vadinti. Lusto išorė gali būti labia apgaulinga. Pavyzdžiui, lustas, naudojamas Arduino plokštėje (mikrokontroleryje) ir tas, kurį naudosime šioje grandinėje (poslinkio registre) atrodo labai panašiai, tačiau yra skirtingi. ATMega lusto, kuris yra ant Arduino plokštės, kaina yra keli doleriai, o 74HC595 lusto – kelios dešimtys centų. Šis lustas labai geras pradedantiesiems ir kai jausitės patogiai dirbdami su juo ir jo duomenimis (prieinama internete http://ardx.org/74HC595), lustų pasaulis bus jūsų austrė. Poslinkio registras (taip pat vadinamas lygiagrečios serijos konverteriu), duos jums papildomus 8 išėjimus (LED ir panašių dirbinių kontroliavimui), naudojant tik tris Arduino kištukus. Jie taip pat gali būti sujungti kartu, kad suteiktų jums beveik neribotą skaičių rezultatų naudojant tuos pačius keturis kaiščius. Kad būtų galima tuo naudotis jūs turite „sulaikrodininti“ duomenis ir tada juos užrakinti (užsklęsti). Norėdami tai padaryti, nustatykite duomenų kaištukus aukštai (high) arba žemai (low), siųskite duomenis į laikrodį, tada nustatykite duomenų kaištuką vėl ir siuntinėkite duomenis į laikrodį tol, kol būsite įregistravę 8bitus duomenų. Tada siunčiate duomenis į sklendę ir 8 bitai yra perkeliami į registrų perjungimo kaiščius. Skamba sudėtingai, bet yra labai paprasta kai jau suprantate.
(jei norite dar giliau pažvelgti į tai, kaip veikia registrų perjungimas apsilankykite: http://ardx.org/SHIF)

Grandinės dalys

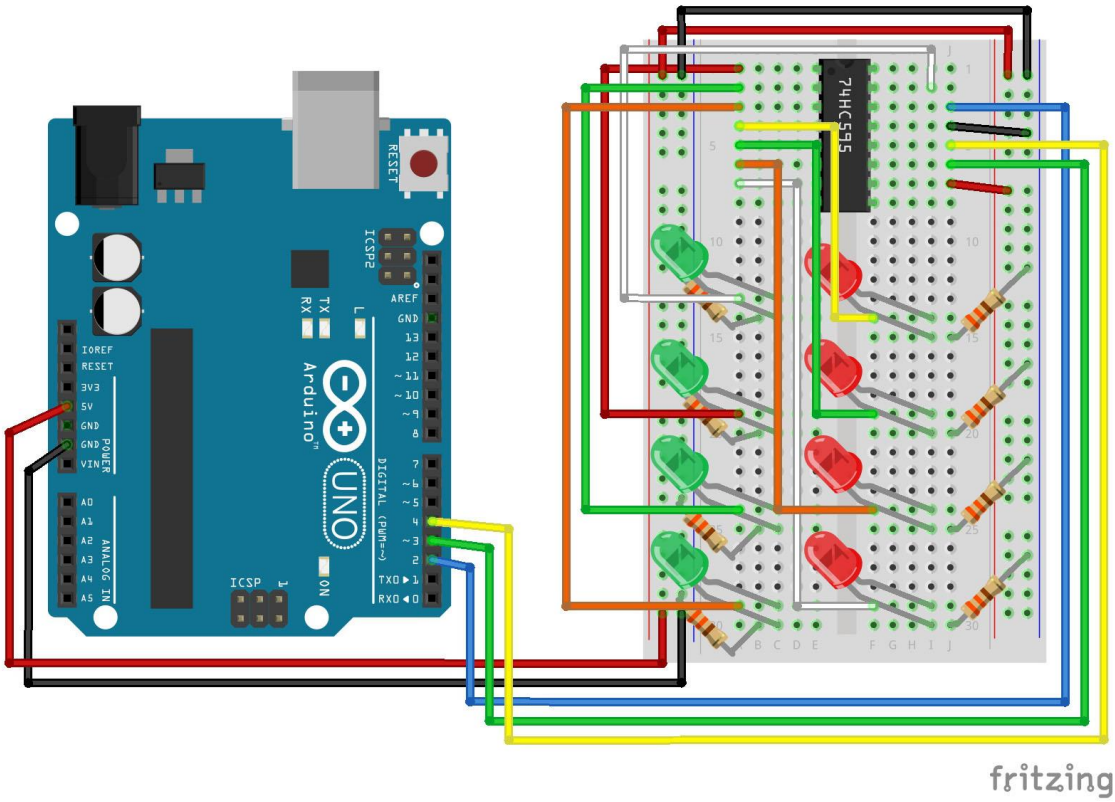
330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x8

Jungiamieji laidai
X13

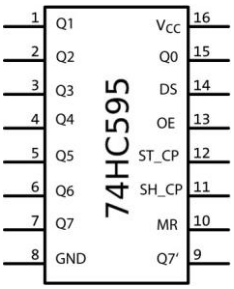
Postūmio registras 74HC595
x1

5mm šviesos diodai
x8

Surinkta grandinė



Postūmio registras 74HC595:



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                           |
 *     |  GRND-13: Daugiau šviesos diodų - Postūmio registras 74HC595   |
 *     ------------------------------------------------------------------------
 *
 * Šaltinis: Arduino Expermentation Kit (ARDX) - Circuit 05: 8 More LEDs - 74HC595 Shift Register
 *
 * Mes jau valdėme aštuonis šviesos diodus, tačiau dabar padarysime tai kiek kitaip.
 * Vietoj to, kad naudotume 8 kištukus, naudosime tik - 3 ir papildomą integrinį grandyną.
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - http://oomlout.com/a/products/ardx/
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 */

 // Pin Definitions
 // Pin Definitions
 // Grandynas 74HC595 naudoja serijinę komunikaciją
 // grandyno 3 kontaktų priskyrimas
 int data = 2;
 int clock = 3;
 int latch = 4;

//inicijuojamas vieno šviesos diodo valdymas
int ledState = 0;
const int ON = HIGH;
const int OFF = LOW;


 /*
 * setup() - ši funkcija paleidžiama vieną kartą, kai Arduino įjungiamas (on)
 * priskiriame tris valdančius kontaktus išvadams (OUTPUT)
 */
void setup()
{
  pinMode(data, OUTPUT);
  pinMode(clock, OUTPUT);
  pinMode(latch, OUTPUT);
}

/*
 * loop() - ši funkcija kartoja nurodytą ciklą, kai įvykdoma setup funkcija
 * we set which LEDs we want on then call a routine which sends the states to the 74HC595
 */
void loop()                     // ciklo kartojimas
{
  int delayTime = 100; // milisekundžių skaičius to delay between LED updates
  for(int i = 0; i < 256; i++){
   updateLEDs(i);
   delay(delayTime);
  }
}

/*
 * updateLEDs() - sends the LED states set in ledStates to the 74HC595
 * sequence
 */
void updateLEDs(int value){
  digitalWrite(latch, LOW);     //Pulls the chips latch low
  shiftOut(data, clock, MSBFIRST, value); //Shifts out the 8 bits to the shift register
  digitalWrite(latch, HIGH);   //Pulls the latch high displaying the data
}

/*
 * updateLEDsLong() - sends the LED states set in ledStates to the 74HC595
 * sequence. Same as updateLEDs except the shifting out is done in software
 * so you can see what is happening.
 */
void updateLEDsLong(int value){
  digitalWrite(latch, LOW);    //Pulls the chips latch low
  for(int i = 0; i < 8; i++){  //Will repeat 8 times (once for each bit)
  int bit = value & B10000000; //We use a "bitmask" to select only the eighth
                               //bit in our number (the one we are addressing this time thro
                    //ugh
  value = value << 1;          //we move our number up one bit value so next time bit 7 will
                    // be
                               //bit 8 and we will do our math on it
  if(bit == 128){digitalWrite(data, HIGH);} //if bit 8 is set then set our data pin high
```

```
  else{digitalWrite(data, LOW);}            //if bit 8 is unset then set the data pin low
  digitalWrite(clock, HIGH);                //the next three lines pulse the clock pin
  delay(1);
  digitalWrite(clock, LOW);
  }
  digitalWrite(latch, HIGH);  //pulls the latch high shifting our data into being displayed
}


//These are used in the bitwise math that we use to change individual LEDs

//Plačiau: http://en.wikipedia.org/wiki/Bitwise operation
int bits[] = {B00000001, B00000010, B00000100, B00001000, B00010000, B00100000, B01000000, B10000000};
int masks[] = {B11111110, B11111101, B11111011, B11110111, B11101111, B11011111, B10111111, B01111111};
/*
 * changeLED(int led, int state) - keičia šviesos diodą (LED)
 * šviesos diodai (LED) yra nuo 0 iki 7 ir jų būsena yra arba 0 - OFF arba 1 - ON
 */
 void changeLED(int led, int state){
   ledState = ledState & masks[led];  //išvalo ledState nuo priskirto
   if(state == ON){ledState = ledState | bits[led];} //if the bit is on we will add it to le
                   //dState
   updateLEDs(ledState);              // siunčia naują LED būseną į postūmio registrą
 }
```

Neveikia? (Trys dalykai bandymui)

Arduino galios šviesos diodas užgęsta	**Nevisai veikia**	**Pyktis?**
Tai atsitiko mums kelis kartus, taip atsitinka kai lustas įdėtas ne ta puse. Jei pataisysite greitai niekas nesuges.	Atsiprašau, kad skamba kaip sukedęs įrašas, bet problema turbūt yra kažkas paprasto kaip supainioti laidai.	Ši grandinė yra kartu ir paprasta ir sudėtinga tuo pačiu metu. Mes norime išgirsti apie problemas kylančias jums, kad galetume jas aprašyti ateities leidiniuose.

Padaryti geriau?

Daroma sunkiuoju būdu:
Arduino padaro kiek sudėtingus veiksmus labai paprastai, duomenų pakeitimas – vienas šių dalykų. Tačiau viena iš naudingų Arduino funkcijų yra ta, kad galima padaryti dalykus lengvus ar sunkius tiek, kiek norite. Pabandykime tokį pavyzdį. Programos `loop()` pakeiskite šią eilutę.

```
updateLEDs(i) -> updateLEDsLong(i);
```

Įkelkite programą ir pastebėsite, kad niekas nepasikeitė. Jei pažvelgsite į programos kodą, galėsite pamatyti kaip komunikuojame su kiekvienu grandyno bitu atskirai (daugiau detalių https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus).

Individualių LED'ų kontroliavimas:
Laikas pradėti kontroliuoti LED'us panašiai kaip darėme GRND-04. Kadangi 8 -ios šviesos diodo būsenos yra išsaugotos viename baite (8 bitų reikšmė) detalėms kaip tai veikia pabandykite: https://en.wikipedia.org/wiki/Binary_number. Arduino yra labai geras manipuliuojant bitais ir yra visas rinkinys operatorių, kurie mums padeda. Detalės su bitais susijusiai matematikai (https://en.wikipedia.org/wiki/Bitwise_operation).

Mūsų įgyvendinimas
Pakeiskite `loop()` kodą su:

```
int delayTime = 100; //the number of milliseconds to delay
                     //between LED updates
for(int i = 0; i < 8; i++){
changeLED(i,ON);
delay(delayTime);
}
for(int i = 0; i < 8; i++){
changeLED(i,OFF);
```

```
delay(delayTime);
}
```

Ir įkėlus tai lemputės užsidegs viena po kitos ir užges taip pat. Patikrinkite kodą ir vikipediją, kad pamatytumėte kaip tai veikia.

Daugiau animacijų:

Dabar darosi įdomiau. Jei pažvelgsite atgal į GRND-04 (Daugiau šviesos diodų) kodą pamatysite, kad mes keičiame LED'us naudodami `digitalWrite(led, state)` tai tas pats formatas kaip rutina, kurią rašėme `changeLED(led, state)`. Jūs galite naudoti animacijas, kurias parašėte GRND-04 nukopijuodami kodą į šį skečą pakeisdami visus `digitalWrite()` į `changeLED()`. Galinga? Labai (jūs taip pat turėsite pakeisti dar kelis kitus dalykus bet sekite sukauptomis klaidomis ir viskas išsispręs).

{GRND-14}

Paišome figūras –
8x8 šviesos diodų modulis

Ką darysime

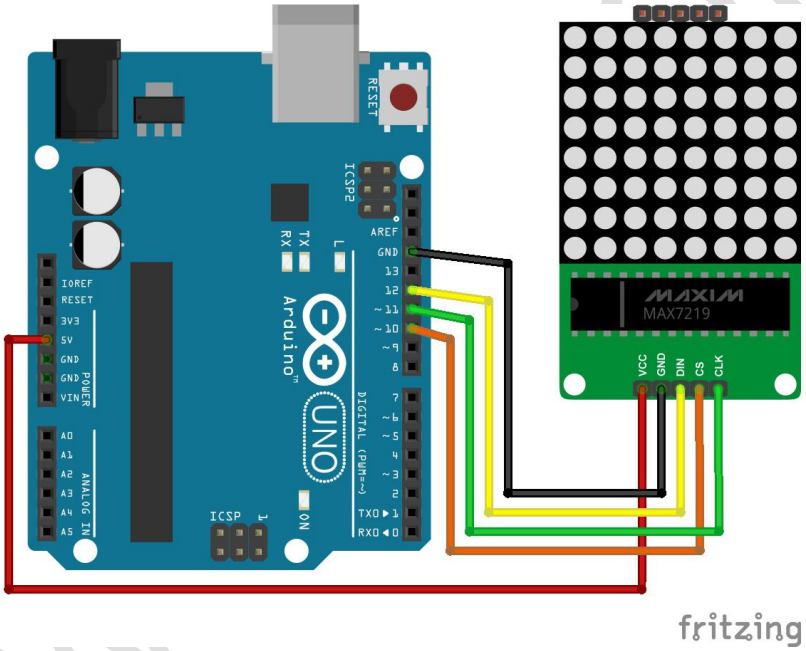
Using a LED Dot Matrix in your next project can be a way to incorporate some cool little animations.
By using these 8X8 matrix modules you can create your own.
Naudojant taškinės LED matricos modulį savo kitą projektą gali būti būdas įtraukti kai kurias kietas mažai animaciją. Naudodami šiuos 8X8 matricos modulius galite sukurti jūsų pačių.
Šie moduliai naudoja šviesos diodus valdantį MAX7219 grandyną. Todėl jo dėka galėsime įjungti ir išjungti kiekvieną iš 64 matricos šviesos diodų, naudodami tik 3 Arduino kontaktus.

Grandinės dalys

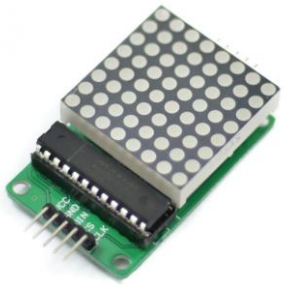
MAX7219 LED taškinės matricos modulis
x1

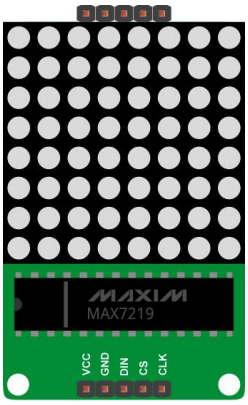
Jungiamieji laidai
x5

Surinkta grandinė



8x8 LED modulis:





Kodas (nukopijuokite kodą į IDE atvertą langą)

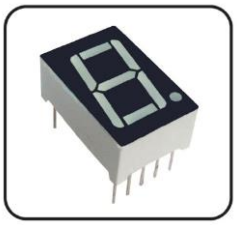
```
/*     ------------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:                |
 *     |  GRND-14: Paišome figūras - 8x8 šviesos diodų modulis     |
 *     ------------------------------------------------------------
 */

#include "LedControl.h"

LedControl lc=LedControl(12,11,10,2);  // Pins: DIN,CLK,CS, # of Display connected

unsigned long delayTime=200;  // Delay between Frames

// Put values in arrays
byte invader1a[] =
{
   B00011000,  // First frame of invader #1
   B00111100,
   B01111110,
   B11011011,
   B11111111,
   B00100100,
   B01011010,
   B10100101
};

byte invader1b[] =
{
  B00011000, // Second frame of invader #1
  B00111100,
  B01111110,
  B11011011,
  B11111111,
  B00100100,
  B01011010,
  B01000010
};
void setup()
{
  lc.shutdown(0,false);  // Wake up displays
  lc.setIntensity(0,5);  // Set intensity levels
  lc.clearDisplay(0);    // Clear Displays
}


//  Take values in Arrays and Display them
void sinvader1a()
{
  for (int i = 0; i < 8; i++)
  {
    lc.setRow(0,i,invader1a[i]);
  }
}

void sinvader1b()
{
  for (int i = 0; i < 8; i++)
  {
    lc.setRow(0,i,invader1b[i]);
  }
}
void loop()
{
// Put #1 frame on Display
    sinvader1a();
    delay(delayTime);
// Put #2 frame on Display
    sinvader1b();
    delay(delayTime);
}
```

Atsisiųsti:

Šviesos diodų matricos valdymo biblioteka: LedControl.zip
Projektas: MAX7219_Tutorial.zip

{GRND-15}

Skaičiuojame –
7 segmentų modulis



Ką darysime

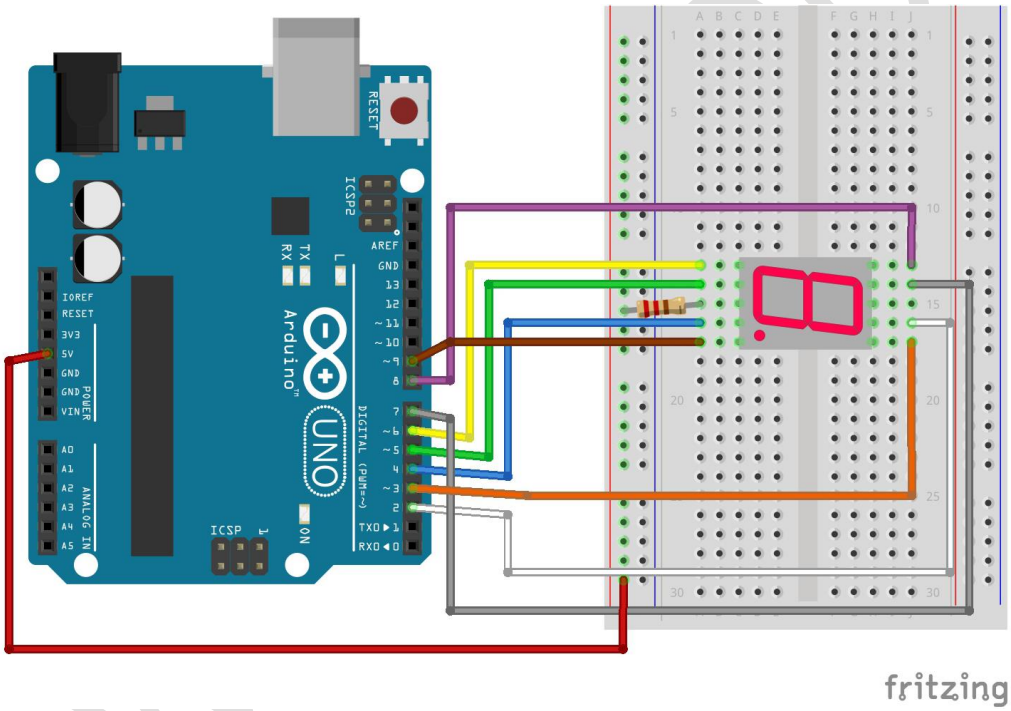
Grandinės dalys

330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x1

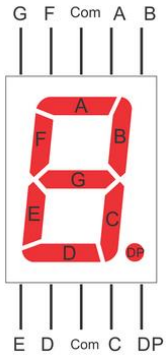
Jungiamieji laidai
x9

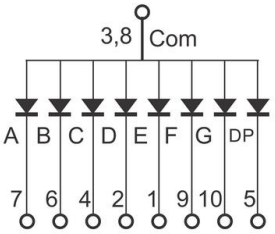
7 segmentų modulis
x1

Surinkta grandinė



7 segmentų modulis:





Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:         |
 *     |  GRND-15: Skaičiuojame - 7 segmentų modulis        |
 *     ------------------------------------------------------
 *
 *
 *  paskelbiamas 2-jų lygių masyvas kiekvienam skaitmeniui
 *  struktūra: skaitmuo[0]={viršus, kairė viršus, kairė apačia, apačia,
 *  dešinė apačia, dešinė, viršus, vidurys}
*/
byte digits[10][7] =
{
  { 0, 0, 0, 0, 0, 0, 1 }, // 0
  { 1, 0, 0, 1, 1, 1, 1 }, // 1
  { 0, 0, 1, 0, 0, 1, 0 }, // 2
  { 0, 0, 0, 0, 1, 1, 0 }, // 3
  { 1, 0, 0, 1, 1, 0, 0 }, // 4
  { 0, 1, 0, 0, 1, 0, 0 }, // 5
  { 0, 1, 0, 0, 0, 0, 0 }, // 6
  { 0, 0, 0, 1, 1, 1, 1 }, // 7
  { 0, 0, 0, 0, 0, 0, 0 }, // 8
  { 0, 0, 0, 1, 1, 0, 0 }  // 9
};

void setup() {
  for (int i = 2; i < 10; i++) {
    pinMode(i, OUTPUT);
  }
  digitalWrite(9, HIGH);
}

void loop() {
  for (int i = 0; i < 10; i++) {
    displayDigit(i);
    delay(1000);
  }
}

void displayDigit(int num) {
  int pin = 2;
  for (int i = 0; i < 7; i++) {
    digitalWrite(pin + i, digits[num][i]);
  }
}
```

Atsisųsti kodą iš: http://fritzing.org/projects/dreams-come-true-with-arduino-circuit-12-1

{GRND-16}

Labas pasauli! –
LCD ekranas

Ką darysime

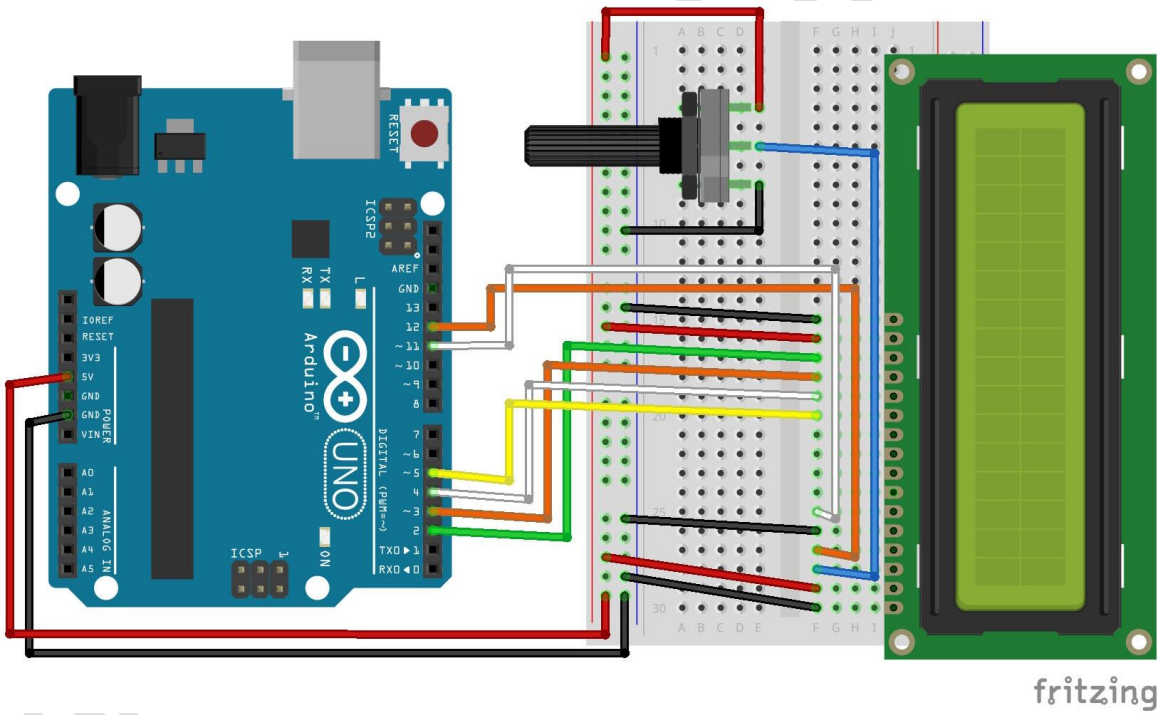
Grandinės dalys

LCD 16x2 ekranas
x1

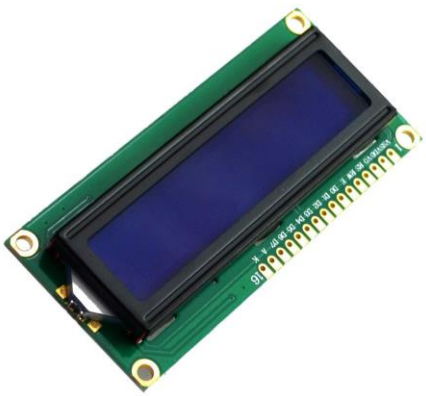
Potenciometras 10k
x1

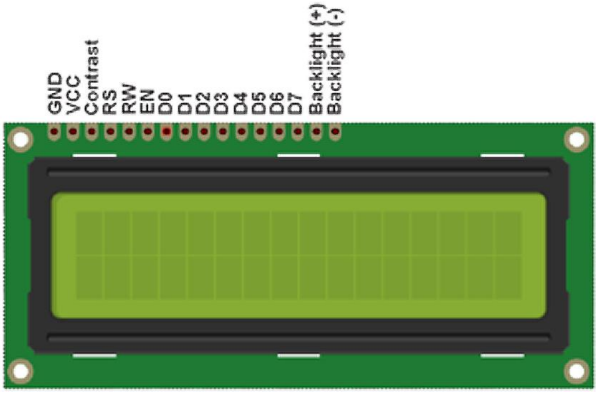
Jungiamieji laidai
X16

Surinkta grandinė



LCD ekranas:





Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*     ------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:         |
 *     |  GRND-16: Labas pasauli! - LCD ekranas           |
 *     ------------------------------------------------------
 *
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 15: Using an LCD
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.
 *
 *
 */

LIQUID CRYSTAL DISPLAY (LCD)

  A Liquid Crystal Display (LCD) is a sophisticated module
  that can be used to display text or numeric data. The display
  included in your SIK features two lines of 16 characters, and
  a backlight so it can be used at night.

  If you've been using the Serial Monitor to output data,
  you'll find that a LCD provides many of the same benefits
  without needing to drag a large computer around.

  This sketch will show you how to connect an LCD to your Arduino
  and display any data you wish.

Hardware connections:

  The LCD has a 16-pin male header attached to it along the top
  edge. Pin 1 is the pin closest to the corner of the LCD.
  Pin 16 is the pin closest to the center of the LCD.

  Plug the LCD into your breadboard.

  As usual, you will want to connect the + and - power rails
  on the side of the breadboard to 5V and GND on your Arduino.

  Plug your 10K potentiometer into three unused rows on your
  breadboard. Connect one side of the potentiometer to 5V,
  and the other side to GND (it doesn't matter which). When you
  run this sketch, you'll use the potentiometer to adjust the
  contrast of the LCD so you can see the display.

  Now connect the LCD pins. Remember that pin 1 on the LCD
  is the one closest to the corner. Start there and work your
  way up.

  1 to GND
  2 to 5V
  3 to the center pin on the potentiometer
  4 to Arduino digital pin 12
  5 to GND
  6 to Arduino digital pin 11
  7 (no connection)
  8 (no connection)
  9 (no connection)
  10 (no connection)
  11 to Arduino  digital pin 5
  12 to Arduino  digital pin 4
  13 to Arduino  digital pin 3
  14 to Arduino  digital pin 2
  15 to 5V
  16 to GND

  Once everything is connected, load this sketch into the
  Arduino, and adjust the potentiometer until the display
  is clear.
```

```
Library

  The LCD has a chip built into it that controls all the
  individual dots that make up the display, and obeys commands
  sent to it by the the Arduino. The chip knows the dot patterns
  that make up all the text characters, saving you a lot of work.

  To communicate with this chip, we'll use the LiquidCrystal
  library, which is one of the standard libraries that comes
  with the Arduino. This library does most of the hard work
  of interfacing to the LCD; all you need to pick a location
  on the display and send your data!

Tips

  The LCD comes with a protective film over the display that
  you can peel off (but be careful of the display surface as
  it scratches easily).

  The LCD has a backlight that will light up when you turn on
  your Arduino. If the backlight doesn't turn on, check your
  connections.

  As we said above, the potentiometer adjusts the contrast of
  the display. If you can't see anything when you run the sketch,
  turn the potentiometer's knob until the text is clear.

  Version 1.0 2/2013 MDG
*/

// Load the LiquidCrystal library, which will give us
// commands to interface to the LCD:

#include <LiquidCrystal.h>

// Initialize the library with the pins we're using.
// (Note that you can use different pins if needed.)
// See http://arduino.cc/en/Reference/LiquidCrystal
// for more information:

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

void setup()
{
  // The LiquidCrystal library can be used with many different
  // LCD sizes. We're using one that's 2 lines of 16 characters,
  // so we'll inform the library of that:

  lcd.begin(16, 2);

  // Data sent to the display will stay there until it's
  // overwritten or power is removed. This can be a problem
  // when you upload a new sketch to the Arduino but old data
  // remains on the display. Let's clear the LCD using the
  // clear() command from the LiquidCrystal library:

  lcd.clear();

  // Now we'll display a message on the LCD!

  // Just as with the Arduino IDE, there's a cursor that
  // determines where the data you type will appear. By default,
  // this cursor is invisible, though you can make it visible
  // with other library commands if you wish.

  // When the display powers up, the invisible cursor starts
  // on the top row and first column.

  lcd.print("hello, world!");

  // Adjusting the contrast (IMPORTANT!)

  // When you run the sketch for the first time, there's a
  // very good chance you won't see anything on the LCD display.
  // This is because the contrast likely won't be set correctly.
```

```
  // Don't worry, it's easy to set, and once you set it you won't
  // need to change it again.

  // Run the sketch, then turn the potentiometer until you can
  // clearly see the "hello, world!" text. If you still can't
  // see anything, check all of your connections, and ensure that
  // the sketch was successfully uploaded to the Arduino.
}

void loop()
{
  // You can move the invisible cursor to any location on the
  // LCD before sending data. Counting starts from 0, so the top
  // line is line 0 and the bottom line is line 1. Columns range
  // from 0 on the left side, to 15 on the right.

  // In additon to the "hello, world!" printed above, let's
  // display a running count of the seconds since the Arduino
  // was last reset. Note that the data you send to the display
  // will stay there unless you erase it by overwriting it or
  // sending a lcd.clear() command.

  // Here we'll set the invisible cursor to the first column
  // (column 0) of the second line (line 1):

  lcd.setCursor(0,1);

  // Now we'll print the number of seconds (millis() / 1000)
  // since the Arduino last reset:

  lcd.print(millis()/1000);

  // TIP: Since the numeric data we're sending is always growing
  // in length, new values will always overwrite the previous ones.
  // However, if you want to display varying or decreasing numbers
  // like a countdown, you'll find that the display will leave
  // "orphan" characters when the new value is shorter than the
  // old one.

  // To prevent this, you'll need to erase the old number before
  // writing the new one. You can do this by overwriting the
  // last number with spaces. If you erase the old number and
  // immediately write the new one, the momentary erase won't
  // be noticeable. Here's a typical sequence of code:

  // lcd.setCursor(0,1);   // Set the cursor to the position
  // lcd.print("       "); // Erase the largest possible number
  // lcd.setCursor(0,1);   // Reset the cursor to the original position
  // lcd.print(millis()/1000); // Print our value

  // NEXT STEPS:

  // Now you know the basics of hooking up an LCD to the Arduino,
  // and sending text and numeric data to the display!

  // The LCD library has many commands for turning the
  // cursor on and off, scrolling the screen, etc. See:
  // http://arduino.cc/en/Reference/LiquidCrystal
  // for more information.

  // Arduino also comes with a number of built-in examples
  // showing off the features of the LiquidCrystal library.
  // These are locted in the file/examples/LiquidCrystal menu.

  // Have fun, and let us know what you create!
  // Your friends at SparkFun.
}
```

Atsisųsti kodą iš: https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32/experiment-15-using-an-lcd

{GRND-17}

Žaidžiame –
Reakcijos žaidimas

Ką darysime

Grandinės dalys

330 Ω Rezistorius
(oranžinis-oranžinis-rudas)
x4

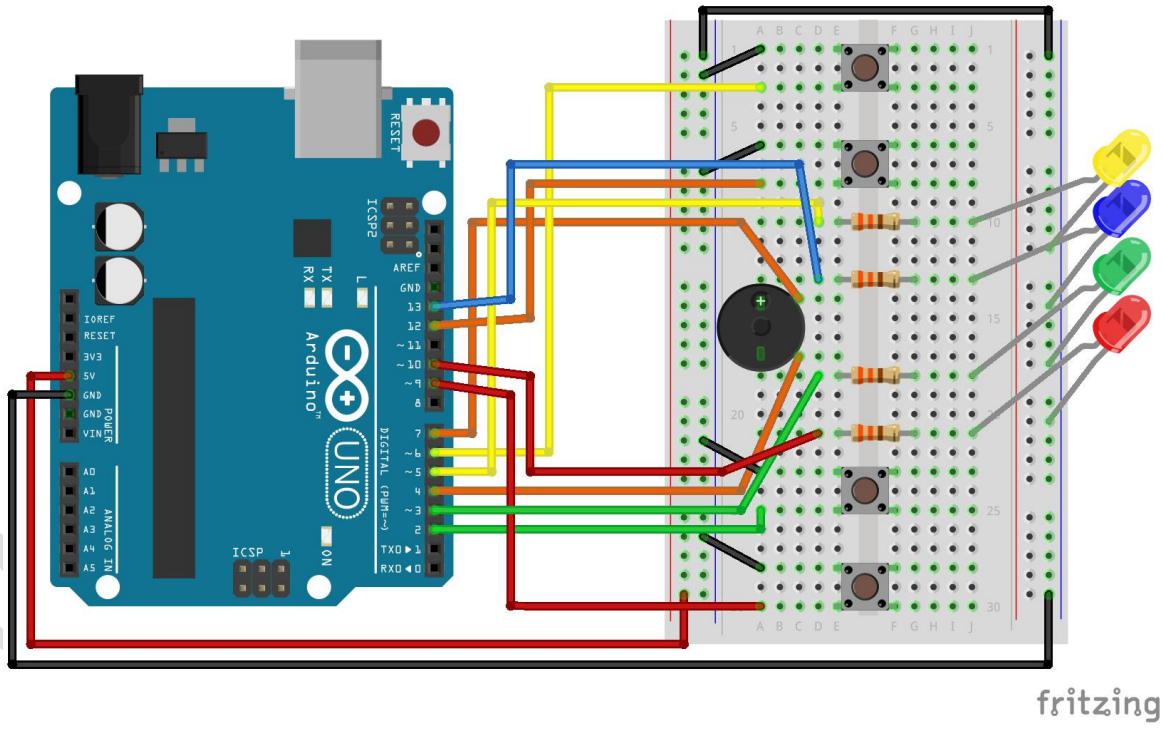
Mygtukai
x4

Jungiamieji laidai
x17

Pjezo skambutis
x1

5mm šviesos diodai (geltonas, mėlynas, žalias, raudonas)
x4

Surinkta grandinė



Kodas (nukopijuokite kodą į IDE atvertą langą)

```
/*
 *     ------------------------------------------------------
 *     |  Arduino rinkinio - IGSA pavyzdinis kodas:         |
 *     |  GRND-17: Žaidžiame – Reakcijos žaidimas           |
 *     ------------------------------------------------------
 *
 *
 * Šaltinis: SparkFun Inventor's Kit - Circuit 16: Simon Says
 *
 *
 * Programos kodas yra laisvai naudojmas bet kokiems tikslams.
 * Aplankykite - https://learn.sparkfun.com/tutorials/sik-experiment-guide-for-arduino---v32
 * Aplankykite - http://www.arduino.cc norėdami sužinoti daugiau apie Arduino.

 Simon Says is a memory game. Start the game by pressing one of the four buttons.
When a button lights up,
 press the button, repeating the sequence. The sequence will get longer and longer.
The game is won after
 13 rounds.

 Generates random sequence, plays music, and displays button lights.

 Simon tones from Wikipedia
 - A (red, upper left) - 440Hz - 2.272ms - 1.136ms pulse
 - a (green, upper right, an octave higher than A) - 880Hz - 1.136ms,
 0.568ms pulse
 - D (blue, lower left, a perfect fourth higher than the upper left)
 587.33Hz - 1.702ms - 0.851ms pulse
 - G (yellow, lower right, a perfect fourth higher than the lower left) -
 784Hz - 1.276ms - 0.638ms pulse

 Simon Says game originally written in C for the PIC16F88.
 Ported for the ATmega168, then ATmega328, then Arduino 1.0.
 Fixes and cleanup by Joshua Neal <joshua[at]trochotron.com>

*/

/************************************************
* Public Constants
************************************************/
#define NOTE_B0 31
#define NOTE_C1 33
#define NOTE_CS1 35
#define NOTE_D1 37
#define NOTE_DS1 39
#define NOTE_E1 41
#define NOTE_F1 44
#define NOTE_FS1 46
#define NOTE_G1 49
#define NOTE_GS1 52
#define NOTE_A1 55
#define NOTE_AS1 58
#define NOTE_B1 62
#define NOTE_C2 65
#define NOTE_CS2 69
#define NOTE_D2 73
#define NOTE_DS2 78
#define NOTE_E2 82
#define NOTE_F2 87
#define NOTE_FS2 93
#define NOTE_G2 98
#define NOTE_GS2 104
#define NOTE_A2 110
#define NOTE_AS2 117
```

```
#define NOTE_B2 123
#define NOTE_C3 131
#define NOTE_CS3 139
#define NOTE_D3 147
#define NOTE_DS3 156
#define NOTE_E3 165
#define NOTE_F3 175
#define NOTE_FS3 185
#define NOTE_G3 196
#define NOTE_GS3 208
#define NOTE_A3 220
#define NOTE_AS3 233
#define NOTE_B3 247
#define NOTE_C4 262
#define NOTE_CS4 277
#define NOTE_D4 294
#define NOTE_DS4 311
#define NOTE_E4 330
#define NOTE_F4 349
#define NOTE_FS4 370
#define NOTE_G4 392
#define NOTE_GS4 415
#define NOTE_A4 440
#define NOTE_AS4 466
#define NOTE_B4 494
#define NOTE_C5 523
#define NOTE_CS5 554
#define NOTE_D5 587
#define NOTE_DS5 622
#define NOTE_E5 659
#define NOTE_F5 698
#define NOTE_FS5 740
#define NOTE_G5 784
#define NOTE_GS5 831
#define NOTE_A5 880
#define NOTE_AS5 932
#define NOTE_B5 988
#define NOTE_C6 1047
#define NOTE_CS6 1109
#define NOTE_D6 1175
#define NOTE_DS6 1245
#define NOTE_E6 1319
#define NOTE_F6 1397
#define NOTE_FS6 1480
#define NOTE_G6 1568
#define NOTE_GS6 1661
#define NOTE_A6 1760
#define NOTE_AS6 1865
#define NOTE_B6 1976
#define NOTE_C7 2093
#define NOTE_CS7 2217
#define NOTE_D7 2349
#define NOTE_DS7 2489
#define NOTE_E7 2637
#define NOTE_F7 2794
#define NOTE_FS7 2960
#define NOTE_G7 3136
#define NOTE_GS7 3322
#define NOTE_A7 3520
#define NOTE_AS7 3729
#define NOTE_B7 3951
#define NOTE_C8 4186
#define NOTE_CS8 4435
#define NOTE_D8 4699
#define NOTE_DS8 4978

#define CHOICE_OFF      0 //Used to control LEDs
#define CHOICE_NONE     0 //Used to check buttons
#define CHOICE_RED  (1 << 0)
```

```
#define CHOICE_GREEN    (1 << 1)
#define CHOICE_BLUE (1 << 2)
#define CHOICE_YELLOW   (1 << 3)

#define LED_RED     10
#define LED_GREEN   3
#define LED_BLUE    13
#define LED_YELLOW  5

// Button pin definitions
#define BUTTON_RED    9
#define BUTTON_GREEN  2
#define BUTTON_BLUE   12
#define BUTTON_YELLOW 6

// Buzzer pin definitions
#define BUZZER1  4
#define BUZZER2  7

// Define game parameters
#define ROUNDS_TO_WIN      13 //Number of rounds to succesfully remember before you
win. 13 is do-able.
#define ENTRY_TIME_LIMIT   3000 //Amount of time to press a button before game times
out. 3000ms = 3 sec

#define MODE_MEMORY  0
#define MODE_BATTLE  1
#define MODE_BEEGEES 2

// Game state variables
byte gameMode = MODE_MEMORY; //By default, let's play the memory game
byte gameBoard[32]; //Contains the combination of buttons as we advance
byte gameRound = 0; //Counts the number of succesful rounds the player has made it
through

void setup()
{
  //Setup hardware inputs/outputs. These pins are defined in the hardware_versions
header file

  //Enable pull ups on inputs
  pinMode(BUTTON_RED, INPUT_PULLUP);
  pinMode(BUTTON_GREEN, INPUT_PULLUP);
  pinMode(BUTTON_BLUE, INPUT_PULLUP);
  pinMode(BUTTON_YELLOW, INPUT_PULLUP);

  pinMode(LED_RED, OUTPUT);
  pinMode(LED_GREEN, OUTPUT);
  pinMode(LED_BLUE, OUTPUT);
  pinMode(LED_YELLOW, OUTPUT);

  pinMode(BUZZER1, OUTPUT);
  pinMode(BUZZER2, OUTPUT);

  //Mode checking
  gameMode = MODE_MEMORY; // By default, we're going to play the memory game

  // Check to see if the lower right button is pressed
  if (checkButton() == CHOICE_YELLOW) play_beegees();

  // Check to see if upper right button is pressed
  if (checkButton() == CHOICE_GREEN)
  {
    gameMode = MODE_BATTLE; //Put game into battle mode

    //Turn on the upper right (green) LED
    setLEDs(CHOICE_GREEN);
    toner(CHOICE_GREEN, 150);
```

```
    setLEDs(CHOICE_RED | CHOICE_BLUE | CHOICE_YELLOW); // Turn on the other LEDs
until you release button

    while(checkButton() != CHOICE_NONE) ; // Wait for user to stop pressing button

    //Now do nothing. Battle mode will be serviced in the main routine
  }

  play_winner(); // After setup is complete, say hello to the world
}

void loop()
{
  attractMode(); // Blink lights while waiting for user to press a button

  // Indicate the start of game play
  setLEDs(CHOICE_RED | CHOICE_GREEN | CHOICE_BLUE | CHOICE_YELLOW); // Turn all LEDs
on
  delay(1000);
  setLEDs(CHOICE_OFF); // Turn off LEDs
  delay(250);

  if (gameMode == MODE_MEMORY)
  {
    // Play memory game and handle result
    if (play_memory() == true)
      play_winner(); // Player won, play winner tones
    else
      play_loser(); // Player lost, play loser tones
  }

  if (gameMode == MODE_BATTLE)
  {
    play_battle(); // Play game until someone loses

    play_loser(); // Player lost, play loser tones
  }
}

//-======-======-======-======-======-======-======-======-======-======-==
//The following functions are related to game play only

// Play the regular memory game
// Returns 0 if player loses, or 1 if player wins
boolean play_memory(void)
{
  randomSeed(millis()); // Seed the random generator with random amount of millis()

  gameRound = 0; // Reset the game to the beginning

  while (gameRound < ROUNDS_TO_WIN)
  {
    add_to_moves(); // Add a button to the current moves, then play them back

    playMoves(); // Play back the current game board

    // Then require the player to repeat the sequence.
    for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
    {
      byte choice = wait_for_button(); // See what button the user presses

      if (choice == 0) return false; // If wait timed out, player loses

      if (choice != gameBoard[currentMove]) return false; // If the choice is
incorect, player loses
    }

    delay(1000); // Player was correct, delay before playing moves
  }
```

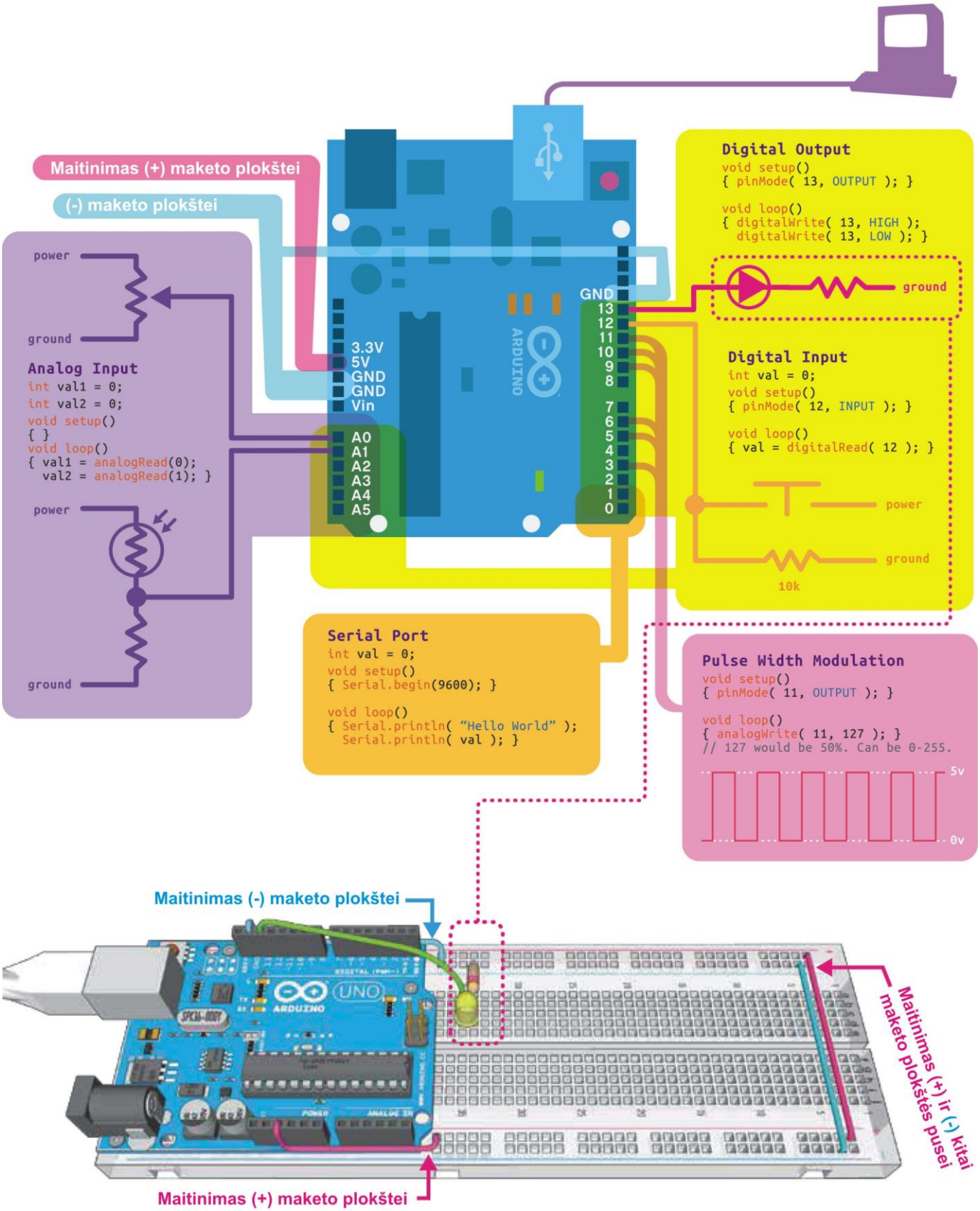
```
  return true; // Player made it through all the rounds to win!
}

// Play the special 2 player battle mode
// A player begins by pressing a button then handing it to the other player
// That player repeats the button and adds one, then passes back.
// This function returns when someone loses
boolean play_battle(void)
{
  gameRound = 0; // Reset the game frame back to one frame

  while (1) // Loop until someone fails
  {
    byte newButton = wait_for_button(); // Wait for user to input next move
    gameBoard[gameRound++] = newButton; // Add this new button to the game array

    // Then require the player to repeat the sequence.
    for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
    {
      byte choice = wait_for_button();

      if (choice == 0) return false; // If wait timed out, player loses.

      if (choice != gameBoard[currentMove]) return false; // If the choice is
incorect, player loses.
    }

    delay(100); // Give the user an extra 100ms to hand the game to the other player
  }

  return true; // We should never get here
}

// Plays the current contents of the game moves
void playMoves(void)
{
  for (byte currentMove = 0 ; currentMove < gameRound ; currentMove++)
  {
    toner(gameBoard[currentMove], 150);

    // Wait some amount of time between button playback
    // Shorten this to make game harder
    delay(150); // 150 works well. 75 gets fast.
  }
}

// Adds a new random button to the game sequence, by sampling the timer
void add_to_moves(void)
{
  byte newButton = random(0, 4); //min (included), max (exluded)

  // We have to convert this number, 0 to 3, to CHOICEs
  if(newButton == 0) newButton = CHOICE_RED;
  else if(newButton == 1) newButton = CHOICE_GREEN;
  else if(newButton == 2) newButton = CHOICE_BLUE;
  else if(newButton == 3) newButton = CHOICE_YELLOW;

  gameBoard[gameRound++] = newButton; // Add this new button to the game array
}

//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
//The following functions control the hardware

// Lights a given LEDs
// Pass in a byte that is made up from CHOICE_RED, CHOICE_YELLOW, etc
void setLEDs(byte leds)
{
  if ((leds & CHOICE_RED) != 0)
```

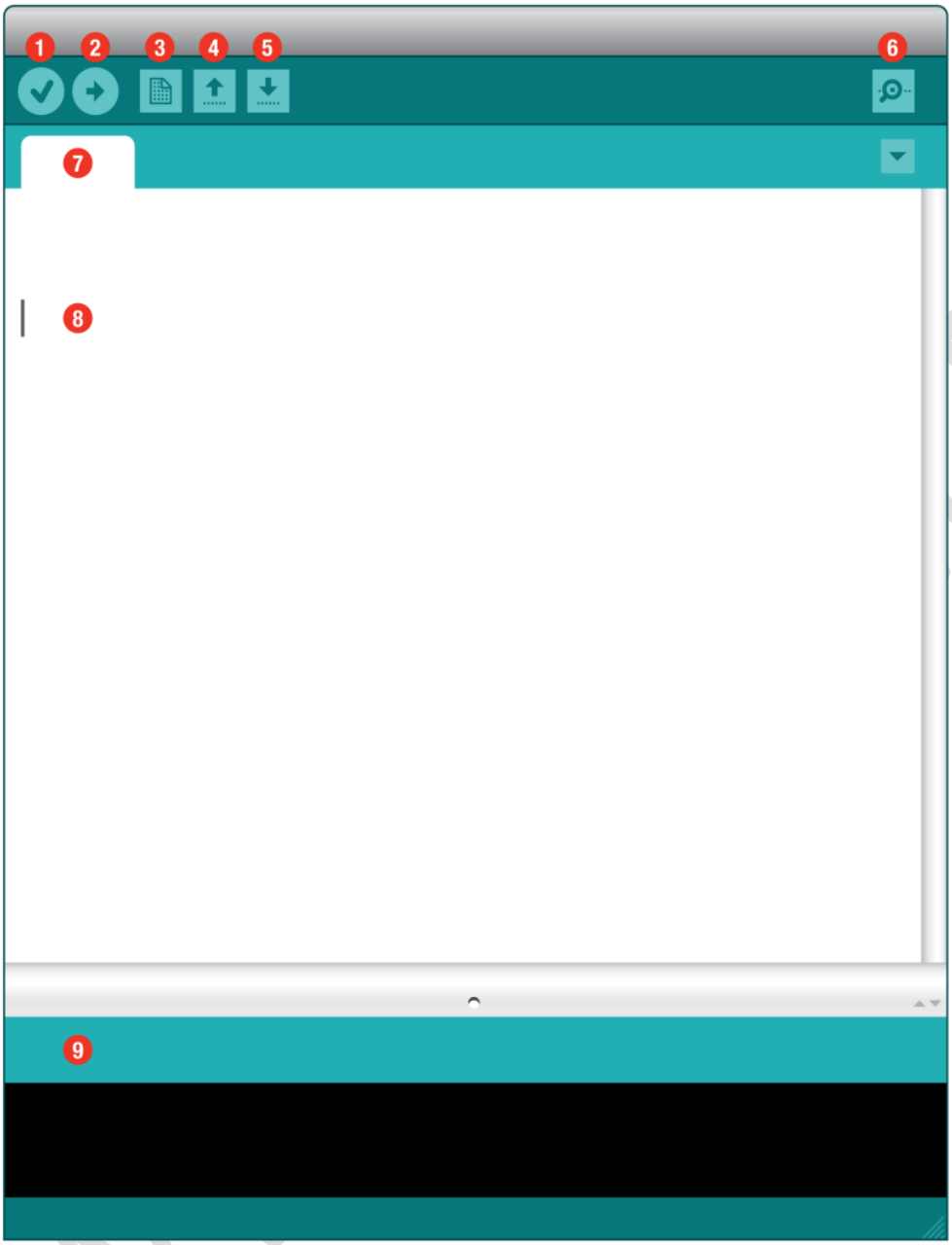
```
    digitalWrite(LED_RED, HIGH);
  else
    digitalWrite(LED_RED, LOW);

  if ((leds & CHOICE_GREEN) != 0)
    digitalWrite(LED_GREEN, HIGH);
  else
    digitalWrite(LED_GREEN, LOW);

  if ((leds & CHOICE_BLUE) != 0)
    digitalWrite(LED_BLUE, HIGH);
  else
    digitalWrite(LED_BLUE, LOW);

  if ((leds & CHOICE_YELLOW) != 0)
    digitalWrite(LED_YELLOW, HIGH);
  else
    digitalWrite(LED_YELLOW, LOW);
}

// Wait for a button to be pressed.
// Returns one of LED colors (LED_RED, etc.) if successful, 0 if timed out
byte wait_for_button(void)
{
  long startTime = millis(); // Remember the time we started the this loop

  while ( (millis() - startTime) < ENTRY_TIME_LIMIT) // Loop until too much time has
passed
  {
    byte button = checkButton();

    if (button != CHOICE_NONE)
    {
      toner(button, 150); // Play the button the user just pressed

      while(checkButton() != CHOICE_NONE) ;  // Now let's wait for user to release
button

      delay(10); // This helps with debouncing and accidental double taps

      return button;
    }

  }

  return CHOICE_NONE; // If we get here, we've timed out!
}

// Returns a '1' bit in the position corresponding to CHOICE_RED, CHOICE_GREEN, etc.
byte checkButton(void)
{
  if (digitalRead(BUTTON_RED) == 0) return(CHOICE_RED);
  else if (digitalRead(BUTTON_GREEN) == 0) return(CHOICE_GREEN);
  else if (digitalRead(BUTTON_BLUE) == 0) return(CHOICE_BLUE);
  else if (digitalRead(BUTTON_YELLOW) == 0) return(CHOICE_YELLOW);

  return(CHOICE_NONE); // If no button is pressed, return none
}

// Light an LED and play tone
// Red, upper left:     440Hz - 2.272ms - 1.136ms pulse
// Green, upper right:  880Hz - 1.136ms - 0.568ms pulse
// Blue, lower left:    587.33Hz - 1.702ms - 0.851ms pulse
// Yellow, lower right: 784Hz - 1.276ms - 0.638ms pulse
void toner(byte which, int buzz_length_ms)
{
  setLEDs(which); //Turn on a given LED

  //Play the sound associated with the given LED
```

```
  switch(which)
  {
  case CHOICE_RED:
    buzz_sound(buzz_length_ms, 1136);
    break;
  case CHOICE_GREEN:
    buzz_sound(buzz_length_ms, 568);
    break;
  case CHOICE_BLUE:
    buzz_sound(buzz_length_ms, 851);
    break;
  case CHOICE_YELLOW:
    buzz_sound(buzz_length_ms, 638);
    break;
  }

  setLEDs(CHOICE_OFF); // Turn off all LEDs
}

// Toggle buzzer every buzz_delay_us, for a duration of buzz_length_ms.
void buzz_sound(int buzz_length_ms, int buzz_delay_us)
{
  // Convert total play time from milliseconds to microseconds
  long buzz_length_us = buzz_length_ms * (long)1000;

  // Loop until the remaining play time is less than a single buzz_delay_us
  while (buzz_length_us > (buzz_delay_us * 2))
  {
    buzz_length_us -= buzz_delay_us * 2; //Decrease the remaining play time

    // Toggle the buzzer at various speeds
    digitalWrite(BUZZER1, LOW);
    digitalWrite(BUZZER2, HIGH);
    delayMicroseconds(buzz_delay_us);

    digitalWrite(BUZZER1, HIGH);
    digitalWrite(BUZZER2, LOW);
    delayMicroseconds(buzz_delay_us);
  }
}

// Play the winner sound and lights
void play_winner(void)
{
  setLEDs(CHOICE_GREEN | CHOICE_BLUE);
  winner_sound();
  setLEDs(CHOICE_RED | CHOICE_YELLOW);
  winner_sound();
  setLEDs(CHOICE_GREEN | CHOICE_BLUE);
  winner_sound();
  setLEDs(CHOICE_RED | CHOICE_YELLOW);
  winner_sound();
}

// Play the winner sound
// This is just a unique (annoying) sound we came up with, there is no magic to it
void winner_sound(void)
{
  // Toggle the buzzer at various speeds
  for (byte x = 250 ; x > 70 ; x--)
  {
    for (byte y = 0 ; y < 3 ; y++)
    {
      digitalWrite(BUZZER2, HIGH);
      digitalWrite(BUZZER1, LOW);
      delayMicroseconds(x);

      digitalWrite(BUZZER2, LOW);
      digitalWrite(BUZZER1, HIGH);
```

```
      delayMicroseconds(x);
    }
  }
}

// Play the loser sound/lights
void play_loser(void)
{
  setLEDs(CHOICE_RED | CHOICE_GREEN);
  buzz_sound(255, 1500);

  setLEDs(CHOICE_BLUE | CHOICE_YELLOW);
  buzz_sound(255, 1500);

  setLEDs(CHOICE_RED | CHOICE_GREEN);
  buzz_sound(255, 1500);

  setLEDs(CHOICE_BLUE | CHOICE_YELLOW);
  buzz_sound(255, 1500);
}

// Show an "attract mode" display while waiting for user to press button.
void attractMode(void)
{
  while(1)
  {
    setLEDs(CHOICE_RED);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_BLUE);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_GREEN);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;

    setLEDs(CHOICE_YELLOW);
    delay(100);
    if (checkButton() != CHOICE_NONE) return;
  }
}

//-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
// The following functions are related to Beegees Easter Egg only

// Notes in the melody. Each note is about an 1/8th note, "0"s are rests.
int melody[] = {
  NOTE_G4, NOTE_A4, 0, NOTE_C5, 0, 0, NOTE_G4, 0, 0, 0,
  NOTE_E4, 0, NOTE_D4, NOTE_E4, NOTE_G4, 0,
  NOTE_D4, NOTE_E4, 0, NOTE_G4, 0, 0,
  NOTE_D4, 0, NOTE_E4, 0, NOTE_G4, 0, NOTE_A4, 0, NOTE_C5, 0};

int noteDuration = 115; // This essentially sets the tempo, 115 is just about right
for a disco groove :)
int LEDnumber = 0; // Keeps track of which LED we are on during the beegees loop

// Do nothing but play bad beegees music
// This function is activated when user holds bottom right button during power up
void play_beegees()
{
  //Turn on the bottom right (yellow) LED
  setLEDs(CHOICE_YELLOW);
  toner(CHOICE_YELLOW, 150);

  setLEDs(CHOICE_RED | CHOICE_GREEN | CHOICE_BLUE); // Turn on the other LEDs until
you release button
```

```
  while(checkButton() != CHOICE_NONE) ; // Wait for user to stop pressing button

  setLEDs(CHOICE_NONE); // Turn off LEDs

  delay(1000); // Wait a second before playing song

  digitalWrite(BUZZER1, LOW); // setup the "BUZZER1" side of the buzzer to stay low,
while we play the tone on the other pin.

  while(checkButton() == CHOICE_NONE) //Play song until you press a button
  {
    // iterate over the notes of the melody:
    for (int thisNote = 0; thisNote < 32; thisNote++) {
      changeLED();
      tone(BUZZER2, melody[thisNote],noteDuration);
      // to distinguish the notes, set a minimum time between them.
      // the note's duration + 30% seems to work well:
      int pauseBetweenNotes = noteDuration * 1.30;
      delay(pauseBetweenNotes);
      // stop the tone playing:
      noTone(BUZZER2);
    }
  }
}

// Each time this function is called the board moves to the next LED
void changeLED(void)
{
  setLEDs(1 << LEDnumber); // Change the LED

  LEDnumber++; // Goto the next LED
  if(LEDnumber > 3) LEDnumber = 0; // Wrap the counter if needed
}
```


Maitinimas (+) maketo plokštei
(-) maketo plokštei
power
ground
Analog Input
int val1 = 0;
int val2 = 0;
void setup()
{ }
void loop()
{ val1 = analogRead(0);
val2 = analogRead(1); }
power
ground
3.3V
5V
GND
GND
Vin
A0
A1
A2
A3
A4
A5
GND
13
12
11
10
9
8
7
6
5
4
3
2
1
0
Digital Output
void setup()
{ pinMode(13, OUTPUT); }
void loop()
{ digitalWrite(13, HIGH);
digitalWrite(13, LOW); }
ground
Digital Input
int val = 0;
void setup()
{ pinMode(12, INPUT); }
void loop()
{ val = digitalRead(12); }
power
ground
10k
Serial Port
int val = 0;
void setup()
{ Serial.begin(9600); }
void loop()
{ Serial.println("Hello World");
Serial.println(val); }
Pulse Width Modulation
void setup()
{ pinMode(11, OUTPUT); }
void loop()
{ analogWrite(11, 127); }
// 127 would be 50%. Can be 0-255.
5v
0v
Maitinimas (-) maketo plokštei
Maitinimas (+) maketo plokštei
Maitinimas (+) ir (-) kitai maketo plokštės pusei



1. **Verify:** patikrina programinį kodą, kurį Jūs parašėte srityje Nr. 8. Paspaudus Jums bus parodyta kokias klaidas padarėte ir kurioje vietoje. Jei klaidų nėra Jums apie tai bus pranešta.
2. **Upload:** perkelia programinį kodą į Arduino plokštę. Kai nuspausite mygtuką ant plokštės esančios LED TX ir RX mirksės – tai reiškia kad kodas įkeliamas.
3. **New:** atidaromas naujas, tuščias programinio kodo langas.
4. **Open:** atidarys Jūsų jau sukurtą programinį kodą iš failo kompiuteryje. Programinio kodo failas vadinamas “Sketch”.
5. **Save:** Išsaugos Jūsų sukurtą programinį kodą į failą kompiuteryje.
6. **Serial Monitor:** atidarys serijinio ryšio langą kuriame galėsite matyti kokią informaciją Arduino rodo.
7. **Sketch name:** Jūsų failo pavadinimas su kuriuo Jūsų dabar dirbate.
8. **Code area:** vieta kurioje rašomas programinis kodas.
9. **Message area:** vieta kur Arduino IDE aplinka Jus informuoja apie klaidas ir problemas.

Arduino programavimui naudojama C kalba. Ši trumpa apžvalga skirta tiems, kas turi bet kurios kitos programavimo kalbos pagrindus ir supažindina su C kalbos ypatumais.
Jei kai kurios sąvokos atrodys neįprastos, nenusiminkite, dirbant daug kas taps labiau suprantama. Daugiau informacijos ieškokite anglų k. http://Arduino.cc puslapyje arba http://arduino.ru/Reference rusų kalba.

Struktūra

Kiekviena Arduino programa (kūrėjų dar vadinama *sketch* – piešinuku, nes Arduino – tai C kalba "menininkams") privalo turėti dvi funkcijas (jos kartais dar vadinamos paprogramėmis - *routines*).

`void setup() { }`

kodas esantis tarp riestinių skliaustų `{ }` vykdomas vieną kartą, kai programa startuoja.

`void loop() { }`

Ši funkcija vykdoma po nustatymų funkcijos. Tai nuolat besikartojantis ciklas.

Sintaksė

Vienas iš sunkiau įsisavinamų C kalbos elementų yra jos sintaksė. Bet būtent sintaksė suteikia C kalbai ypač daug galimybių.

`//` Vienos eilutės komentarai

Programuotojai dažnai rašo kodo paaiškinimus, kurie skirti jiems patiems ir bet kam kas skaitys jų programą. Norėdami parašyti trumpą komentarą tiesiog įveskite `//` ir toje eilutėje parašytą tekstą programa ignoruos.

`/* */` kelių eilučių komentaras

Jei reikia ką nors pakomentuoti plačiau, surinkite `/*` pradžioje ir `*/` pabaigoje teksto. Tarp šių simbolių esantis komentaras bus praleistas vykdant programą.

`{ }` Riestiniai skliaustai

Jie pažymi kur prasideda ir kur baigiasi tam tikras kodo segmentas. Naudojami funkcijose, taip pat cikluose.

`;` Kabliataškis

Kiekvienos kodo eilutės pabaigoje turi būti kabliataškis. Praleisti kabliataškiai -dažna programos kodo kompiliavimo klaidų priežastis.

Kintamieji

Programa tėra instrukcija kaip manipuliuoti duomenimis tam tikru būdu. Kintamieji padeda tai daryti paprasčiau.

`int (integer)`

Tai kintamųjų darbinis arkliukas. Jis naudojama sveikiesiem (be kablelių) skaičiams nuo -32 768 iki 37 767. Skaičiai užima 2 baitus (16 bitų) atminties.

`long`

naudojamas dideliems skaičiams, kai neužtenka `int`. Užima 4 baitus (32 bitus) RAM atminties.
Talpina skaičius nuo -2 147 483 648 iki 2 147 483 647.

`boolean`

Paprastas kintamasis kurio reikšmės gali būti tiesa (True) arba netiesa (False). naudingas tuo, kad užima tik 1 RAM atminties bitą.

`float`

naudojamas operacijoms su realiaisiais skaičiais. Užima 4 baitus (32 bitus) RAM atminties ir apima skaičius nuo -3.4028235E+38 iki 3.4028235E+38.

`char`

Talpina ASCII kodo simbolius (raides, skaičius, simbolius). (Pvz. A = 65). Užima 1 baitą (8 bitus) RAM atminties. Arduino naudoja tekstą kaip `char` simbolių masyvus.

Aritmetika

Operatoriai naudojami veiksmams su skaičiais, labai panašiai kaip įprastoje matematikoje.

= (priskyrimas) suteikia tam tikrą reikšmę (dažniausiai kintamajam). Pvz. `x = 10 * 2` , dabar x reikšmė bus lygi 20.
`%` (modulis) atskiria dalybos veiksmo liekaną. Pvz. `12 % 10` bus lygu 2.
+ (sudėtis)
– (atimtis)
* (daugyba)
/ (dalyba)

Palyginimo operatoriai naudojami loginiam palyginimui.

`==` (lygu), pvz. veiksmo `12 == 10` rezultatas bus FALSE - netiesa; `12 == 12` rezultatas bus TRUE - tiesa.
`!=` (nelygu) Pvz. `12 != 10` yra TRUE - tiesa, `12 != 12` yra FALSE - netiesa.
`<` (mažiau negu) Pvz. `12 < 10` yra FALSE - netiesa; `12 < 14` yra TRUE - tiesa.
`>` (daugiau negu) Pvz. `12 > 10` yra TRUE - tiesa; `12 > 12` yra FALSE - netiesa.

Programos valdymas

Programos veikia tam tikra seka. Priklausomai nuo tam tikrų sąlygų vykdymo seka gali keistis.

```
if(sąlyga - condition){ }
else if (sąlyga - condition){ }
else { }
```

Čia tarp riestinių skliaustų `{ }` esantis kodas bus vykdomas jei sąlygos kodo reikšmė bus TRUE. Jei sąlygos kodo reikšmė bus FALSE - netiesa, toliau bus tikrinama sąlyga po `else if` ir jei sąlygos reikšmė yra FALSE, galiausiai vykdomas kodas `{ }` skliausteliuose po `else`.

Skaitmeninis įvedimas / išvedimas

```
pinMode(pin, mode);
```

Naudojamas nustatyti jungčiai, `pin` yra jungties numeris nuo 0 iki 19 (analoginė 0 - 5 yra 14 - 19). Veikimo režimas mode gali būti įvestis INPUT arba išvestis OUTPUT.

```
digitalWrite(pin, value);
```

Kai jungties veikimo režimas nustatomas kaip OUTPUT - išvestis, ją galima įjungti paduodant +5V įtampą, reikšmė value HIGH. Nustačius reikšmę LOW įtampa išjungiama (žemė).

```
int digitalRad(pin);
```

Įvedus jungties pin veikimo režimą kaip įvestį INPUT galime nustatyti ar į ją paduodama +5V įtampa (HIGH reikšmė), ar ne (LOW reikšmė).

Analoginis įvedimas / išvedimas

Arduino yra skaitmeninė platforma, bet, taikydami tam tikras gudrybes, galime naudoti ir analoginius signalus.

```
int analogWrite(pin, value);
```

Kai kurios Arduino jungtys gali būti naudojamos pulsams us moduliacija (3, 5, 6, 9, 10, 11).
Jungtis įjungiama ir išjungiama labai greitai, tai imituojant analoginę išvestį. Reikšmė `value` gali būti bet koks skaičius nuo 0 (0% signalo ciklo ~0V) ir 225 (100% signalo ciklo ~5V).

```
int analogRead(pin);
```

Kai analoginės išvesties jungtis yra nustatyta įvesties režimu galime nuskaityti įtampą. Reikšmė `value` tarp 0 (0 V) ir 1024 (5V).

Arduino produktų/modelių charakteristikos

Charakteristika \ Modelis	UNO Rev3	UNO SMD Rev3	Mega 2560 Rev3	Leonardo
Taktinis dažnis	16 Mhz	16 Mhz	16Mhz	16 Mhz
Flash atmintis	32 KB	32 KB	256 KB	32 KB
Mikrokontrolerio tipas	ATmega328	ATmega328	ATmega2560	ATmega32u4
Mikrokontrolerio jungtis	DIP	Lituojama tiesiogiai	DIP	Lituojama tiesiogiai
Skaitmeninės jungtys	14	14	54	20
Analoginės jungtys	6	6	16	12

Charakteristika \ Modelis	Mega ADK Rev3	Due	Micro	Mini 05
Taktinis dažnis	16 Mhz	84 Mhz	16 Mhz	16 Mhz
Flash atmintis	256 KB	512 KB	32 KB	32 KB
Mikrokontrolerio tipas	ATmega2560	32bit ARM AT91SAM3X8E	ATmega32u4	ATmega328
Mikrokontrolerio jungtis	Lituojama tiesiogiai	Lituojama tiesiogiai	Lituojama tiesiogiai	Lituojama tiesiogiai
Skaitmeninės jungtys	54	54	20	14
Analoginės jungtys	16	12	12	8

ANODAS
electronics