

OMA LIKUMS. DIODES. C++ DATU TIPI.

1.1 Praktiskais darbs #1 (Blink.ino)

Vajadzīgās detaļas: Arduino kontrolieris, USB vads (adapteris no USB-A uz USB-B).

1. Palaist Arduino IDE. Atvērt izvēlni **File > Examples > 01.Basics > Blink**. Saglabāt **Blink.ino** stāp saviem failiem ar **File > Save As**. Piemēram, C:\Users\MansVards\Arduino\Blink\Blink.ino (skripts jāliek direktorijā ar to pašu nosaukumu).
2. Izvilk no kastītes kontrolieri un USB vadu. Vienu USB galu pievienot datoram, otru galu pievienot Arduino platei. Pārliecināties, ka uz plates iedegas diode **ON**.
3. Uzstādīt kontroliera tipu un portu. Arduino IDE atvērt **Tools > Board > Arduino AVR Boards > Arduino Uno** un arī **Tools > Port > usb.serial** (vai **Tools > Port > Com**, ja nav minēts USB).
4. Kompilet programmiņu ar **Sketch > Verify/Compile**.
5. Nosūtīt programmiņu uz Arduino: **Sketch > Upload**. Pārliecināties, ka lampiņa **L** (cita diode uz Arduino plates) sāk mirkšķināt, ik pēc sekundes ieslēdzoties vai izslēdzoties.

```
void setup() {
    pinMode(LED_BUILTIN, OUTPUT); // uzstāda 13.kontaktu kā izvadi
}

void loop() { // Šis cikls atkārtojas, kamēr Arduino ir ieslēgts
    digitalWrite(LED_BUILTIN, HIGH); // ieslēdz LED ar spriegumu HIGH uz 13. kontakta
    delay(1000); // gaida 1 sekundi,
    digitalWrite(LED_BUILTIN, LOW); // izslēdz LED ar spriegumu LOW uz 13.kontakta
    delay(1000); // gaida 1 sekundi.
}
```

6. LED_BUILTIN vienāds ar 13 (jo iebūvēto diodi iededz 13.kontakts uz plates). Pārliecinieties par to, LED_BUILTIN visur aizstājot ar skaitli 13 un nosūtot programmu uz Arduino vēlreiz.
 - **Pārbaudīt maksimālo spriegumu/strāvu:** Nepārsniegt atlautos voltus/ampērus. LED slēgt pie 5 V tikai kopā ar rezistoru.
 - **Lietot iezemējumu:** Katra ierīce ir savienota ar Arduino **GND**; nevajag veidot kēdes stāp nepazīstamiem Arduino kontaktiem. Nevajag saslēgt Arduino kontaktus uz īso.
 - **Atslēgt USB, ja maina kēdi:** Kēdi jāpārbauda pirms pieslēdz strāvu; nevajag pārsprauzt vadiņus tad, ja shēma ir pieslēgta USB vadam.
 - **Tīras rokas:** Rokām jābūt sausām un uz darba galda neturēt ēdamo. (Pēc nodarbības, pēc pieskaršanās metāla kontaktiem un lodējumiem vajag nomazgāt rokas.)

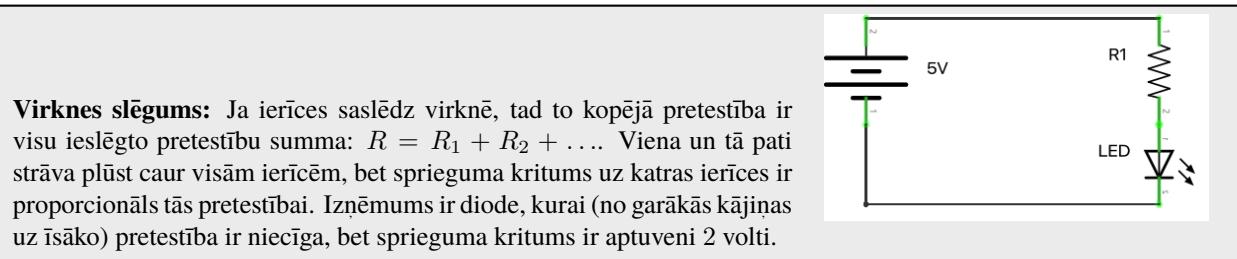
1.2 Elektrotehnika Arduino un LED saslēgšanai

Lai droši lietotu LED ar Arduino, jāizmanto rezistors. Tas neļauj LED caurlaist pārāk lielu strāvu, kas to var sabojāt.

Oma likums:

- **Spriegums (U):** Līdzīgi kā ūdens spiediens caurulē vai ūdenskrītuma augstums.
- **Strāvas stiprums (I):** Līdzīgi kā ūdens plūsmas ātrums.
- **Pretestība (R):** Pretojas strāvas plūsmai, līdzīgi kā caurules sašaurinājums ūdenim.

Oma likums: $I = \frac{U}{R}$. Maza pretestība nozīmē ļoti lielu strāvas stiprumu (*īsslēgums*).



Pretestības aprēķināšana:

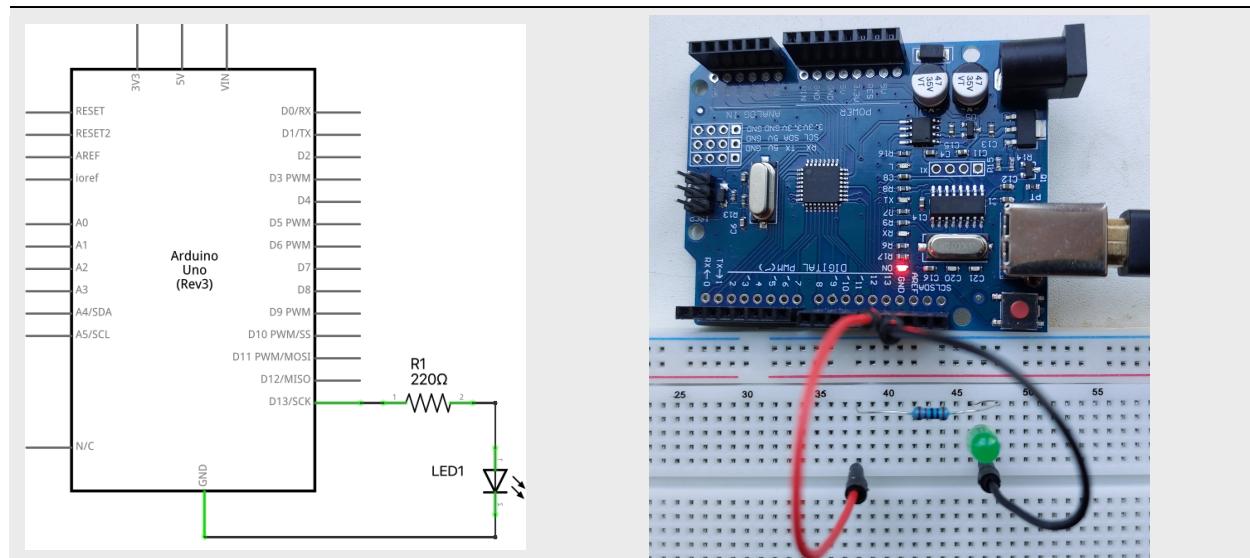
Ja pieslēdz LED ar 2 V sprieguma kritumu un maksimālo strāvu 20 mA (20 miliamperi jeb 0.020 A) Arduino platei ar 5 V spriegumu, no strāvas avota sprieguma jāatņem LED sprieguma kritums. Uz rezistoru R_1 spriegums ir $5\text{ V} - 2\text{ V} = 3\text{ V}$. Izmantojam Oma likumu $R = \frac{U}{I}$ un atrodam $R = \frac{3\text{ V}}{0.020\text{ A}} = 150\text{ }\Omega$.

Var izmantot arī $220\text{ }\Omega$ rezistoru no komplekta, kas vēl drusku samazinās strāvas stiprumu. Šādā kēdē LED droši darbojas kopā ar Arduino!

1.3 Praktiskais darbs #2: Ārēja LED diode (Blink.ino)

Vajadzīgās detalas: Arduino kontrolieris, USB vads, maketēšanas plate (*breadboard*), krāsaina LED diode, $220\text{ }\Omega$ rezistors, divi savienotājvadi M-M (ar adatiņām abos galos).

Šajā darbā scenārijs joprojām ir `Blink.ino`, bet pievienosim ārēju diodi.

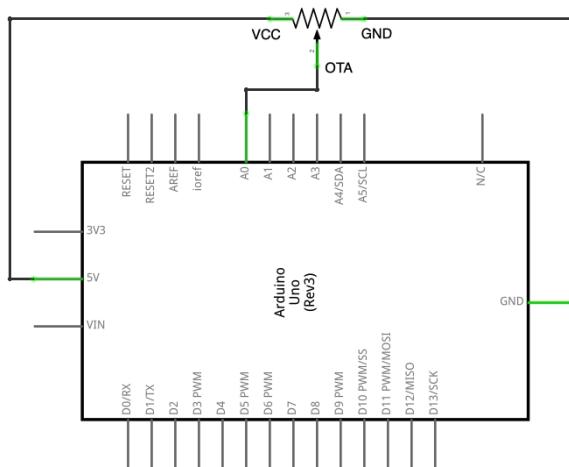


- Atvienot Arduino kontrolieri no USB vada. Saslēgt attēlā redzamo kēdi. LED garāko kājiņu likt tuvāk plusam, īsāko – tuvāk mīnusam. (**Uzmanību:** Pārliecinieties, ka diode un rezistors ir saslēgti virknē, lai nepārsniegtu maksimālo strāvas stiprumu uz ārējās LED.)
- Pievienot Arduino kontrolieri USB vadam. Pārliecināties, ka tagad mirkšķinās gan mazā diode **L** uz Arduino plates, gan arī F5 gaismas diode mūsu kēdē.

1.4 Praktiskais darbs #3: Potenciometrs (ReadingValues.ino)

Vajadzīgās detalas: Arduino kontrolieris, USB vads, potenciometrs, maketēšanas plate, trīs vadiņi M-F (ar adatinu vienā galā un kontaktligzdu otrā), divi vadiņi M-M (ar adatinām abos galos).

- Atvienot vadus no Arduino uz ārējo gaismas diodi LED. Ieslēgt potenciometru starp **GND** un **5V** (potenciometra **VCC**), izvadu **OTA** savieno ar Arduino **A0**.



- Izveidot jaunu skriptu, saglabāt to kā `ReadingValues.ino`

```
const int pinAnalog = A0;

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

void loop(void) {
    int valInt = analogRead(pinAnalog);
    double valU = (5. / 1023) * valInt;
    String separator = " spriegums:";
    Serial.print(valInt);
    Serial.print(" ");
    Serial.print(A0);
    Serial.print(separator);
    Serial.println(valU , 2);
    delay(100);
}
```

- Kompilēt un nosūtīt šo skriptu uz Arduino. Novērot strauju **RX** (seriālā porta) lampiņas mirkšķināšanu.

4. No Arduino IDE atvērt **Tools > Serial Monitor**. Pārliecināties, ka uz datora ekrāna parādās skaitli (skaitli no 0 līdz 1023 un spriegums no 0.0 līdz 5.0 voltiem). Pārvietot potenciometra regulatoru, novērot vērtību maiņu.
5. Atvērt **Tools > Serial Plotter**. Pārvietot regulatoru un novērot grafika izmaiņas.
6. Izmainīt Arduino skriptā $(5. / 1023) * \text{valInt}$ skaitli 5. ar 5 (bez punkta beigās) un nosūtīt to Arduino.
7. No jauna atvērt **Tools > Serial Monitor** un novērot tur redzamās sprieguma vērtības. Kāpēc tās visas ir 0?

Table 1: C++ datu tipi

| | | |
|-----------|-------------------------------------|-----------------------------------------------|
| const int | const int pinAnalog = A0; | Vesels konstants skaitlis (piemēram A0=14) |
| int | int valInt = analogRead(pinAnalog); | Potenciometra vesels stāvoklis no 0 līdz 1023 |
| double | double valU = (5. / 1023) * valInt; | Reāls skaitlis (spriegums U voltos) |
| String | String separator = " spriegums:"; | Strings jeb burtu virknīte |

1.5 Uzdevumi

Kas ir frekvence:

Frekvence raksturo, cik bieži notiek kāda darbība. Frekvenci mēra hercos. Piemēram, 50 Hz (50 hercu) frekvence nozīmē, ka kaut kas atkārtojas 50 reizes sekundē.

Piemērs:

Atradīsim frekvenci LED mirkšķināšanai piemērā **Blink.ino**. Tā kā pilns periods starp divām LED iemirgošanās reizēm ir $1 + 1 = 2$ sekundes, tad frekvence mirkšķināšanai ir $\frac{1}{2} \text{ s}^{-1} = 0.5 \text{ Hz}$ jeb 0.5 herci.

1.uzdevums:

Daži cilvēki ir jūtīgi pret signāllampiņu mirkšķināšanu noteiktās frekvencēs ([Photosensitive Epilepsy](#)). Īpaši problemātiskas mēdz būt 15 līdz 20 hercu frekvences. Kā jāizvēlas abi gaidīšanas laiki izsaukumos `delay(milliseconds)`, lai iegūtu šādu nekomfortablu diodes mirkšķināšanas frekvenci? Izmainiet Arduino skriptu **Blink.ino** no 2.praktiskā darba, lai krāsainā diode radītu šo frekvenci.

2.uzdevums:

Izveidot kēdi, kur gaismas diode sāk mirkšķināt tikai tad, kad potenciometrs ir aizbīdīts pāri pusei. To var pārbaudīt divos veidos:

- Skaitlis, ko nolasa no analogās ieejas A0 (`int valInt = analogRead(pinAnalog)`) ir virs 512 (t.i. pārsniedz $1023/2$ jeb pusi no maksimālās vērtības).
- *Potenciālu starpību* starp kontaktiem **GND** un **OTA** ir vismaz 2.5 V.

Jums var noderēt “if” operators. Piemēram,

```
if (valInt >= 512) {
    // mirkšķina diodi
}
else {
    // izslēdz diodi
}
```

Pabeigt šo skriptu. Saslēgt kēdi tā, lai potenciometra kontakts **OTA** rakstītu uz Arduino **A0**. Pieslēgt arī LED diodi (virknē ar rezistoru!) starp kontaktiem **GND** un **13**. Faktiski – kombinēt praktiskos darbus #2 un #3.

3.uzdevums:

Vai ir iespējama tāda kēde un Arduino skripts, kas ar potenciometru maina uz diodes padoto spriegumu no 0 V līdz 2 V?