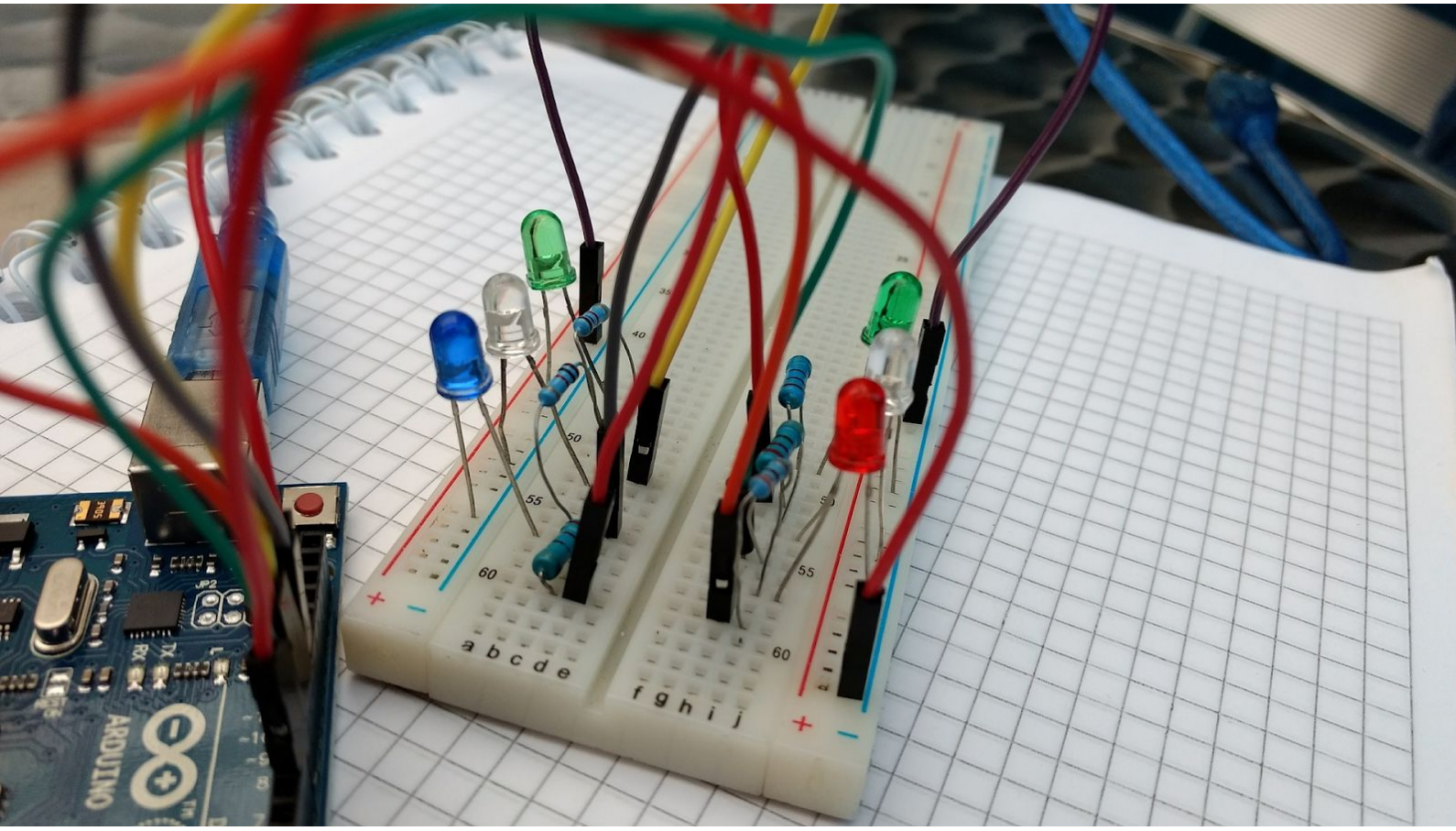
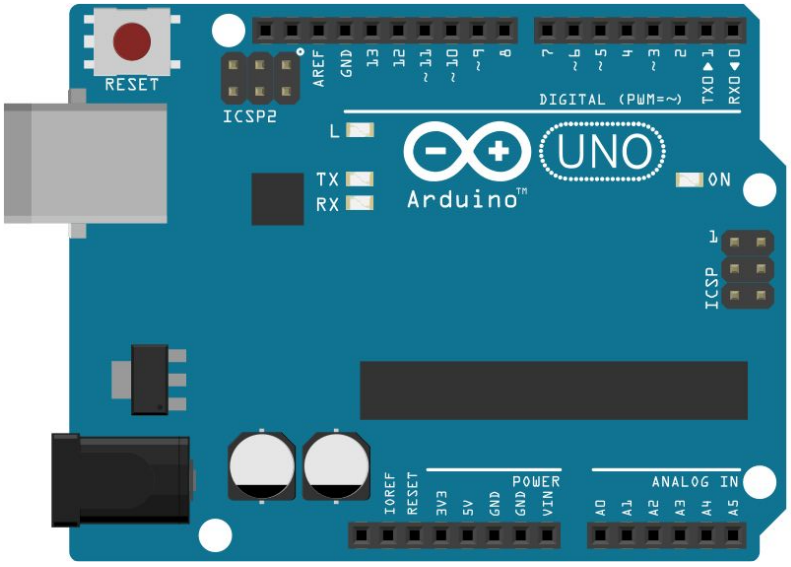
Yläkoulun Arduino-projektit





| Alajärven sivistystoimi 2019 | Nuoret Värkkärit -hanke |

KAIKKI TÄSSÄ KIRJASSA ESITETYT KYTKENNÄT JA VIRTAPIIRIT ON TARKOITETTU KÄYTETTÄVÄKSI **AINOASTAAN PARISTOLLA.**

ÄLÄ KOSKAAN TEE MITÄÄN KYTKENTÖJÄ TAI MITTAUKSIA PISTORASIASTA TULEVAAN VERKKOVIRTAAN. SE ON HENGENVAARALLISTA.

PISTORASIASTA SAATU SÄHKÖISKU AIHEUTTAA KUOLEMAN.

Kun lopetat kytkennän käytön, niin irrota paristo kytkennästä ja aseta eristävä teippi pariston napojen päälle.

Jos pariston napoja ei teipata, niin johtavan esineen vahingossa aiheuttama oikosulku voi kuumentaa pariston tai esineen vaarallisen kuumaksi. Tällaisessa tilanteessa akut voivat jopa räjähtää.

AINA, KUN ITSE RAKENNETTUA LAITETTA EI VALVOTA, TULEE PARISTO IRROITTAA SIITÄ JA PARISTON NAVAT SUOJATA TEIPILLÄ.

SISÄLLYSLUETTELO

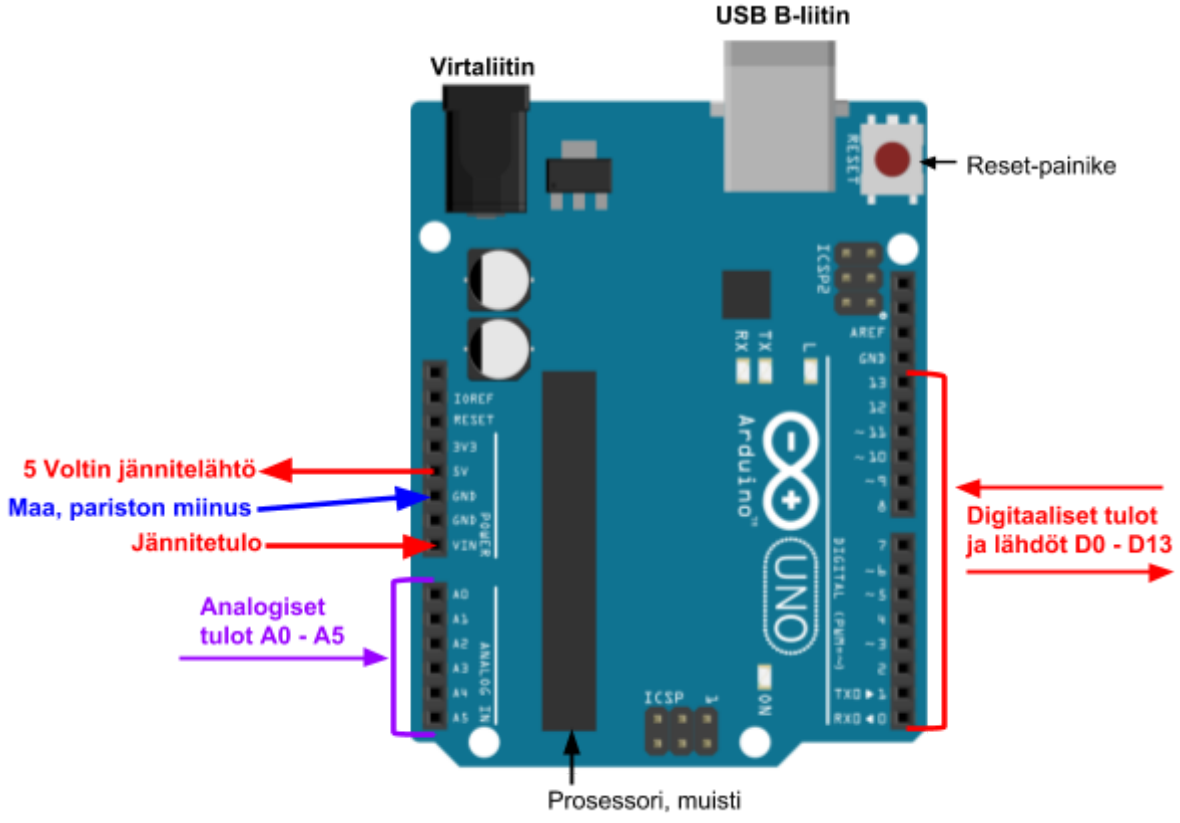
Arduino Uno mikro-ohjain

Digitaaliset tulot ja lähdöt

Kaikissa mikro-ohjaimissa on tulo- ja lähtöpinnejä. Tulopinneihin voidaan liittää esimerkiksi antureita, joiden lähettämä signaali kulkee tulopinnin kautta mikro-ohjaimeen käsiteltäväksi (esim. lämpötilaa mittaava anturi). Lähtöpinneihin liitetään ulkoisia virtapiirejä, joita ohjataan mikro-ohjaimeen laaditulla ohjelmalla. Yksinkertainen virtapiiri on esimerkiksi vastuksen ja LEDin muodostama kytkentä. Arduinon digitaalinen pinni D0-D13 asetetaan ohjelmakoodilla toimimaan joko tulona tai lähtönä.

Analogiset tulot

Analogiset tulot A0-A5 lukevat analogisen signaalin jännitteen 0-5 Volttia ja muuttavat sen jännitetasoa vastaavaksi luvuksi 1-1023. Analogiseen tuloon voidaan liittää esim. potentiometri (säätövastus) tai valoisuutta mittaava LDR-vastus. Tulet myöhemmin oppimaan kytkentöjen rakentamisen sekä ohjelmoinnin digitaalisiin lähtöihin ja tuloihin sekä analogisiin tuloihin. Tässä vaiheessa riittää, että tiedät missä nämä pinnit sijaitsevat mikro-ohjaimessa.



5 Voltin jännitelähtö ja maa

Useimmat itse rakennetut ulkoiset virtapiirit tarvitsevat käyttöjännitteen toimintaansa. Tämä käyttöjännite voidaan ottaa Arduinon **5 Voltin jännitelähdöstä**. Virta voi kulkea vain suljetussa virtapiirissä, joten ulkoiseen virtapiiriin tulee kytkeä lisäksi maa-johto. Maa-johto kytketään Arduinon pinniin **GND** (ground, maa). GND-pinniejä on Arduino Unossa kaikkiaan kolme ja ne kaikki toimivat samalla tavalla.

Jännitetulo Vin

Pinni **Vin** toimii Arduinon jännitetulona. Siihen voidaan kytkeä esim. 9 Voltin pariston plus-napa. Pariston miinus-napa kytketään Arduinon pinniin **GND**. Näin mikro-ohjainta on ohjelmoinnin jälkeen mahdollista käyttää ilman tietokonetta.

Virtaliitin

Virtaliittimeen voidaan kytkeä 9-12 Voltin käyttöjännite Arduinolle.

USB-liitin

Ohjelmointia varten Arduino liitetään tietokoneeseen USB-kaapelilla (B-urosliitin Arduinoon, A-urosliitin tietokoneeseen). Kun Arduino on liitetty USB-kaapelilla tietokoneeseen, niin se saa myös tarvitsemansa käyttöjännitteen tietokoneen USB-portista, eikä ulkoista paristoa silloin tarvita.

Windows-ohjelmointiympäristö

Lataus ja asennus

Suorita tämän kappaleen ohjeet vain, jos ohjelmointiympäristöä ei ole vielä asennettu käyttämääsi tietokoneeseen.

Arduinon ohjelmointiympäristö on ilmainen ja se löytyy osoitteesta https://www.arduino.cc/en/main/software

Klikkaa sivulla linkkiä **"Windows Installer, for Windows XP and up"**, lataa tiedosto tietokoneellesi ja asenna ohjelma. Tee ohjelman pikakuvake työpöydälle.

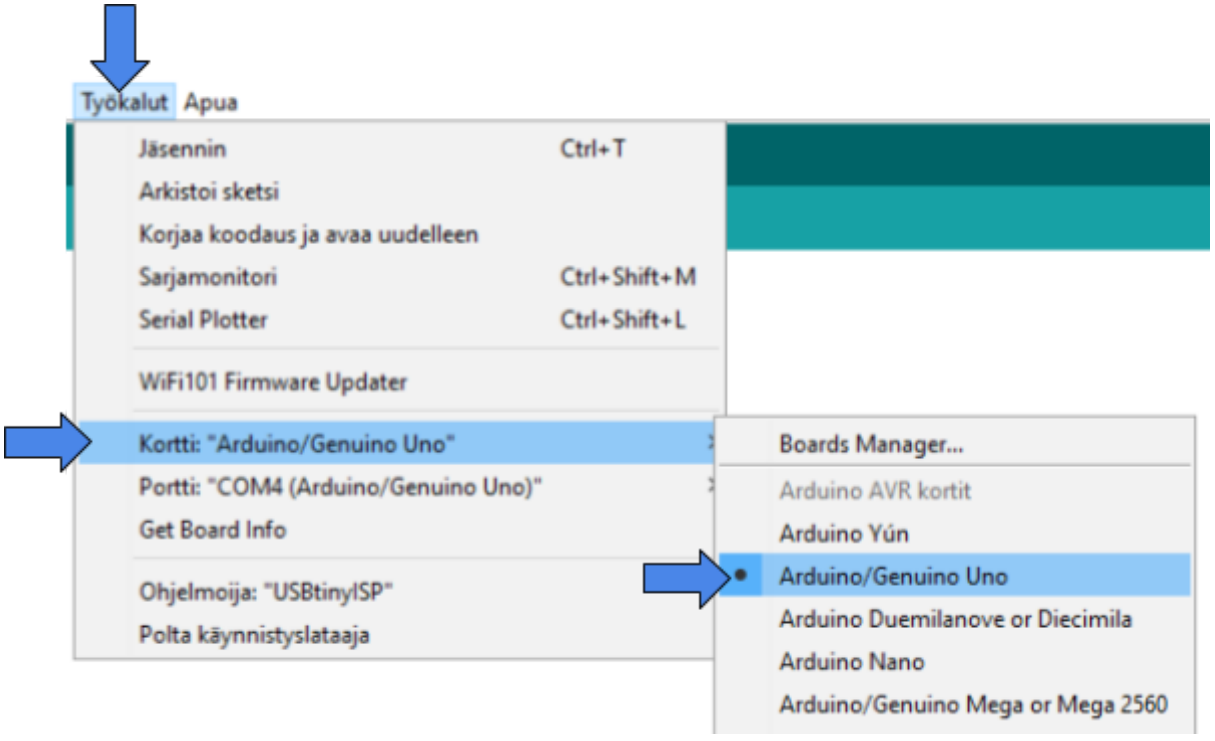
Ohjelmointiympäristön asetukset

Liitä Arduino Uno tietokoneeseen USB-kaapelilla ja käynnistä ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.

Käytettävän mikro-ohjaimen valinta

Arduino mikro-ohjaimia on useita erilaisia ja aluksi tulee valita käytettävä malli.

- Klikkaa valikkoa **Työkalut**
- Siirrä hiiri valinnan **Kortti: xxx** päälle
- Sivuun aukeaa uusi valikko, valitse siitä valinta **Arduino/Genuino Uno**



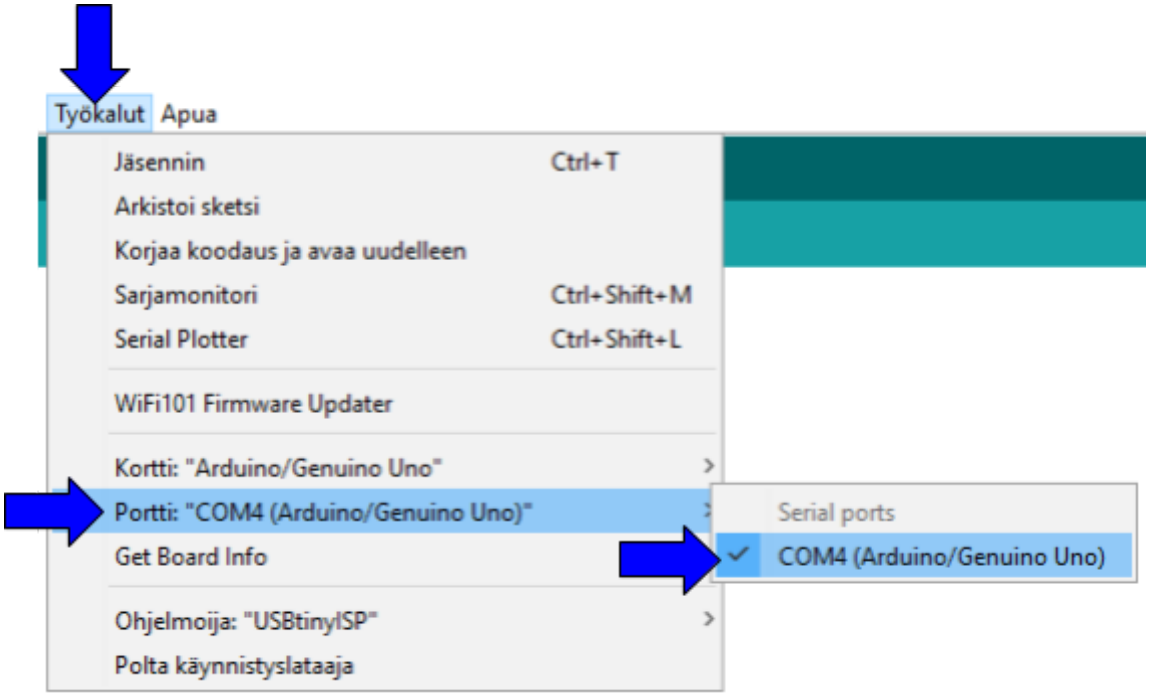
Nyt olet valinnut käytettäväksi mikro-ohjaimeksi Arduino Unon. Asetus säilyy ohjelmassa.

Sarjaportin valinta

Ohjelmointiympäristölle tulee kertoa vielä, mihin sarjaporttiin mikro-ohjain on kytketty.

- Klikkaa valikkoa **Työkalut**
- Siirrä hiiren osoitin valinnan **Portti: COMxxx** päälle
- Sivuun aukeaa uusi valikko, valitse siitä rivi, jossa tekstin COMx perässä lukee teksti **(Arduino/Genuino Uno)**

Tekstin **COM** perässä oleva luku vaihtelee, kuvassa se on **4**, eli kyseessä on sarjaportti **COM4**.



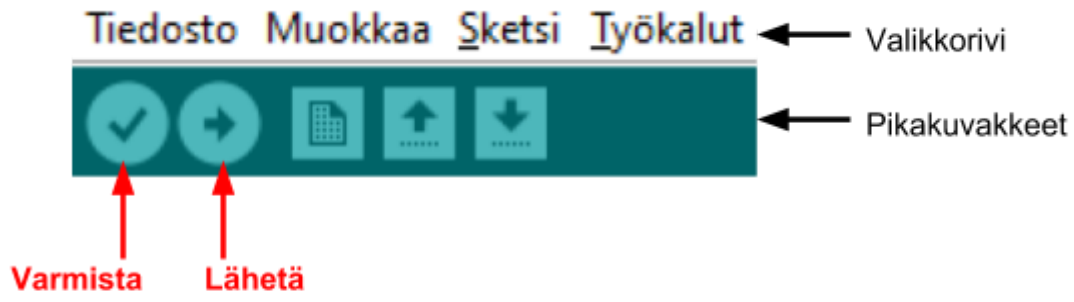
Nyt olet valinnut oikean sarjaportin.

Ohjelmointiympäristön käyttö

Kooditiedostojen tallennus- ja avaus-toiminnot tehdään Tiedosto-valikosta normaalin Windows-ohjelman mukaisesti.

Valikkorivin alla kaksi keskeistä **pikakuvaketta**:

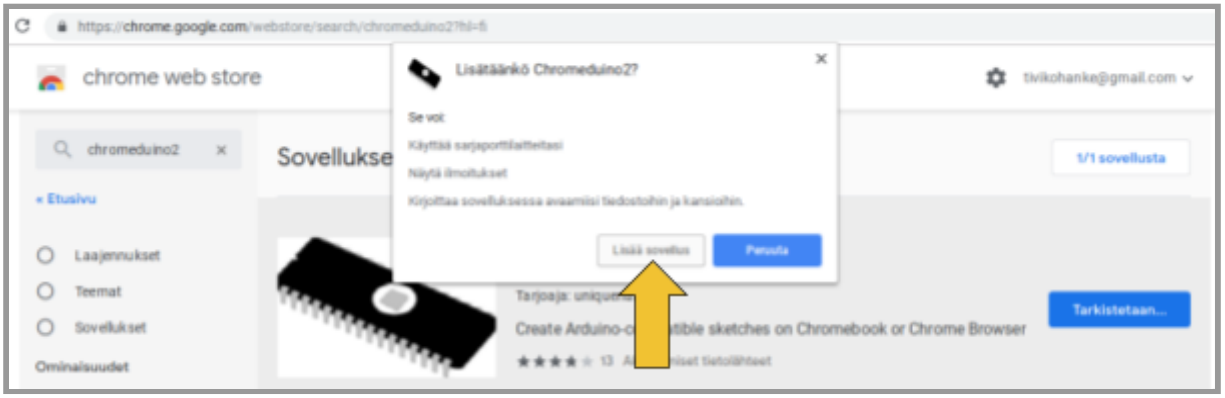
- **Varmista**:
  - Kääntää ohjelmakoodin ja ilmoittaa mahdollisista syntaksivirheistä.
- **Lähetä**:
  - Kääntää ohjelmakoodin (ilmoittaa myös virheistä) ja lähettää sen Arduino mikro-ohjaimeen. Lähetyksen jälkeen koodin suoritus käynnistyy mikro-ohjaimessa automaattisesti. Samanaikaisesti mikro-ohjaimessa voi olla vain yksi ohjelma. Uusi lähetys poistaa vanhan ohjelman mikro-ohjaimesta.



Chromebook-ohjelmointiympäristö

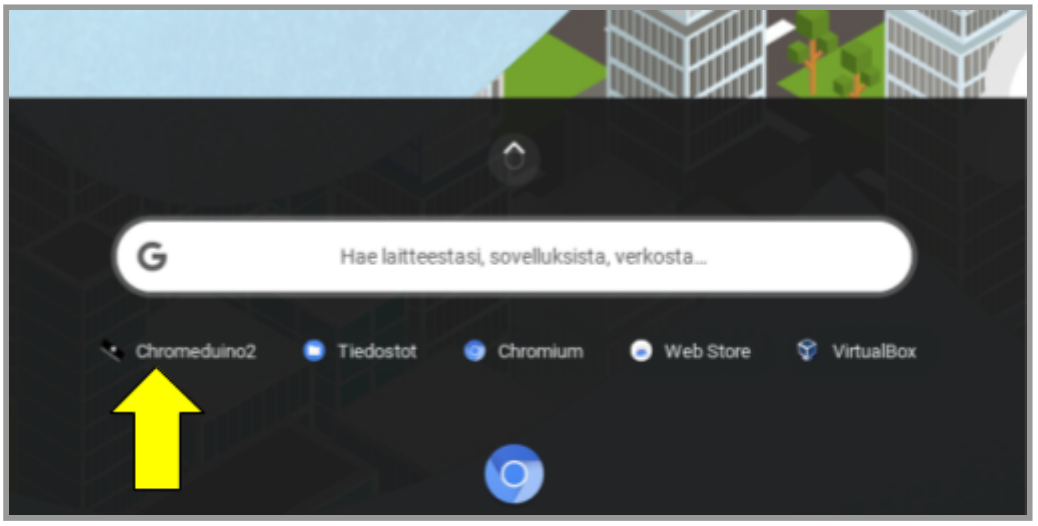
Luonnollisesti Arduinon ohjelmointiympäristöjä löytyy myös Chromebookeille. Useimmat niistä tosin ovat maksullisia, mutta käytämme ilmaista Chromeduino 2 -sovellusta. Arduinon perusohjelmointi onnistuu tällä ihan hyvin. Ainoastaan sellaiset projektit, joissa käytetään ulkoista koodikirjastoa, tulee ohjelmoida Windows-ympäristössä, koska Chromeduino 2 ei mahdollista ulkoisten kirjastojen käyttöä.

Lataa ja asenna **Chromeduino 2** sovellus alla olevasta linkistä:
https://chrome.google.com/webstore/search/chromeduino2?hl=fi

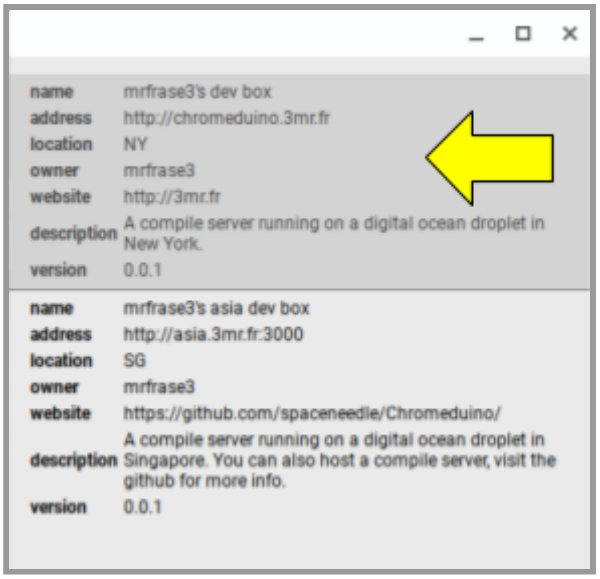


Ohjelmointiympäristön käyttö

Aina ennen kuin käynnistät ohjelmointiympäristön, niin liitä Arduino Uno USB-kaapelilla kiinni tietokoneeseen. Jos Arduino ei ole liitettynä ennen käynnistystä, niin ohjelmointiympäristö ei löydä sitä. Liittämisen jälkeen käynnistä ohjelmointiympäristö.

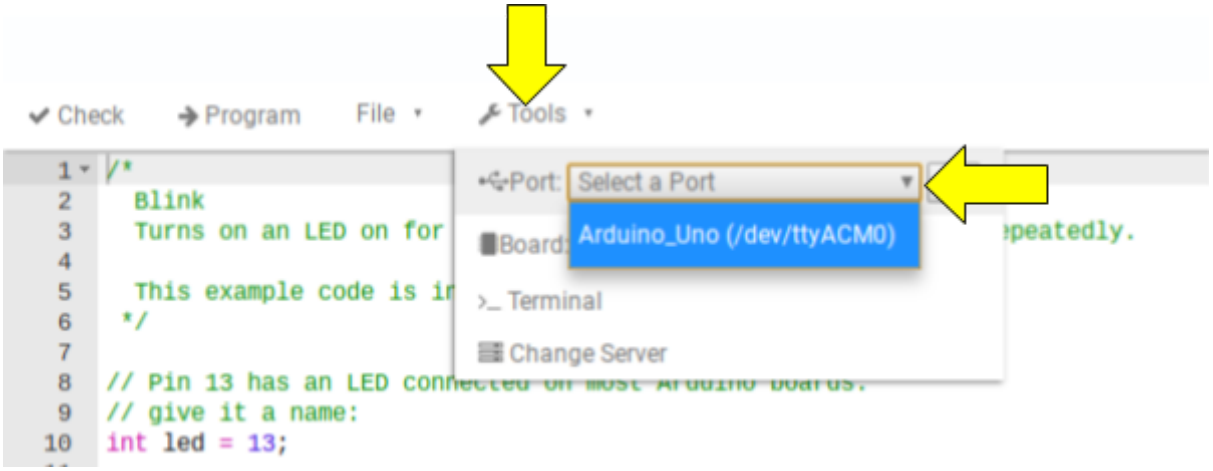


Valitse sovelluksen käyttämä serveri, klikkaa jompaa kumpaa osiosta.



Portin valinta

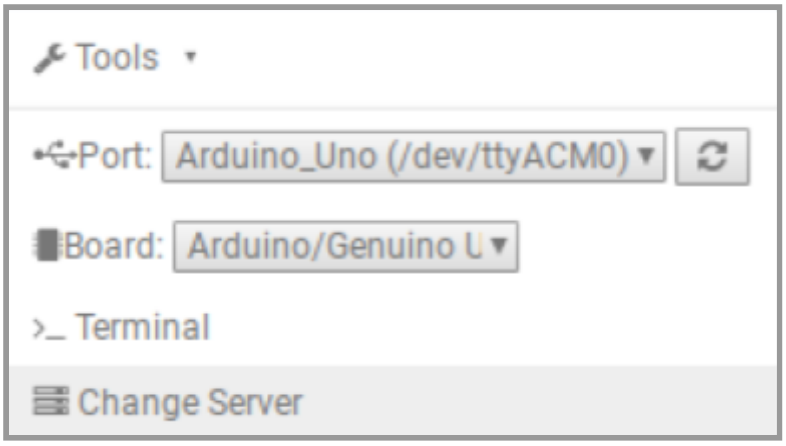
Klikkaa valikkoa **Tools** ja valitse kohtaan **Port** se portti, johon Arduino on liitetty.



Kortin valinta

Klikkaa uudelleen valikkoa **Tools** ja valitse kohdasta **Boards** valinta **Arduino/Genuino Uno**.

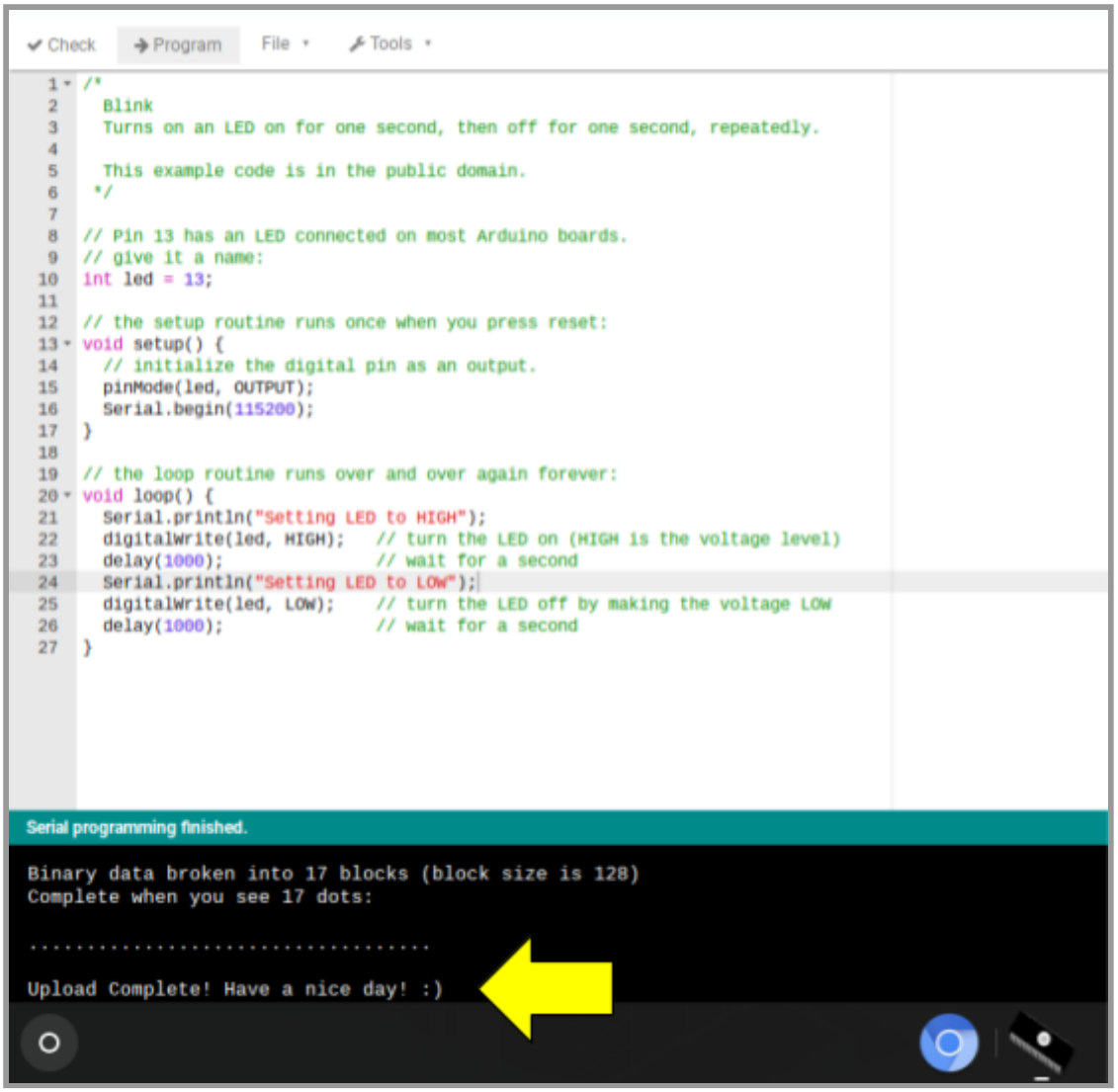
Valmiit valinnat:



Koodin tarkistus ja lataus Arduinoon

Valikon toiminnolla **Check** voit kääntää ja tarkistaa kirjoittamasi koodin syntaksin. Toiminto **Program** kääntää ja lähettää ohjelman Arduinoon. Halutessasi voit käyttää pelkästään **Program**-toimintoa, joka sekä kääntää koodin että lähettää sen Arduinoon.

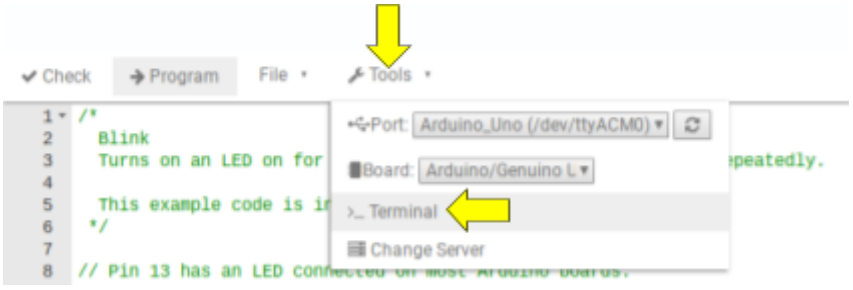
Alla oleva kuva esittää näkymää ohjelman lähetyksen jälkeen.



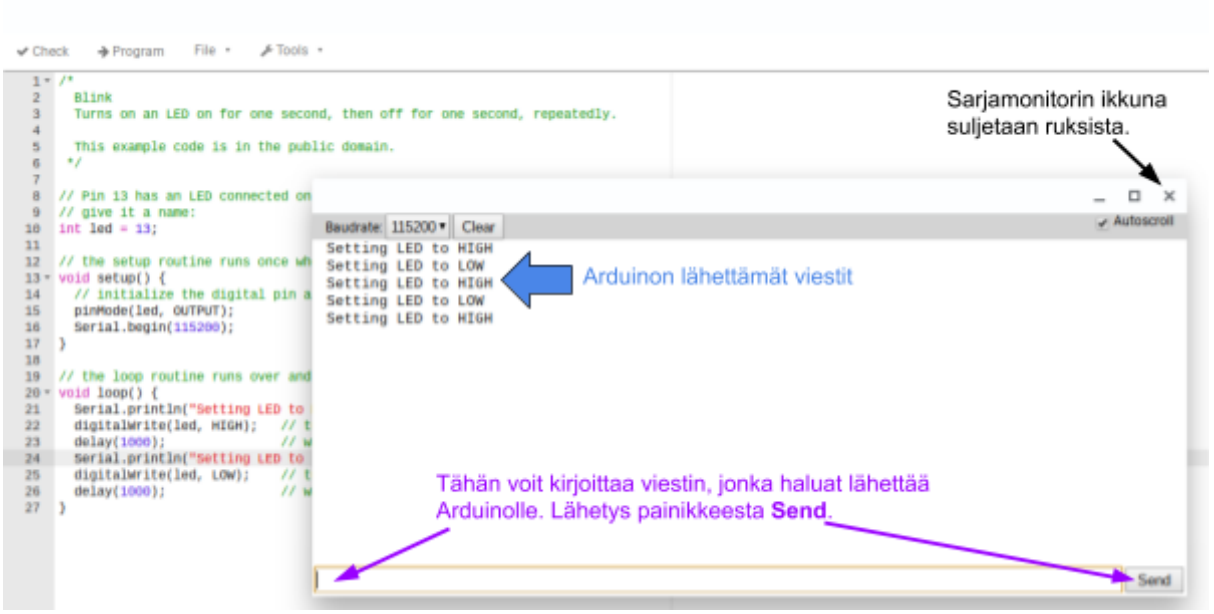
Sarjamonitori

Sarjamonitorin kautta voit vastaanottaa Arduinon lähettämiä viestejä ja myös itse lähettää Arduinolle viestejä (esim. ohjata Arduinoon liitettyä komponenttia).

Klikkaa valikkoa **Tools** ja valitse valinta **Terminal**.



Sarjamonitori aukeaa omaan ikkunaan.



File-valikko

File-valikosta löytyvillä toiminnoilla tallennetaan kooditiedosto Drivee, avataan tiedosto Drivesta tai tehdään tyhjä runko uudelle ohjelmalle (New).

Ohjelmakoodi

Ohjelmointiympäristössä ohjelma muodostetaan kirjoittamalla syntaksin mukaista koodia. Uudessa tyhjässä Arduino-ohjelmassa on valmiina kaksi funktiota:

- **setup**
  - Setup-funktioon sijoitetaan kerran käynnistyksen yhteydessä suoritettavat komennot. Funktiota käytetään tarvittavien alustustoimien suorittamiseen.
- **loop**
  - Loop-funktioon sijoitetaan pääohjelma. Tähän funktioon sijoitettuja komentoja toistetaan ikuisesti, niin kauan kuin mikro-ohjaimessa on virta päällä.

Tyhjän ohjelman runko:

```
void setup() {
  // put your setup code here, to run once:

}

void loop() {
  // put your main code here, to run repeatedly:

}
```

Arduino-ohjelman rakenne:

1. Komennolla #include tuodaan ohjelmaan käytettäväksi ulkoisten kirjastojen funktiot. Ohjelmointikielessä itsessään ei ole kovin suurta määrä valmiita funktioita, joten ulkoisia kirjastoja otetaan näin käyttöön aina tarvittaessa. Kirjastot tuodaan käyttöön koodin alussa.

2. Komennolla #define määritetään vakio, joka ei muutu ohjelman aikana. Esimerkissä annetaan vakiolle ledPin arvoksi 2. Tämä tarkoittaa, että LEDi on kytketty mikro-ohjaimen pinniin D2. Jatkossa tähän pinniin viitataan vakion nimellä ledPin, joka on helpompi muistaa. Ohjelmassa ei voi olla kahta samannimistä vakiota, eikä vakion nimi voi olla mikään kielen varattu sana.

3. Kaikki käytettävät muuttujat tulee esitellä kertomalla niiden tyyppi ja nimi. Muuttujan tyyppi määrittää, millaista sisältöä muuttujaan voidaan tallentaa. Esim. int tarkoittaa kokonaislukua ja näin ollen muuttujaan laskuri voidaan tallentaa vain kokonaislukuja. Jos muuttujat esitellään tässä kohtaa, niin ne ovat ns. globaaleja muuttujia, eli ne ovat käytettävissä ohjelman kaikissa funktioissa. Jos muuttuja esitellään funktion sisällä, niin se on käytettävissä ja näkyvissä vain kyseisessä funktiossa. Ohjelmassa ei voi olla kahta samannimistä muuttujaa, eikä muuttujan nimi voi olla mikään kielen varattu sana.

4. Samanlaisen koodin toistamista ohjelmassa tulee välttää. Jos toistettavaa koodia tarvitaan ohjelman eri kohdissa, kannattaa siitä tehdä oma funktio.

5. Kerran suoritettava **setup-funktio**.

6. Ikuisesti suoritettava **loop-funktio**.

Esimerkki ohjelman rakenteesta:

```
//1. kirjastojen tuonti
#include "wire.h"

//2. vakioiden määritys
#define ledPin 2

//3. muuttujat
//globaalien muuttujien esittely
int laskuri;
//muuttuja voidaan myös alustaa esittelyn yhteydessä
int pituus = 12345;
//samalla rivillä voidaan esitellä useita muuttujia
int a,b,c,x,y;

//4. jokin oma funktio
void omaFuntio() {

}

//5.
void setup() {
  // put your setup code here, to run once:

}

//6.
void loop() {
  // put your main code here, to run repeatedly:
  //paikallinen muuttuja, se on käytettävissä vain funktiossa loop
  int i;
}
```

Kommentit

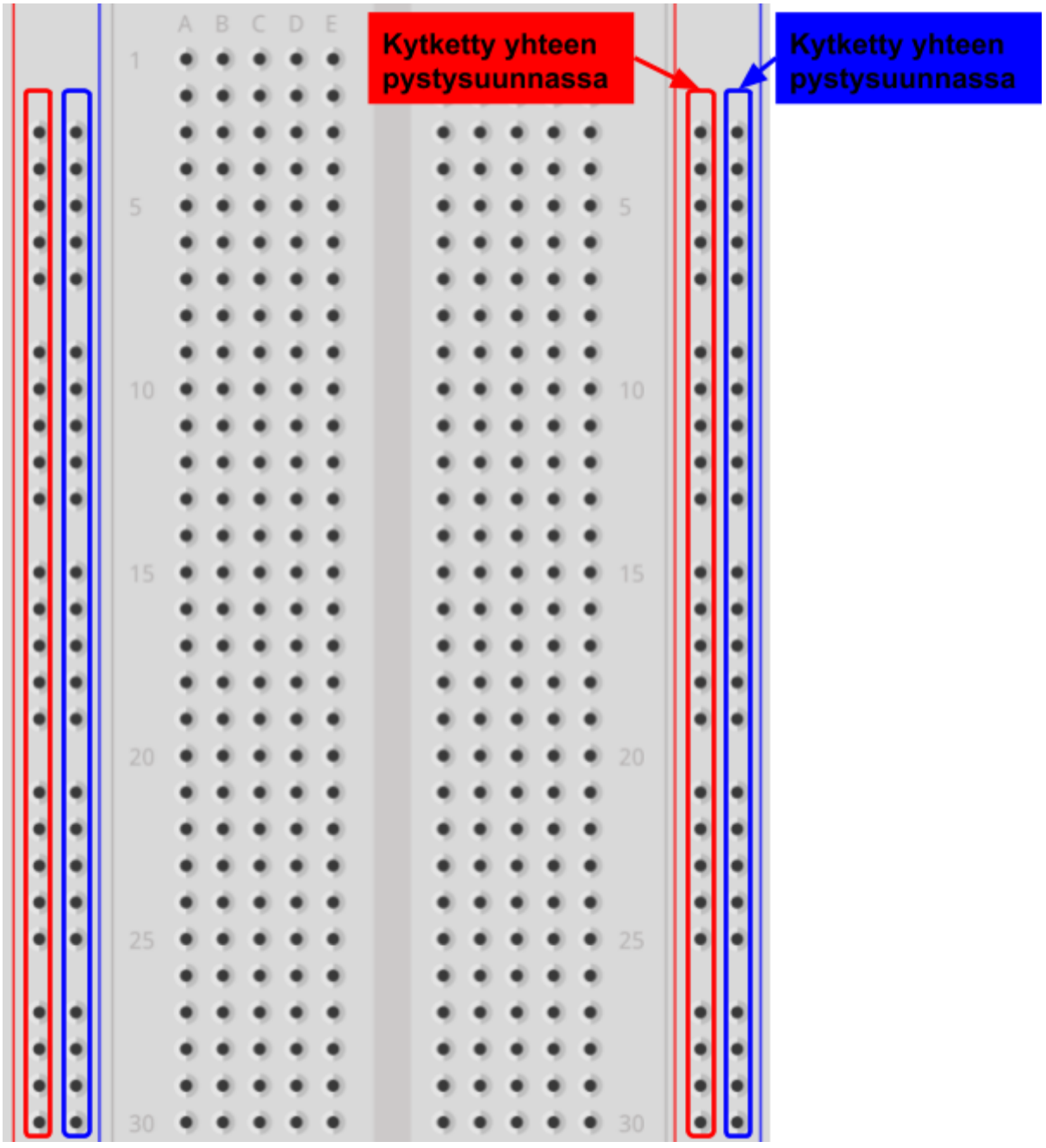
Koodia on hyvä selventää lyhyillä kommenteilla. Yksittäinen rivi kommentoidaan kahdella kenoviivalla **//**. Useampi rivi voidaan kommentoida kirjoittamalla alkuun **/*** ja loppuun ***/**. Kommentiksi merkitty osa ei vaikuta ohjelman toimintaan mitenkään ja esim. testauksessa myös jokin osa koodista voidaan siirtää kommentiksi.

```
//yksi kommenttirivi

/*
 tässä
 on
 useita
 rivejä
 kommentoituna
 */
```

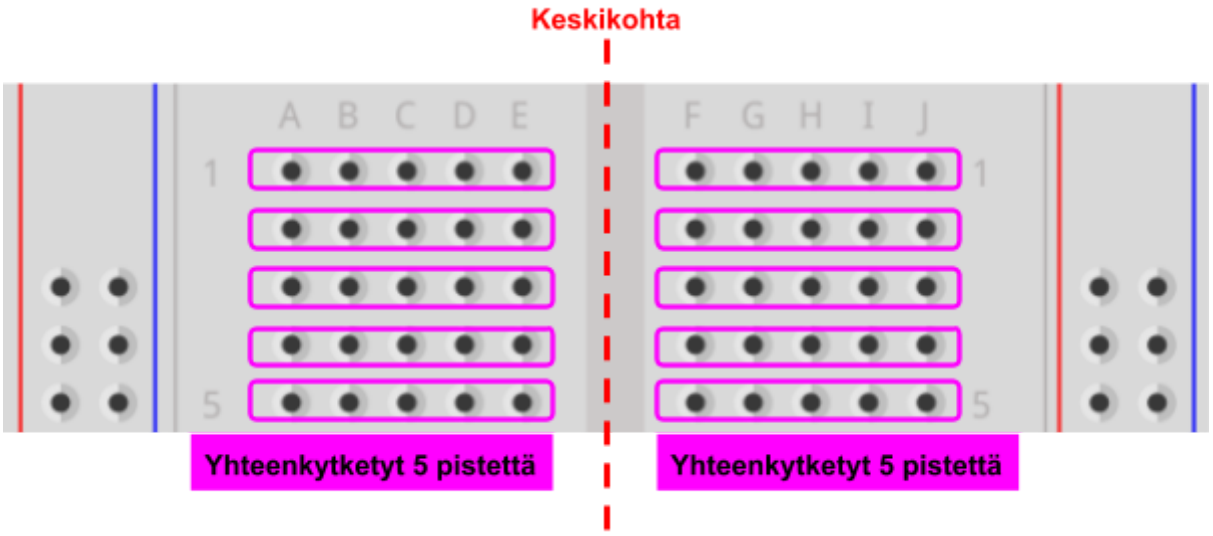
Kytkentäalustan käyttö

Kytkentäalusta on erittäin kätevä väline omien virtapiirien nopeaan ja helppoon rakentamiseen. Lisäksi komponentit saa siitä helposti irti ja niitä voidaan käyttää toisissa kytkennöissä.

Kytkentäalusta vasemmassa ja oikeassa reunassa on pystysuunnassa yhteenkytketyt pisteet. Punaista käytetään positiiviselle jännitteelle ja sinistä maadoitukselle (GND).

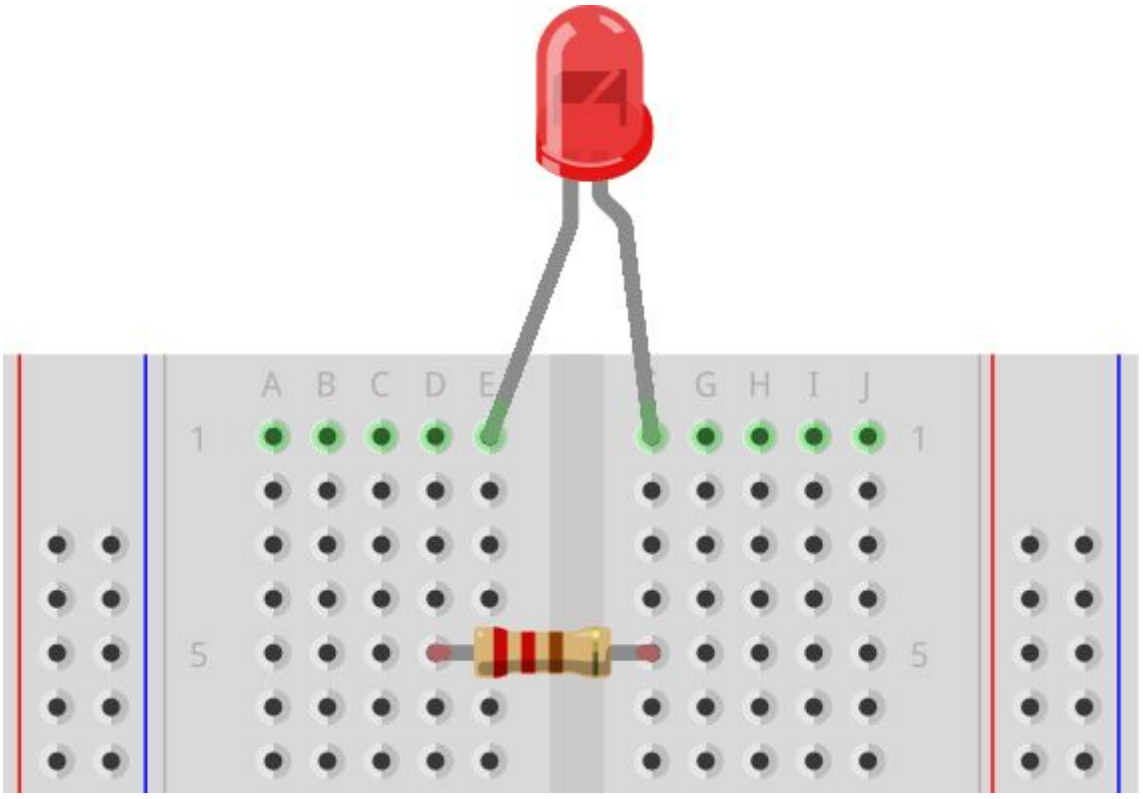


Kytkentäalustan keskikohdan kummallakin puolella on viiden pisteen muodostama rivi ja jokaisen rivin viisi pistettä on kytketty yhteen.



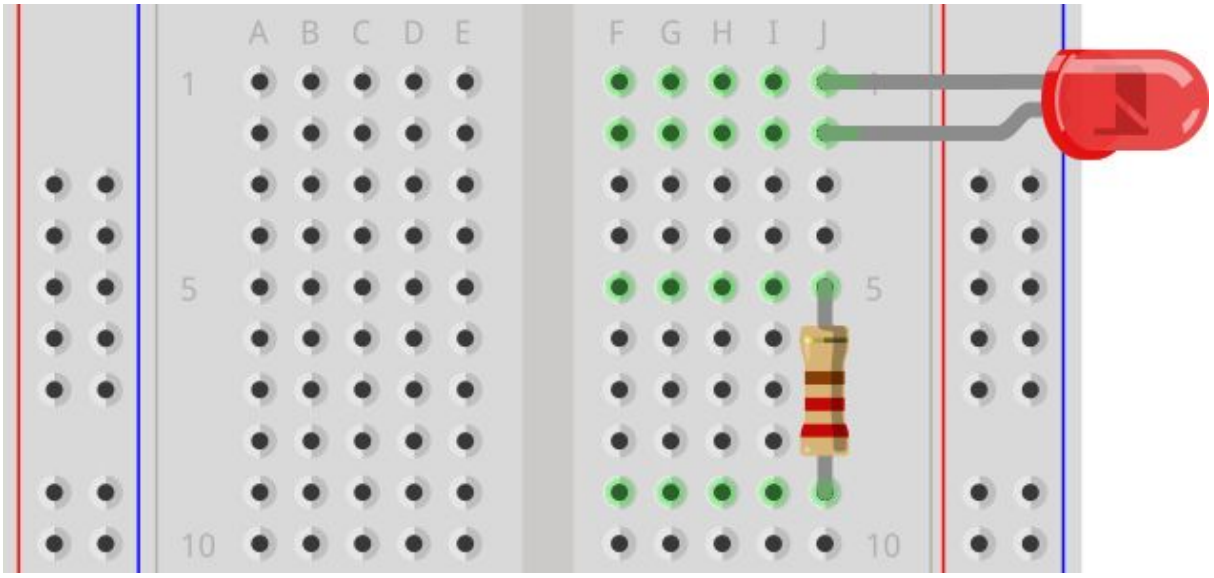
Komponentit liitetään kytkentäalustalle niin että:

- **Komponentin jalat tulevan eri puolelle keskikohtaa.**
  tai
- **Komponentin jalat tulevat eri riveille, jos ne ovat kaikki samalla puolella.**

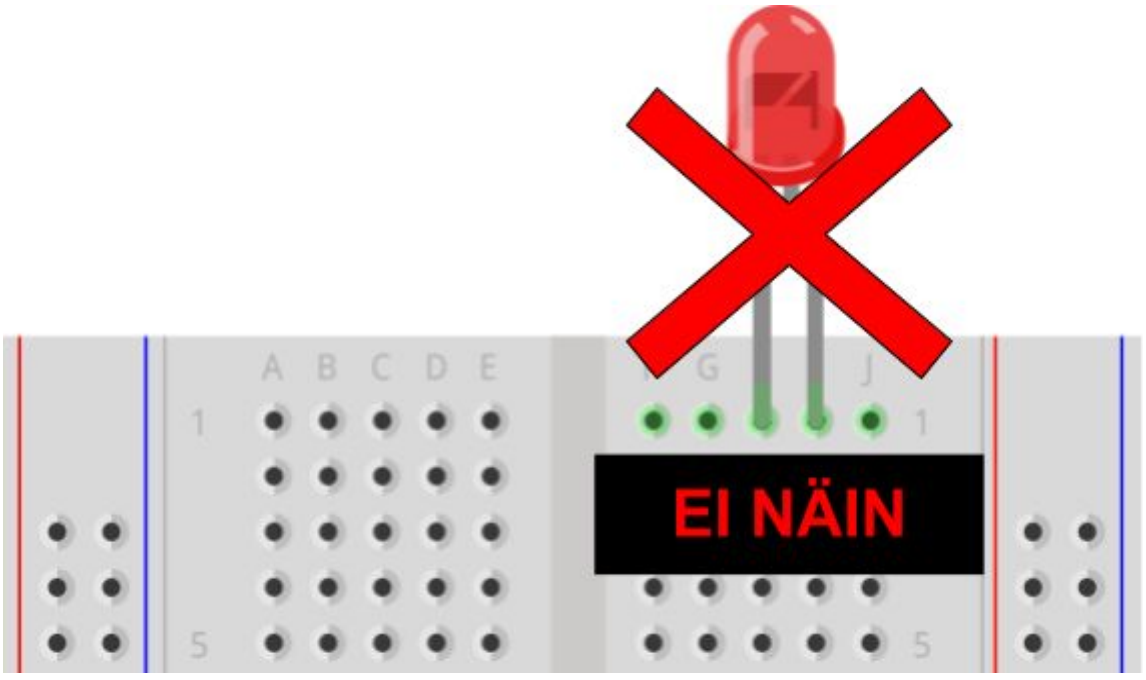


Yllä olevassa kuvassa komponettien jalat ovat eri puolella keskikohtaa.

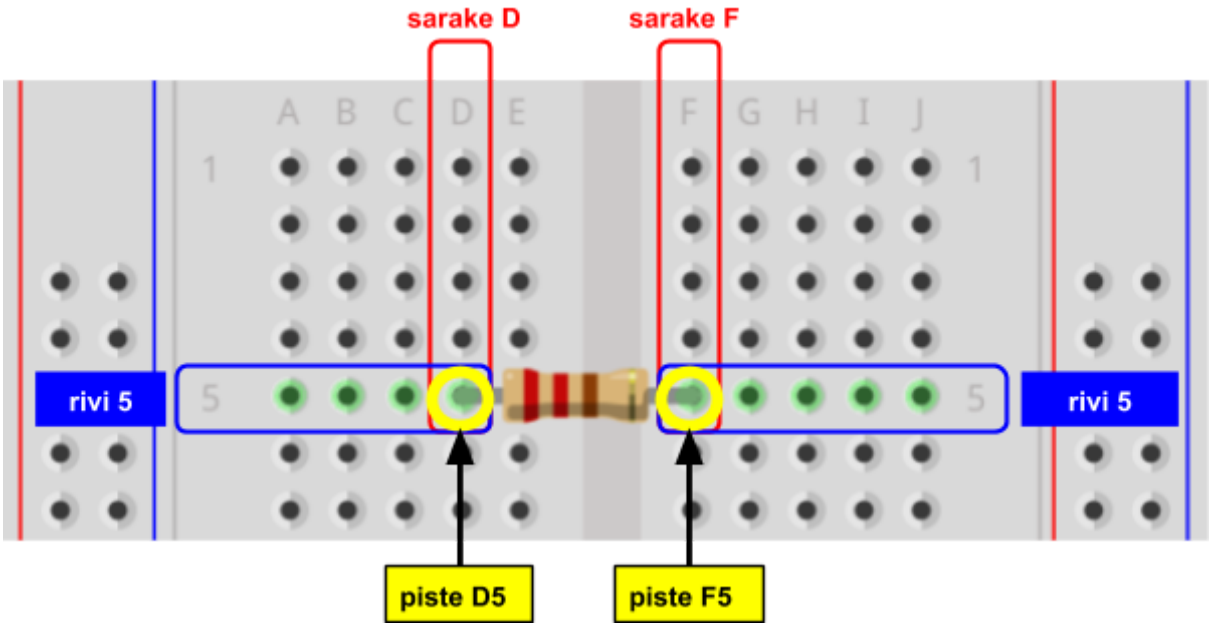
Alla olevassa kuvassa komponettien jalat on eri riveillä.



Molemmat yllä esitetyt kytkentätavat ovat oikeita. **Huomaa että komponenttia ei saa sijoittaa niin, että sen jalat ovat samalla puolella keskikohtaa ja samalla rivillä.** Rivin pisteet on kytketty yhteen ja siten komponentin jalat on kytketty täydelliseen oikosulkuun.



Työohjeissa on kerrottu, mihin kytkentäalustan pisteeseen mikin komponentin jalka tulee kytkeä. Piste kerrotaan sarakkeen kirjaimella ja rivin numerolla. Esim. ohjeessa voi lukea näin: "Kytke vastuksen ensimmäinen jalka **pisteeseen D5** ja toinen jalka **pisteeseen F5**." Alla oleva kuva selventää kytkentäohjetta.



Yleismittarin peruskäyttö

Käytä yleismittaria vain paristolla toimivien virtapiirien mittauksiin.

ÄLÄ KOSKAAN TEE MITÄÄN MITTAUKSIA PISTORASIASTA TULEVASTA VERKKOVIRRASTA.

PISTORASIASTA SAATU SÄHKÖISKU AIHEUTTAA KUOLEMAN.

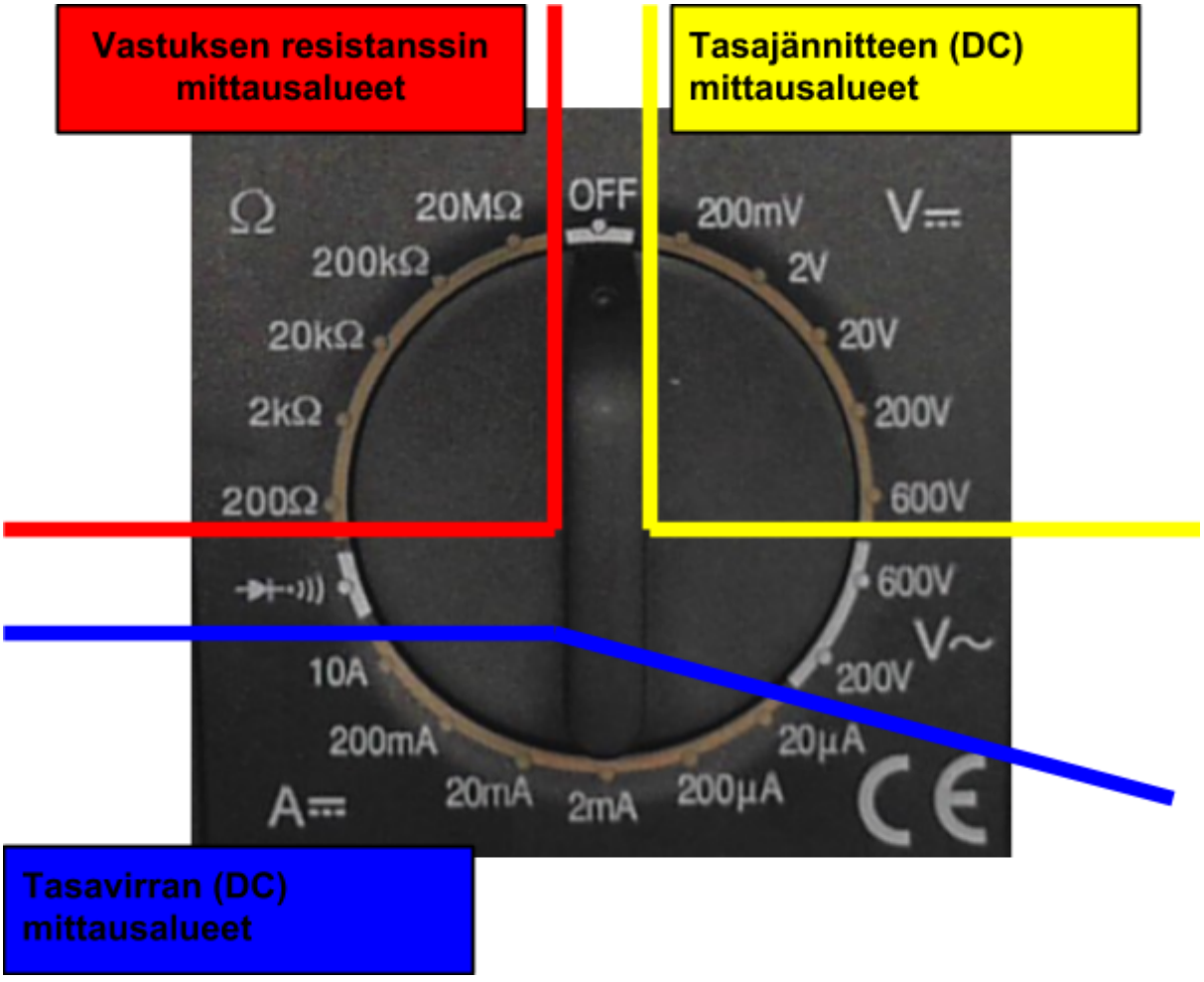
Yleismittari

Alla oleva kuva esittää yleismittaria. Mittarin yläosassa on näyttö, jossa esitetään mittaustulokset. Näytön alapuolella on kierrettävä valintakytkin. Tästä kytkimestä mittari kytketään päälle ja valitaan mitattava suure. Tällä yleismittarilla voidaan mitata mm. vastuksen resistanssia, jännitettä ja virtaa. Mittarin alaosassa on mittajohtimen liittimet.

Valintakytkin

Valintakytkintä kiertämällä mittari kytketään päälle ja valitaan mitattava suure. Lopuksi mittari myös sammutetaan tästä kytkimestä kääntämällä se **OFF-asentoon**.

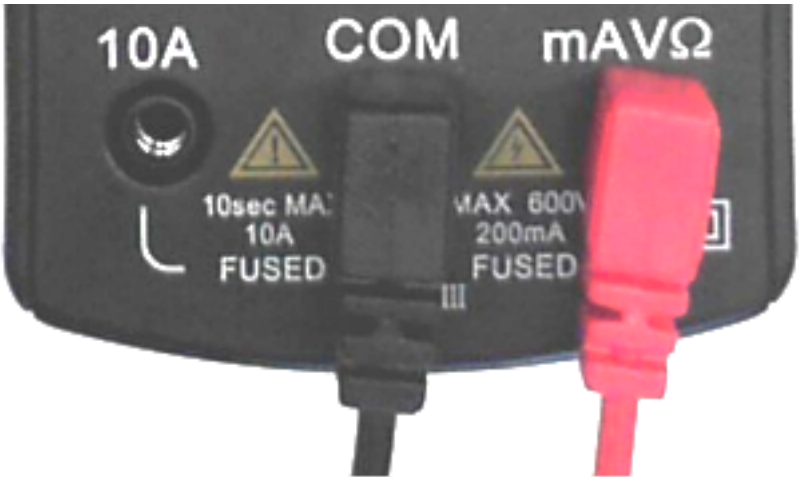
OFF-asennon vasemmalle puolelle on kuvaan merkitty punaisella vastuksen resistanssin mittausalueet. **OFF-asennon** oikealle puolelle on keltaisella merkitty tasajännitteen mittausalueet. Valintakytkimen alaosassa on sinisellä merkitty tasavirran mittausalueet.



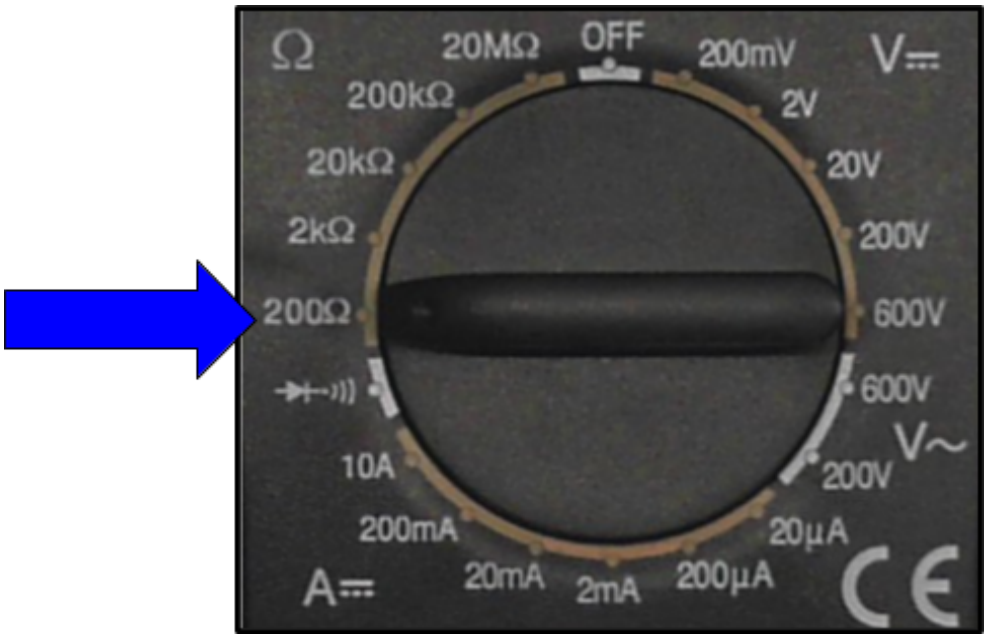
Vastuksen resistanssin mittaus

Vastuksen resistanssin mittaus on yksi yleisimpiä yleismittarilla tehtäviä mittauksia. Esimerkiksi työohjeissa on ohjeena käyttää kytkennässä tietyn resistanssin omaavaa vastusta. Yleismittarilla voidaan helposti etsiä oikea vastus.

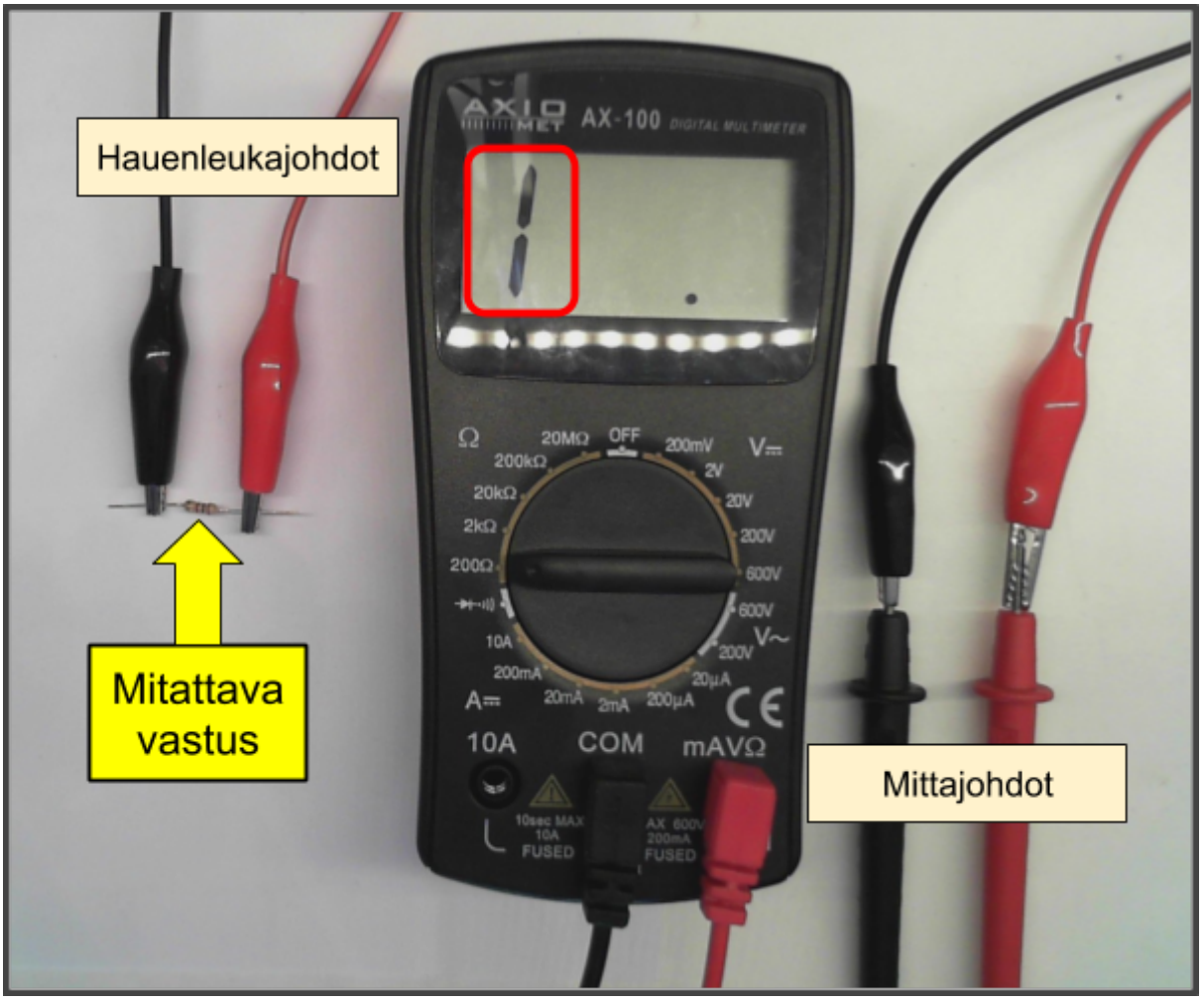
1.
Varmista, että mittarin mittajohdot on liitetty kuvan mukaisesti keskimmäiseen ja oikeanpuoleiseen liittimeen. Musta mittajohto on liittimessä **COM** ja punainen liittimessä **mAVΩ**



2.
Kierrä valintakytkin resistanssin mittausalueelle asentoon **200Ω.** Tässä asennossa mittari mittaa enintään 200 Ohmin resistanssin.

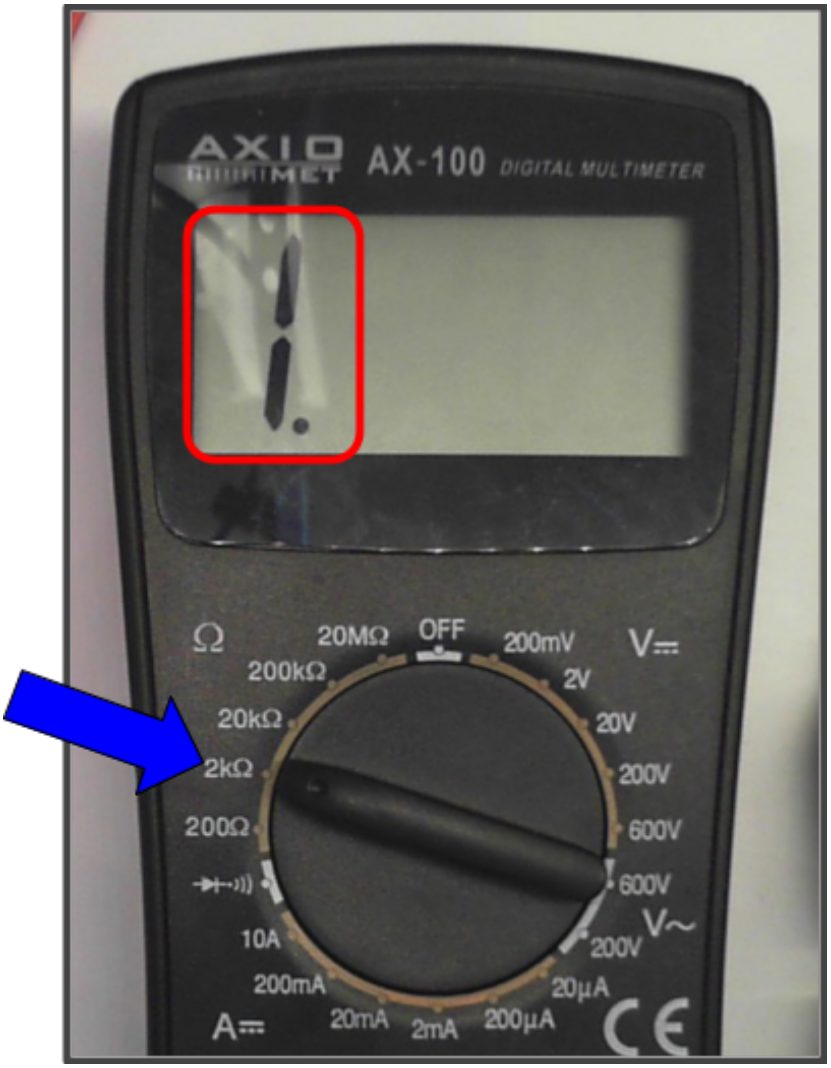


3.
Mittauksen helpottamiseksi voit liittää hauenleukajohdot mittarin mittajohtimiin. Hauenleukajohtojen toiset päät kytket sitten mitattavaan vastukseen alla olevan kuvan mukaisesti. Älä kosketa mitattavaa kohdetta molemmin käsin, sillä se voi vääristää mittaustulosta.



4.

Jos mittari näyttää lukua 1 NÄYTÖN VASEMMASSA REUNASSA (katso edellinen ja seuraava kuva), se tarkoittaa että vastuksen resistanssi on suurempi kuin mittarista valittu mittausalue. Kierrä silloin valintakytkin seuraavan suuremman mittausalueen kohdalle. Alla olevassa kuvassa valintakytkin on kierretty asentoon **2kΩ**. Tässä asennossa mitattava resistanssi voi olla väliltä **0 Ω - 2 000 Ω**. Oikea mittaustulos tulee näytön oikeaan reunaan.



5.

Mittari näyttää edelleen lukua 1 näytön vasemmassa reunassa. eli vastuksen resistanssi on suurempi kuin **2kΩ**. Käännä valintakytkin seuraavaksi suurempaan asentoon **20kΩ**. Tässä asennossa mitattava resistanssi voi olla väliltä **0 Ω - 20 000 Ω**.

20kΩ mittausalueella saimme kelvollisen lukeman. Mittarin näytön oikeassa reunassa on luku 2,65 ja koska olemme kilo-ohmin mittausalueella se tarkoittaa että vastuksen resistanssi on **2,65 kΩ**.

Tämän vastuksen resistanssiksi oli merkitty 2,7 kΩ, mutta mittaustulos oli hiukan pienempi. Tämä on aivan normaalia, sillä vastuksen todellinen resistanssi voi olla ilmoitettua arvoa hiukan pienempi tai suurempi.

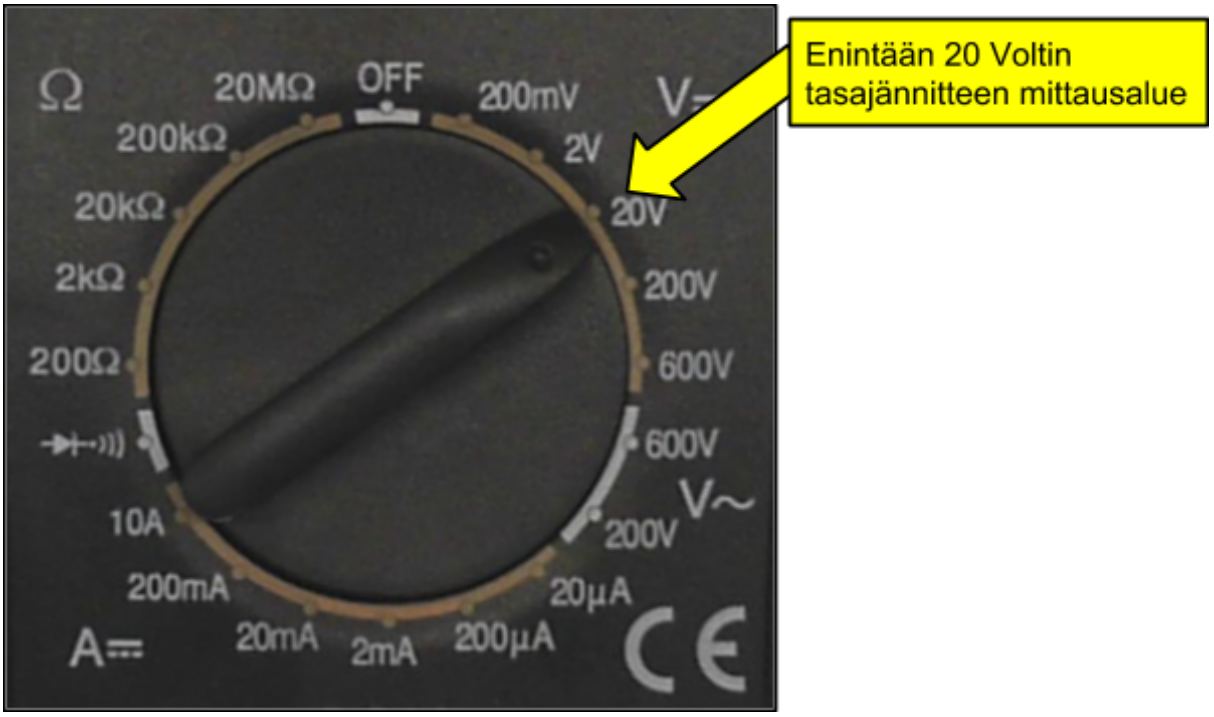
Mahdollisimman tarkan resistanssiarvon mittaamiseksi mittaukset kannattaa aloittaa pienimmältä mittausalueelta. Jos mittari näyttää lukua 1 näytön vasemmassa reunassa, niin silloin siirrytään seuraavalle suuremmalle mittausalueelle. Tätä jatketaan kunnes näytön oikeassa reunassa on kelvollinen mittaustulos.

Mittausten jälkeen muista sammuttaa yleismittari kääntämällä valintakytkin asentoon OFF.

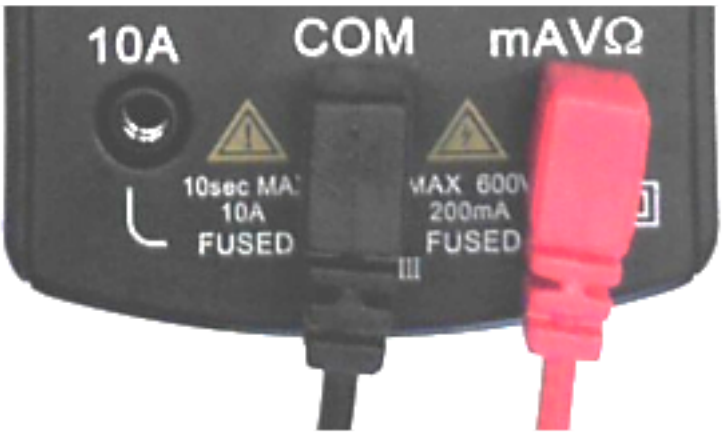
Jännitteen mittaus

Kaikissa Nuoret Värkkärit -hankkeen projekteissa käytetään alle 9 Voltin tasajännitettä. Micro:bit mikro-ohjain toimii 3 Voltin jännitteellä, Crumble 4,5 Voltin ja Arduino 5 Voltin jännitteellä. Näin ollen näiden töiden osalta voimme kaikissa jännitemittauksissa käyttää yleismittarista 20 Voltin tasajännitteen mittausaluetta. Kierrä valintakytkin tälle alueelle.

1.
Käännä valintakytkin enintään 20 Voltin tasajännitteen mittausalueelle.

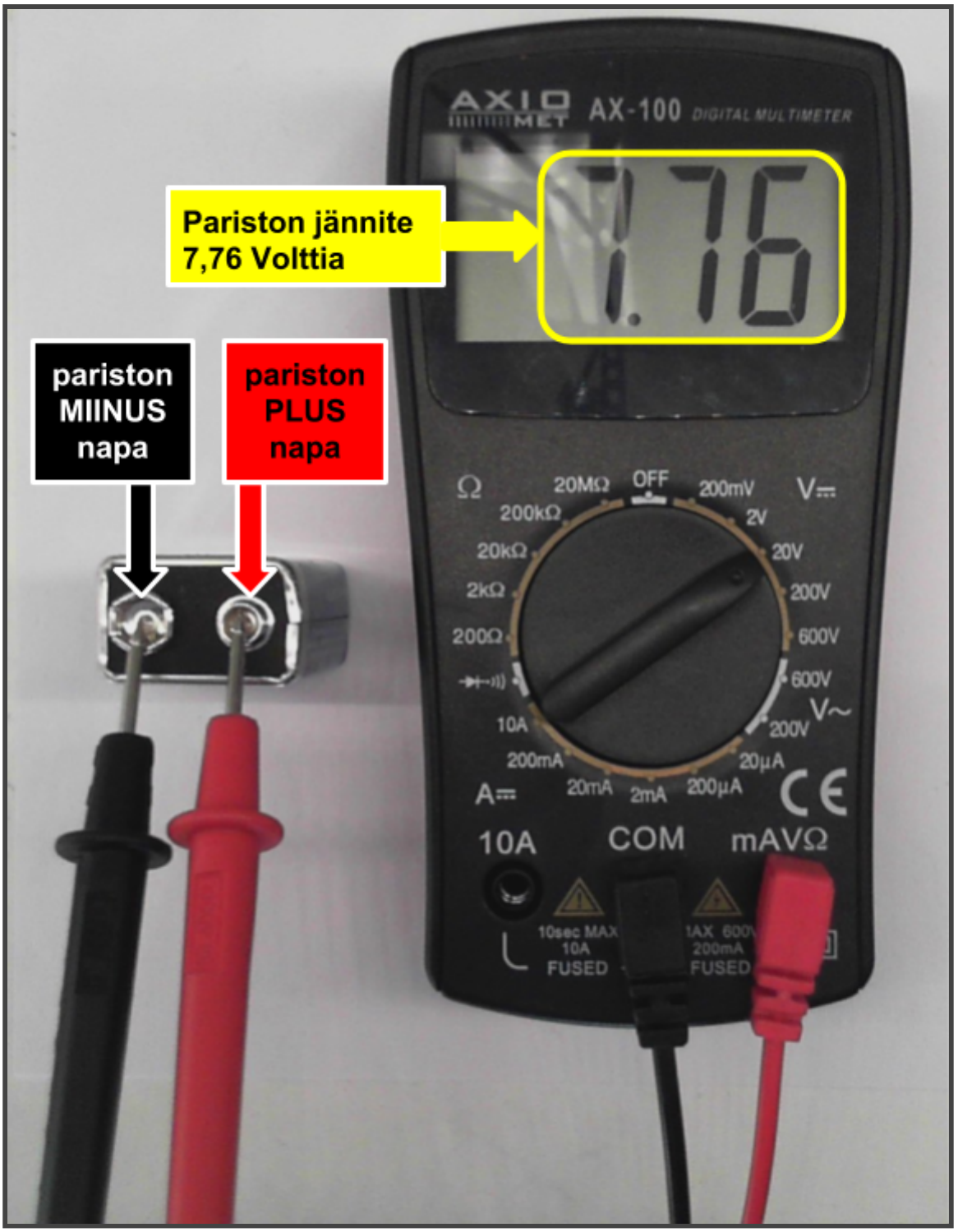


2.
Varmista, että mittarin mittajohdot on liitetty kuvan mukaisesti keskimmäiseen ja oikeanpuoleiseen liittimeen. Musta mittajohto on liittimessä **COM** ja punainen liittimessä **mAVΩ**

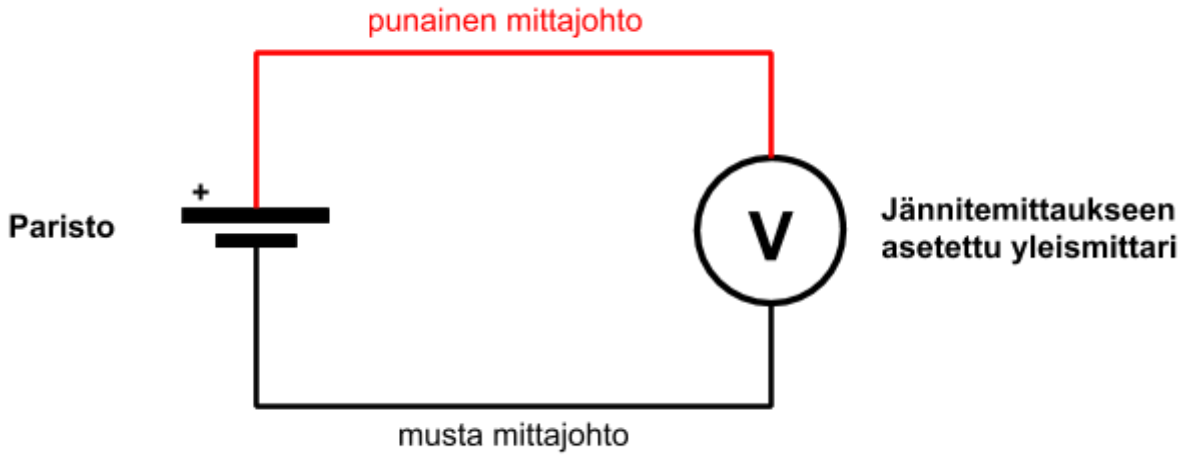


3.

Jännitettä mitattaessa mittajohdot kytketään mitattavan kohteen rinnalle. Yksinkertainen jännitteen mittaus voidaan tehdä paristosta. Kosketa samanaikaisesti **punaisella mittajohtimella pariston plus-napaa ja mustalla mittajohtimella pariston miinus-napaa.**



Alla olevassa kuvassa on esitetty jännitemittauksen kytkentäkaavio. Siitä nähdään selvemmin, miten jännitemittari on kytketty rinnan mitattavan kohteen kanssa.



Jos mittajohdot kytkee väärin päin, niin mikään ei kuitenkaan hajoa. Mittaustulos muuttuu ainoastaan negatiiviseksi.

Mittausten jälkeen muista sammuttaa yleismittari kääntämällä valintakytkin asentoon OFF.

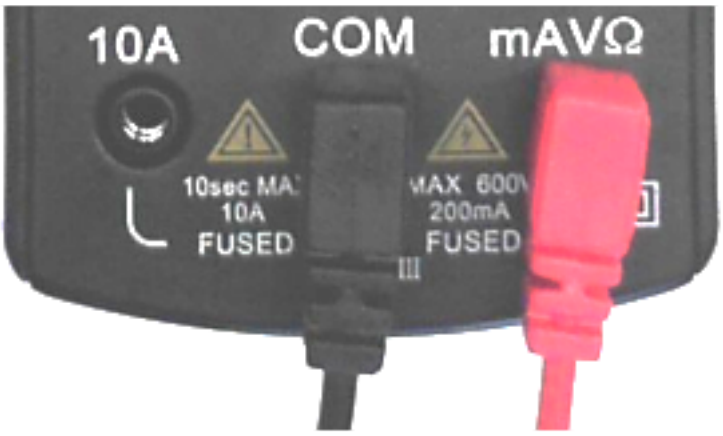
Virran mittaus

Tehdään tässä esimerkissä tasavirran mittaus alueella 200 mA eli 0,2 Ampeeria. Jos haluat mitata tätä suurempaa virtaa, niin silloin

valintakytkin tulee asettaa alueelle 10A ja punainen mittajohto kytkeä yleismittarin vasemmassa alareunassa olevaan liittimeen 10A. Näiden muutosten jälkeen tässä esimerkissä esitetyt periaatteet pätevät myös suurempien virtojen mittauksessa.

1.
Käännä valintakytkin **200 mA** tasavirran mittausalueelle.

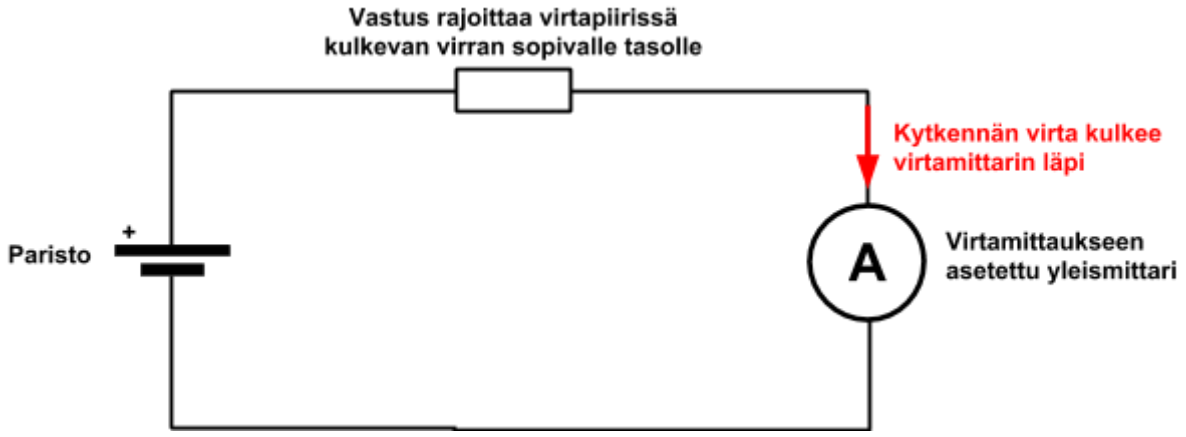
2.
Varmista, että mittarin mittajohdot on liitetty kuvan mukaisesti keskimmäiseen ja oikeanpuoleiseen liittimeen. Musta mittajohto on liittimessä **COM** ja punainen liittimessä **mAVΩ**



3.
VIRTAMITTARIA EI KOSKAAN SAA KYTKEÄ KIINNI SUORAAN PARISTON TAI MUUN VIRTALÄHTEEN NAPOIHIN. Se olisi täydellinen oikosulku ja silloin mittarin sisällä oleva sulake palaa.

Mitattavassa virtapiirissä tulee olla aina jokin virtaa rajoittava komponentti, esimerkiksi vastus tai hehkulamppu. Virtamittari voidaan kytkeä sarjaan osaksi tällaista virtapiiriä. Tällöin virtapiirin virta kulkee mittarin läpi ja tulee mitatuksi.

Alla olevassa kytkentäkaaviossa on esitetty, miten virtamittari kytketään sarjaan mitattavan virtapiirin kanssa.



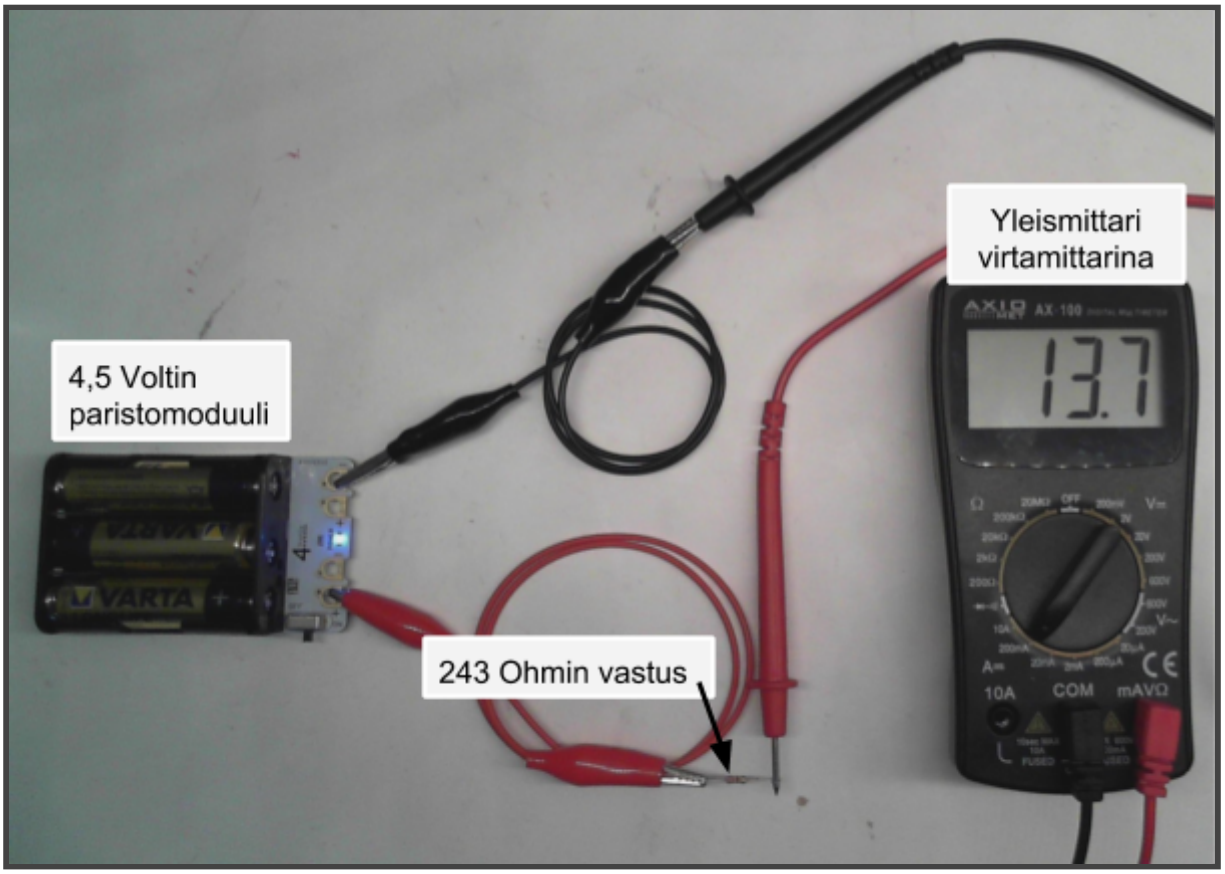
Tehdään kytkentä virran mittaamiseksi, tarvitsemme siihen:

- 4,5 Voltin paristomoduulin (tai pariston)
- 243 Ohmin vastuksen rajoittamaan virtaa

- Yleismittarin
- Punaisen ja mustan hauenleukajohdon

Kytkentä tehdään edellisellä sivulla olevan kaavion mukaisesti:

- Liitä punainen hauenleukajohto paristomoduulin plus-napaan ja kytke vastus johdon toiseen päähän.
- Liitä vastuksen toiseen jalkaan yleismittarin punainen mittajohto.
- Liitä musta hauenleukajohdin yleismittarin mustaan mittajohtoon ja kytke johdon toinen pää paristomoduulin miinus-napaan.
- Kytke paristomoduulin virta päälle.



Mitattu virta riippuu suuresti pariston jännitetasosta (kunnosta). Kuvan paristot ovat jo ihan lopussa, joten mitattu virta on vain 13,7 mA.

Mittausten jälkeen muista sammuttaa yleismittari kääntämällä valintakytkin asentoon OFF ja kytke paristomoduulista virta pois päältä.

LEDin vilkutus

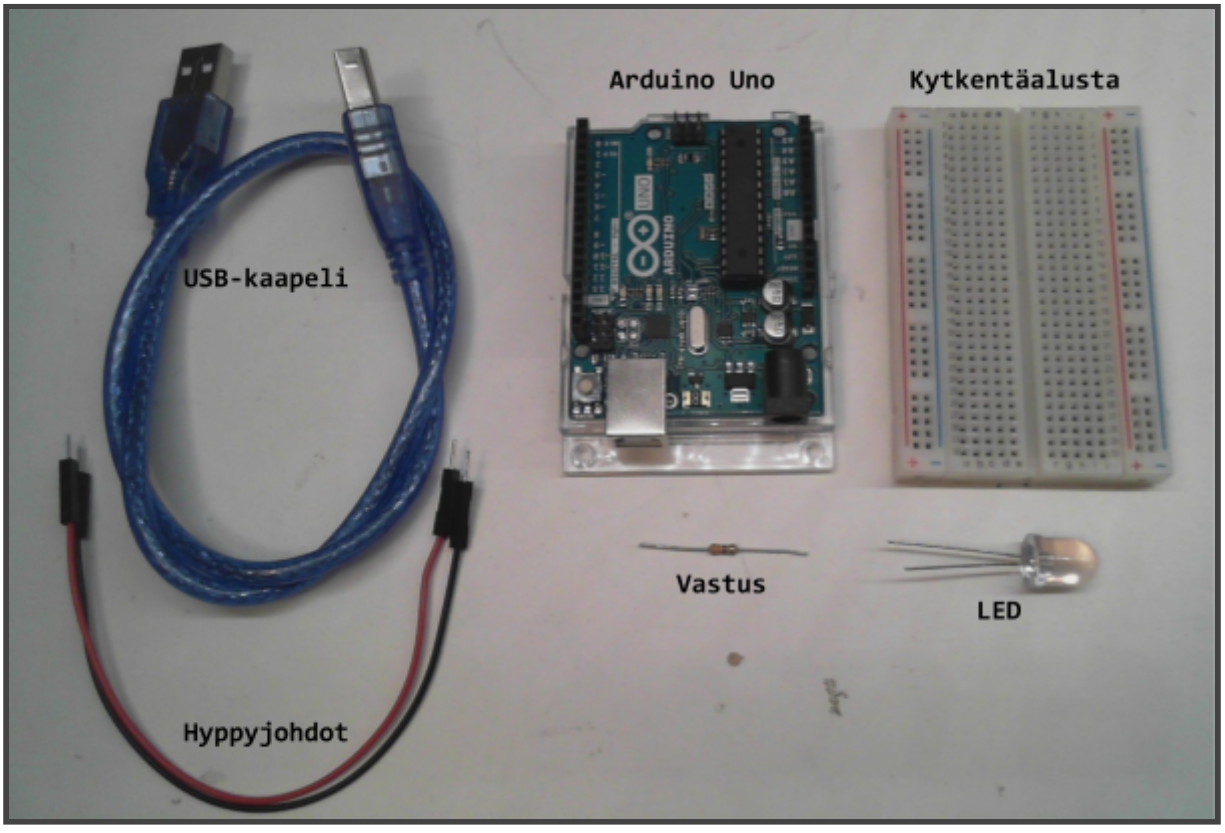
Työn kuvaus

LEDi on valoa tuottava komponentti ja se on nykyään korvannut perinteisen hehkulampun monessa kohteessa. LED on paljon kestävämpi ja energiatehokkaampi kuin hehkulamppu.

Tässä työssä rakennetaan LEDin toimintaan tarvittava kytkentä ja ohjelmoidaan Arduino kytkemään LED päälle ja pois, eli vilkuttamaan sitä. Työssä harjoitellaan LEDin ja vastuksen muodostaman virtapiirin rakentamista sekä **mikro-ohjaimen digitaalisen lähdön käyttöä** LEDin ohjaamiseen.

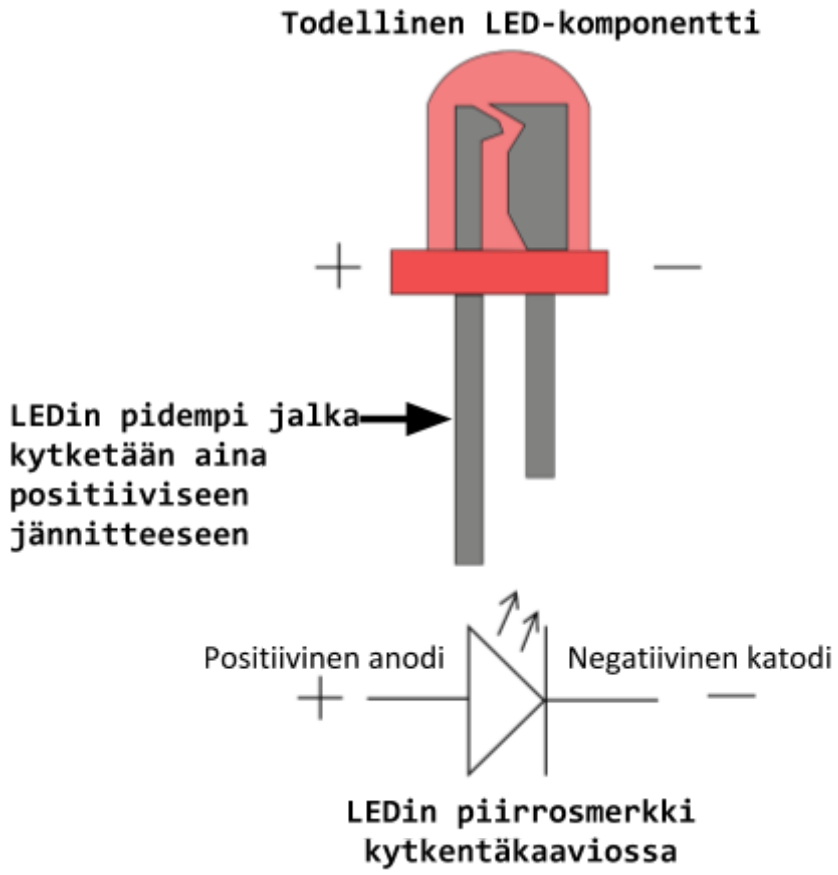
Tarvittavat komponentit

- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- 330 Ohmin vastus
- LED
- Kaksi hyppyjohdinta (uros-uros): punainen ja musta
- Kytkentäalusta



LED-komponentti

LEDissä on kaksi jalkaa, joiden kautta se kytketään osaksi virtapiiriä. Toinen jaloista on pidempi ja se tulee kytkeä aina positiiviseen jännitteeseen. Jos LEDin kytkee väärin päin, niin se ei kuitenkaan rikkoudu, mutta ei myöskään tuota valoa.



Vastus

Tässä kytkennässä vastuksen tehtävänä on rajoittaa virtapiirissä kulkeva virta sopivalle tasolle. LEDi toimii hyvin pienellä virralla, yleensä 10-20 milliampeeria on sopiva virta (0,010 - 0,020 Ampeeria).

LEDin yhteyteen on aina kytkettävä vastus sarjaan LEDin kanssa rajoittamaan virtaa. Muutoin LED ja/tai mikro-ohjain rikkoontuvat.

Todellinen vastus

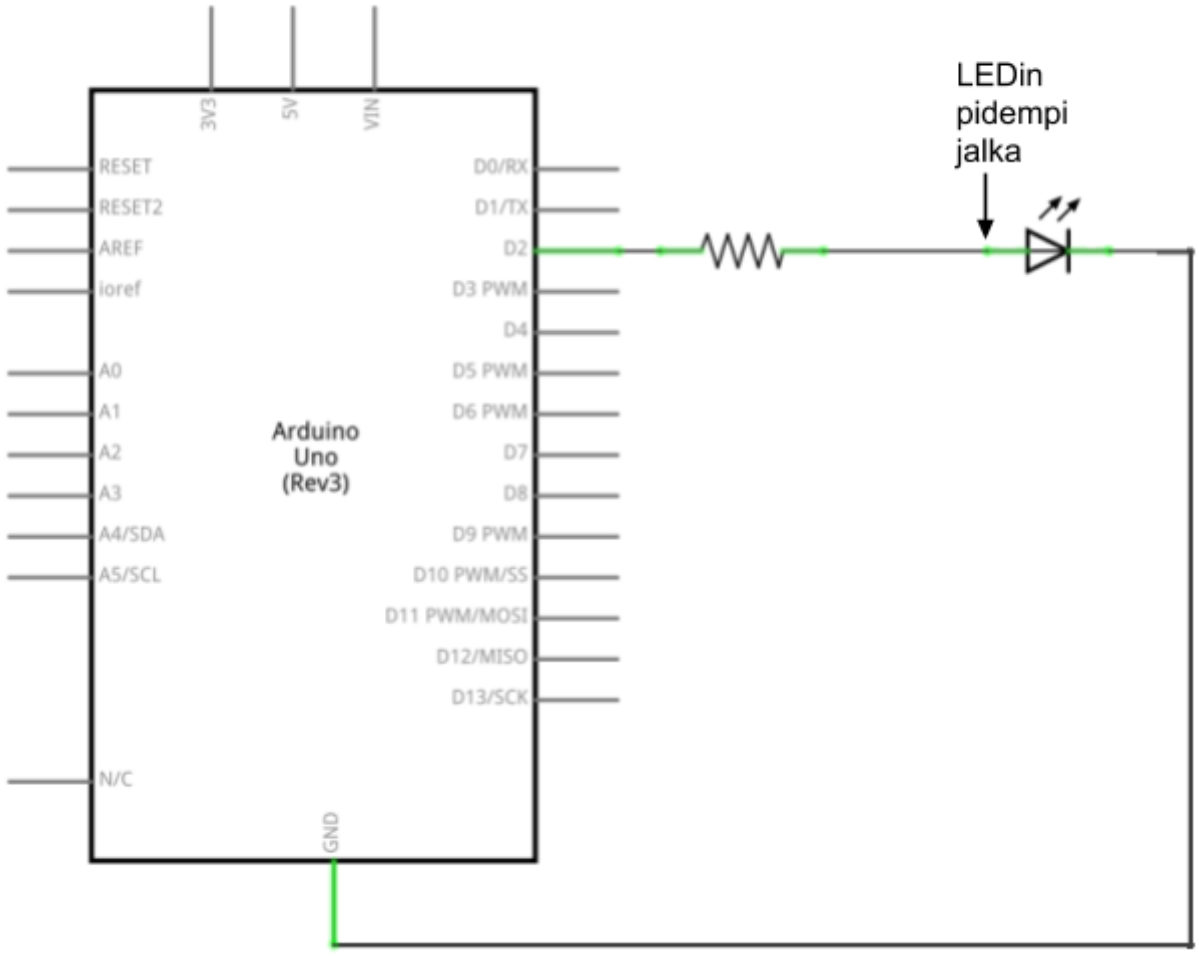
Vastuksen piirrosmerkki kytkentäkaaviossa

Vastuksen toinen yleinen piirrosmerkki
(käytössä mm. Yhdysvalloissa)

Kytkennän rakennus

Rakennetaan oheisen kytkentäkaavion mukainen kytkentä. Arduinosta käytämme pinniä D2 ja teemme siitä ohjelmallisesti digitaalisen lähdön. Pinnistä lähtee hyppyjohto kytkentäalustalle, jossa vastus ja LED on kytketty sarjaan. LEDin toisesta jalasta lähtee hyppyjohto Arduinon GND-pinniin. Näin muodostuu suljettu virtapiiri, jossa virta voi kulkea.

Kytkentäkaavio



1.

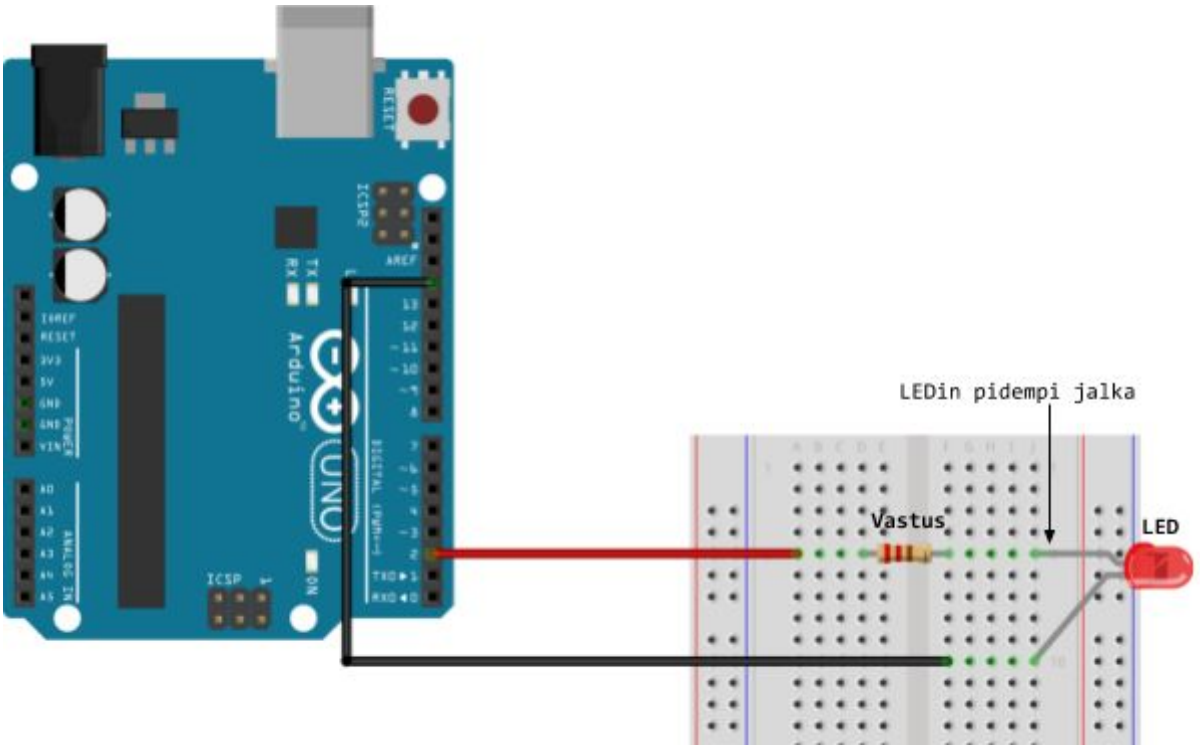
Liitä punainen hyppyjohto Arduinon pinniin **D2** (pinnissä vain numero 2) ja kytke johdon toinen pää kytkentäalustan pisteeseen **A5**.

2.
Taivuta vastuksen jalkoja ja kytke se pisteisiin **D5** ja **F5**. Vastus kulkee kytkentäalusta keskikohdan yli. Vastuksen voi kytkeä virtapiiriin miten päin vain.

3.
Taivuta hiukan LEDin jalkoja ja kytke pitempi positiivinen jalka pisteeseen **J5** ja lyhyempi jalka pisteeseen **J10**.

4.
Kytke musta hyppyjohto pisteeseen **F10** ja johdon toinen pää Arduinon pinniin **GND**. Nyt olet rakentanut kytkentäkaavion mukaisen kytkennän.

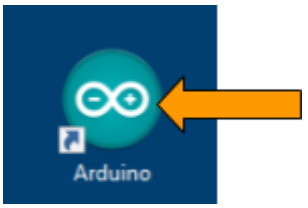
Valmis kytkentä:



Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.



Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Kaikissa Arduino-ohjelmissa on valmiina kaksi funktiota: **setup()** ja **loop()**. Kun mikro-ohjain käynnistetään (kytketään virta, resetoidaan tai avataan sarjamonitori), niin silloin suoritetaan kaikki **setup()-funktioon** sijoitetut komennot yhden kerran. Tähän funktioon sijoitetaan kaikki alustukseen tarvittavat komennot ja aloitukseen liittyvät toiminnot.

setup()-funktion suorituksen jälkeen ohjelmassa siirrytään **loop()-funktioon** ja tätä funktiota toistetaan koko ajan, niin kauan kuin mikro-ohjaimessa on virta päällä. Varsinainen ohjelma sijoitetaan tähän funktioon, joka toimii ikuisesti toistettavana toistorakenteena.

Ennen näitä kahta funktiota esitellään mm. **globaalit muuttujat** ja **vakiot.**
Nimensä mukaisesti muuttujan arvo voi muuttua ohjelman aikana, mutta vakion arvo pysyy samana koko ajan.

Vakio

Määritetään ohjelman alussa vakio nimeltä **ledPin** (vakion nimi voi olla mikä tahansa). Sen arvo kertoo, mihin pinniin meidän virtapiirimme LEDi on kytketty. Vakio määritellään avainsanalla #define <vakion nimi> <vakion arvo>. Rivin loppuun ei tule puolipistettä. Meidän ulkoinen virtapiiri on kytketty Arduinon pinniin **D2**, joten kirjoitamme koodiin:

```
#define ledPin 2
```

Jatkossa viittaamme **pinniin D2** vakion nimellä **ledPin**. Laajemmissa ohjelmissa pinnien nimeäminen niiden käyttötarkoituksen mukaan selventää ohjelmaa ja tekee siitä helpommin ylläpidettävän.

setup-funktio
Arduinon digitaalinen pinni voi toimia joko digitaalisena lähtönä tai tulona. Jompana kumpana. Digitaalisella lähdöllä voidaan ohjata ulkoista virtapiiriä ja digitaalisella tulolla voidaan lukea signaali ulkoisesta virtapiiristä (esim. painikkeen painallus).

Pinnin toimintatila asetetaan komennolla pinMode(<pin>, <mode>).
Käytämme määrittämäämme vakiota **ledPin** viittaamaan pinniin **D2**.

```
void setup() {
  pinMode(ledPin, OUTPUT);
}
```

loop-funktio
Kirjoitamme tähän funktioon koodin, joka kytkee 5 Voltin jännitteen pinniin **D2** ja pitää sen kytkettynä yhden sekunnin ajan (LEDi loistaa). Tämän jälkeen kytkemme jännitteen pois pinnistä ja odotamme yhden sekunnin ajana (LEDi on sammuksissa).

Digitaalista lähtöä ohjataan komennolla digitalWrite(<pin>, <value>). Digitaalisella signaalilla voi olla vain kaksi mahdollista arvoa, **1 ja 0**. Ykköstä vastaa jännitetaso +5 Volttia ja nollaa 0 Voltin jännite, eli ei jännitettä ollenkaan. Käytämme jälleen vakiota ledPin viittaamaan pinniin D2.

Viive ohjelmoidaan komennolla delay(<kesto ms>).

```
void loop() {
  digitalWrite(ledPin, HIGH);
  delay(1000);

  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Valmis koodi

```
//vakion määritys, jatkossa viittaamme pinniin D2 nimellä ledPin
#define ledPin 2

void setup() {
  //asetetaan pinni toimimaan digitaalisen lähtönä
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //kytketään 5 Voltin jännite pinniin ja odotetaan 1 sekunti
  //LED on päällä
  digitalWrite(ledPin, HIGH);
  delay(1000);

  //poistetaan jännite pinnistä (0 Volttia) ja odotetaan 1 sekunti
  //LED on pois päältä.
  digitalWrite(ledPin, LOW);
  delay(1000);
}
```

Kaksi kenoviivaa **//** on kommenttirivin merkki koodissa. Kommenttirivit eivät vaikuta mitenkään ohjelman toimintaan, vaan ainoastaan selventävät koodia ohjelmoijalle. Kommentteja on aina hyvä käyttää ohjelmassa.

Ohjelman kääntäminen ja lähetys Arduinoon

Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

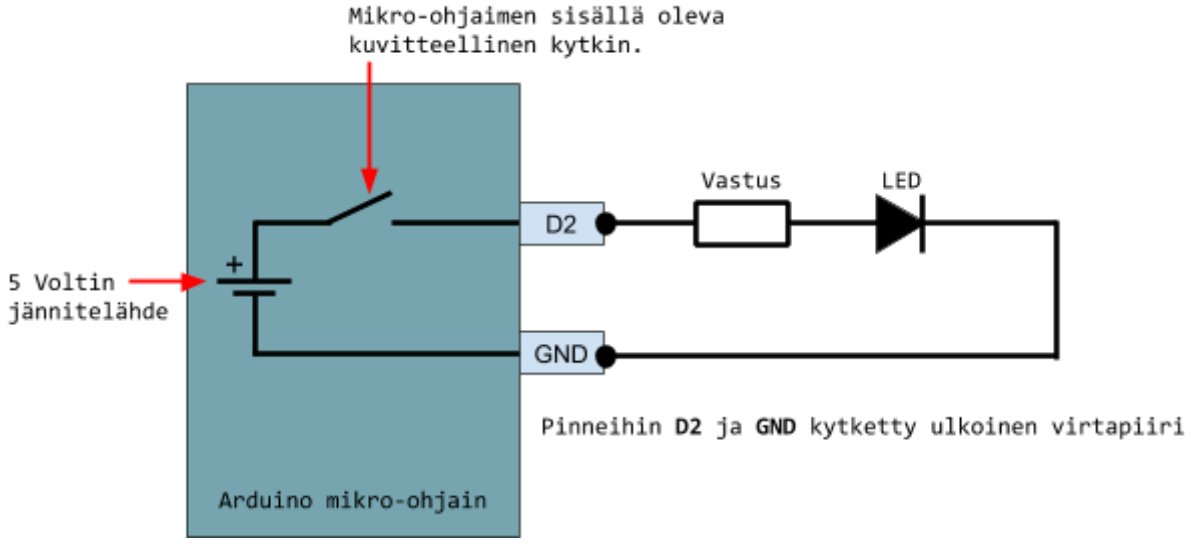
Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ohjelma konekieliseksi ja lähetä se Arduinoon klikkaamalla kuvaketta **"Lähetä"**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua. Jos ohjelman koodissa oli virhe, niin prosessi keskeytyy virheilmoitukseen. Tarkista koodi, tee tarvittavat korjaukset ja lähetä uudelleen.

Jos kytkentä on oikein tehty, niin LEDi alkaa vilkkua. Se on sekunnin päällä ja sekunnin sammuksissa. Tämä toiminta jatkuu niin kauan kuin Arduinossa on virta kytkettynä.

Koodin ja kytkennän selitys

Voimme ajatella, että digitaalinen lähtö toimii kuin kuvitteellinen kytkin mikro-ohjaimen sisällä. Se voidaan avata ja sulkea komennolla. Kun kytkin suljetaan kytkeytyy lähtöpinniin 5 Voltin jännite ja kun kytkin avataan, niin pinnissä ei ole jännitettä ollenkaan.



Pinni D2 on ohjelmallisesti määritetty toimimaan **digitaalisena lähtönä**:

```
pinMode(ledPin, OUTPUT);
```

Kuvitteellinen kytkin suljetaan näin:

```
digitalWrite(ledPin, HIGH);
```

Ja kytkin avataan näin:

```
digitalWrite(ledPin, LOW);
```

Kun kuvitteellinen kytkin suljetaan muodostuu **suljettu virtapiiri**, jossa **virta voi kulkea**. Virta matkaa 5 Voltin jännitelähteen plus-navasta suljetun kytkimen kautta **pinniin D2**. Siitä matka jatkuu ulkoiseen virtapiiriin vastuksen ja LEDin läpi. Lopuksi virta palaa mikro-ohjaimen **GND-pinniin** ja siitä jännitelähteen miinus-napaan.

Virtapiirissä kulkeva virta tekee työtä. Tässä kytkennässä se saa LEDin tuottamaan valoa. Kun kuvitteellinen kytkin avataan, kytkennästä tulee **avoin virtapiiri**. Siinä **virta ei enää voi kulkea**, koska jännitelähteen plus-navasta sen miinus-navalle ei ole enää sähköä johtavaa reittiä. Ja kun virtapiirissä ei kulje enää virtaa, niin myös LEDi luonnollisesti sammuu.

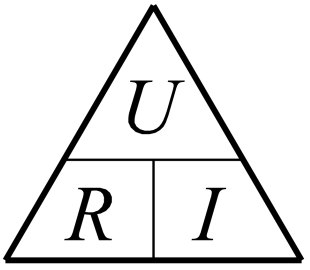
Suljetussa tasavirtapiirissä kulkeva virta voidaan laskea **Ohmin lailla**. Se on kaava, joka määrittää kolmen suureen suhteen toisiinsa. Kun kaksi tiedetään, niin kolmas voidaan laskea. Kaavan suureet ovat:

- Jännite U [V]
- Virta I [A]
- Resistanssi R [Ohm]

Resistanssi on vastuksen ominaisuus ja se kertoo, kuinka paljon vastus rajoittaa virtaa. Mitä suurempi on vastuksen resistanssi, sitä pienempi virta pääsee kulkemaan vastuksen läpi.

Ohmin laki määrittää, miten jännite, virta ja resistanssi liittyvät toisiinsa. Ohmin laki on helpoin muistaa ns. muistikolmiosta.

Peitä kolmiosta suure, jonka haluat laskea, niin näet sen laskentakaavan.



Jännite U = Resistanssi R x Virta I
Resistanssi R = Jännite U / Virta I
Virta I = Jännite U / Resistanssi R

Meidän kytkennässä jännite U = 5 Volttia ja ulkoisessa virtapiirissä olevan vastuksen resistanssi on 330 Ohmia. Laskuissa meidän tulee ottaa vielä huomioon LEDin ns. kynnysjännite, jonka suuruus löytyy komponentin datalehdestä. (lisätietoa LED-virtapiirien kytkennöistä)

Kun esim. laskemme virtapiirissä kulkevan virran, niin kynnysjännite U_D tulee vähentää jännitelähteen jännitteestä U.

$$I = (U - U_D) / R = (5\,V - 2,1\,V) / 330\,Ohm = 8,79\,mA$$

Lisätehtävät

Kokeile toteuttaa ohjelmaan seuraavat muutokset:

1. LED vilkkuu hitaammin.
2. LED vilkkuu nopeammin.
3. LED on päällä ja pois päältä eri pituisen ajan.

Liikennevalo

Työn kuvaus

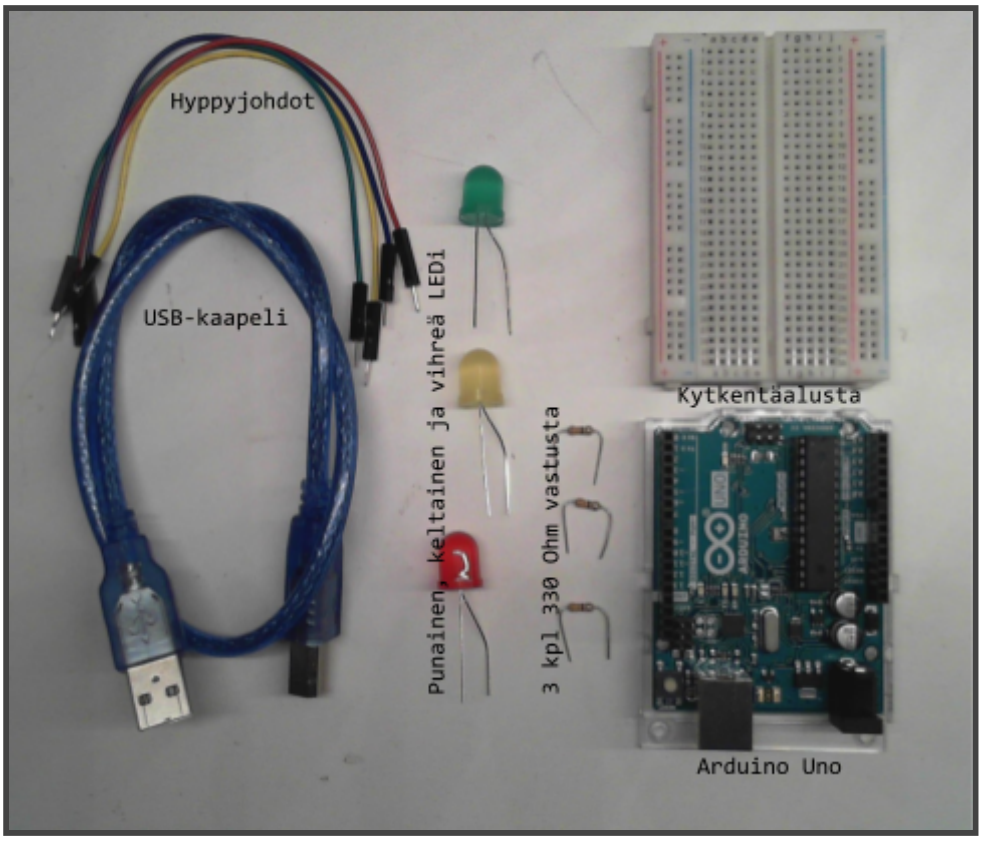
Mikro-ohjaimen digitaalisen lähdön käyttö on sinulle jo tuttua aiemmasta työstä (LEDin vilkutus). Niin ikään osaat rakentaa toimivan virtapiirin vastuksesta ja LEDistä. Nämä ovat tarvittavat esitaidot tämän työn tekemiseksi.

Sovellamme aiemmin oppimaamme ja teemme hiukan monipuolisemman ohjelman ja sillä ohjattavan kytkennän. Liikennevalo on meille kaikille tuttu laite ja tässä työssä rakennetaan sellainen minikoossa. Työohjeessa ei anneta lopullista koodia valmiina, vaan sinun tulee osata soveltaa ohjetta koodin ohjelmoimiseksi. Ohjelmointia voi oppia vain itse tekemällä, pohtimalla ja yrittämällä.

Tarvittavat komponentit

- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- 3 kpl 330 Ohmin vastusta
- 3 kpl LEDejä: yksi vihreä, yksi keltainen ja yksi punainen
- Neljä hyppyjohdinta (uros-uros): vihreä, keltainen, punainen ja sininen
- Kytkentäalusta

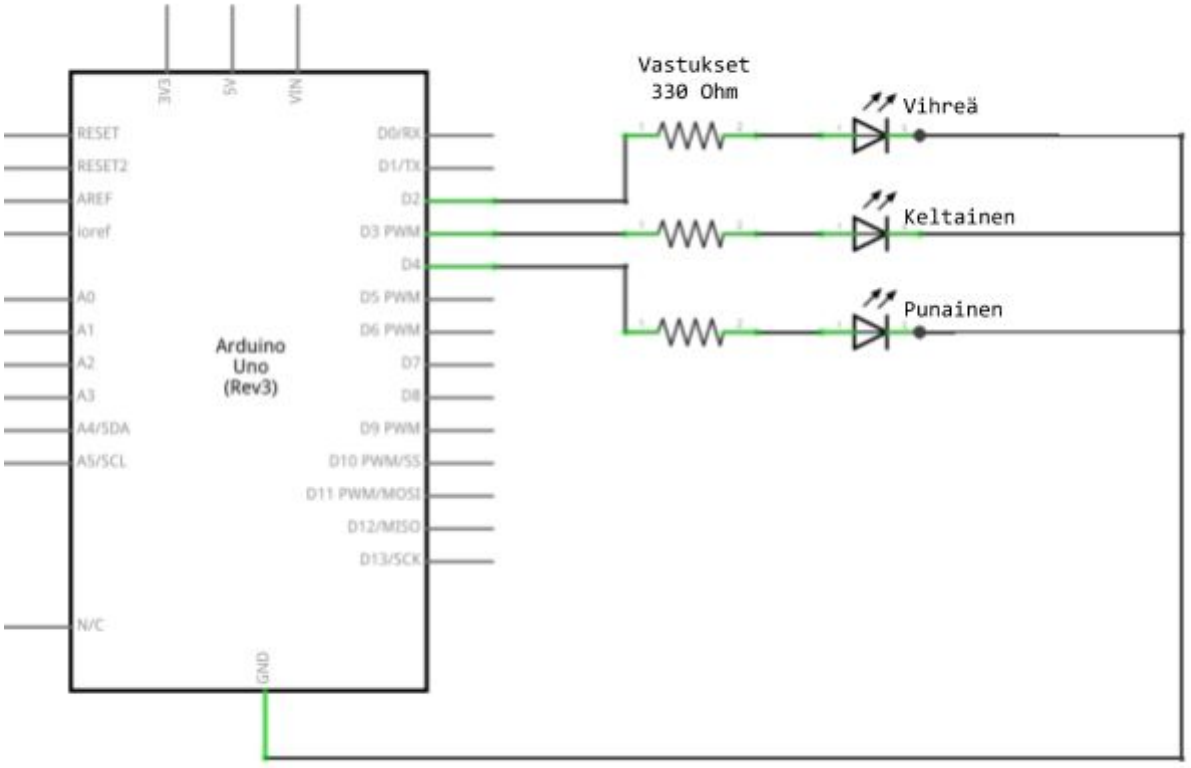
Seuraavalla sivulla on kuva tarvittavista komponenteista.


Hyppyjohdot
USB-kaapeli
Punainen, keltainen ja vihreä LEDi
3 kpl 330 Ohm vastusta
Kytkentäalusta
Arduino Uno

Kytkennän rakennus

Rakennetaan oheisen kytkentäkaavion mukainen kytkentä. Käytämme Arduinon pinnejä **D2**, **D3** ja **D4 digitaalisina lähtöinä**. Jokaisesta pinnistä lähtee hyppyjohto kytkentäalustalle, jossa vastus ja LED on kytketty sarjaan:

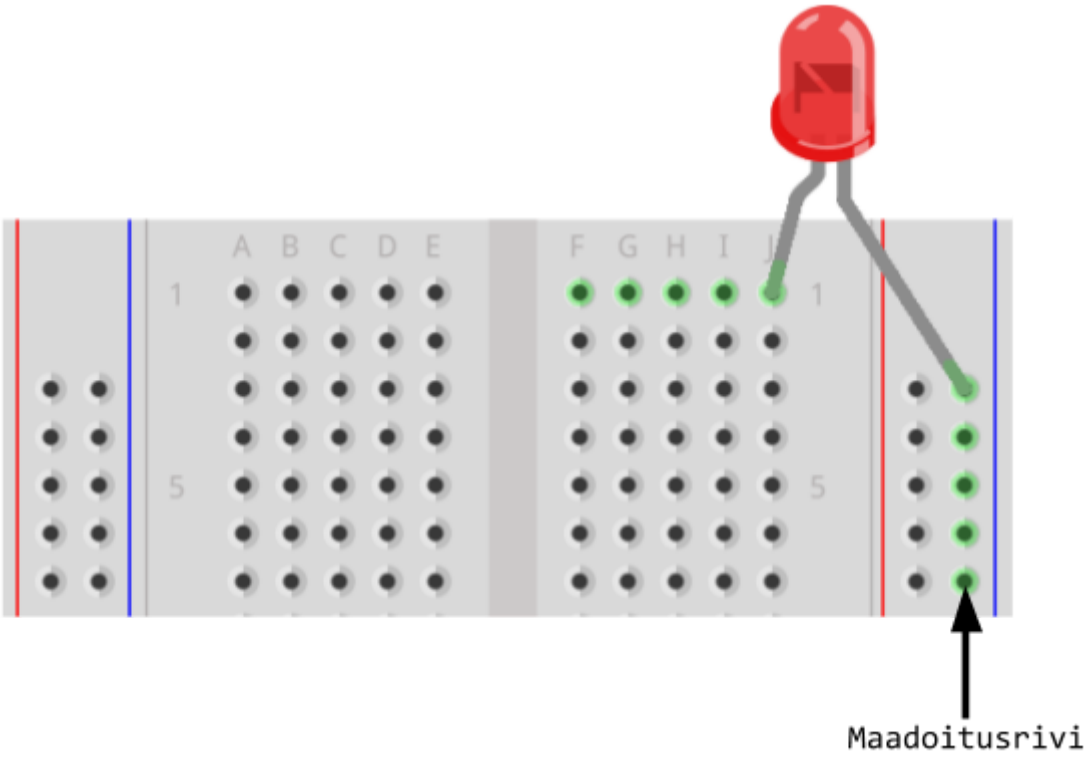
- Pinni D2 ohjaa vihreää LEDiä.
- Pinni D3 ohjaa keltaista LEDiä.
- Pinni D4 ohjaa punaista LEDiä.
- GND pinnin on yhteinen kaikille em. pinneille



Huomaa että todellisessa kytkennässä punainen LEDi on ylimpänä (niin kuin oikeassa liikennevalossakin) ja vihreä alimpana.

Voit kokeilla rakentaa kytkennän suoraan yllä olevan kytkentäkaavion mukaisesti. Vaihtoehtoisesti voit rakentaa sen myös seuraavalla sivulla olevien kytkentäohjeiden mukaisesti.

1.
Liitä punaisen LEDin pidempi jalka **pisteen J1** ja toinen jalka **maadoitusriviin**.



2.
Liitä keltaisen LEDin pitempi jalka **pisteeseen J10** ja toinen jalka **maadoitusriviin**.

3.
Liitä vihreän LEDin pitempi jalka **pisteeseen J20** ja toinen jalka **maadoitusriviin**.

4.
Kytke 330 Ohmin vastus **pisteisiin D1** ja **F1**.

5.
Kytke 330 Ohmin vastus **pisteisiin D10** ja **F10**.

6.
Kytke 330 Ohmin vastus **pisteisiin D20** ja **F20**.

7.
Liitä vihreä hyppyjohto (uros-uros) Arduinon **pinniin D2** ja johdon toinen pää **pisteeseen A20**.

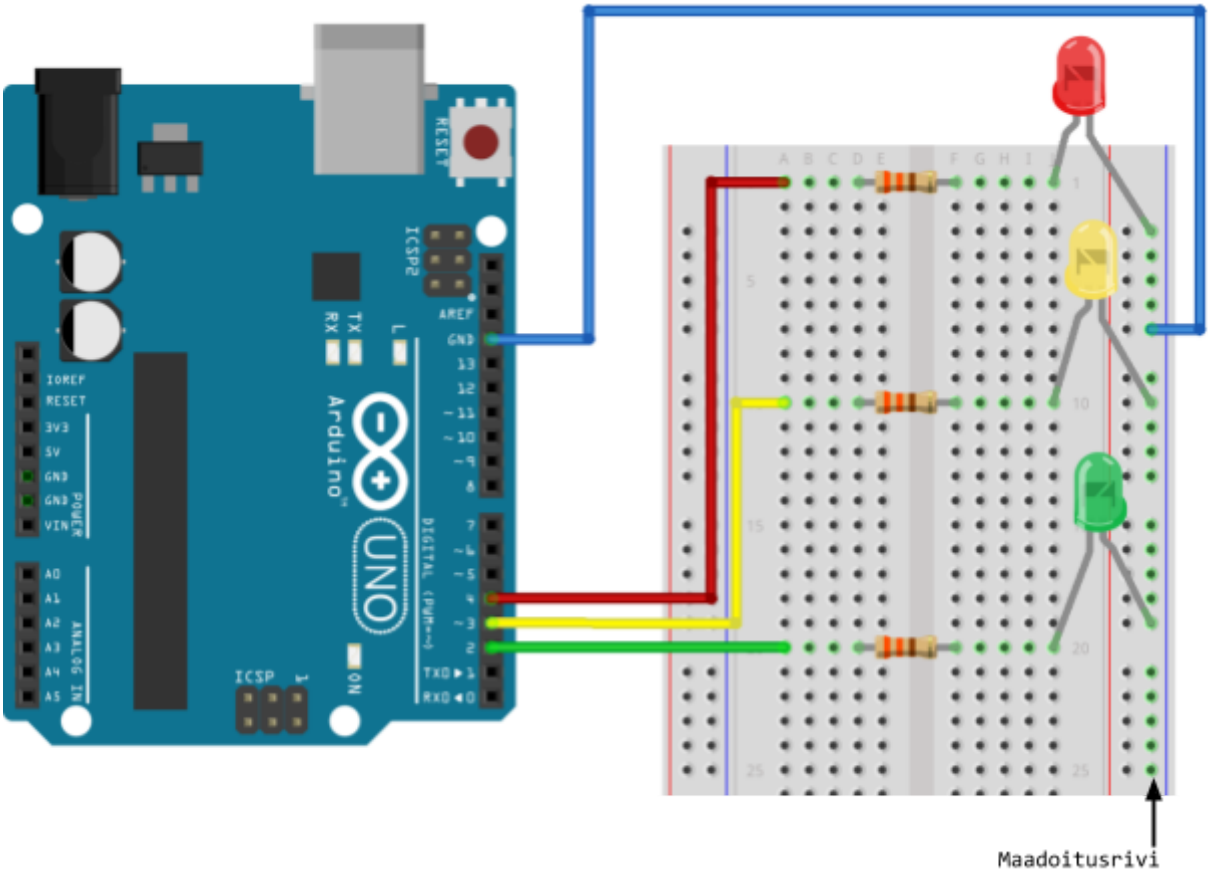
8.
Liitä keltainen hyppyjohto Arduinon **pinniin D3** ja johdon toinen pää **pisteeseen A10**.

9.
Liitä punainen hyppyjohto Arduinon **pinniin D4** ja johdon toinen pää **pisteeseen A1**.

10.
Liitä sininen hyppyjohto Arduinon **pinniin GND** ja johdon toinen pää johonkin **maadoitusrivin pisteeseen** (ei väliä mihin, ne kaikki on kytketty yhteen).

Nyt olet rakentanut kytkentäkaavion mukaisen kytkennän.

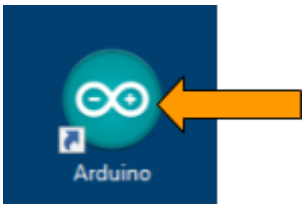
Valmis kytkentä:



Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.



Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakio

Määritetään ohjelman alussa vakiot, joilla myöhemmin viittaamme pinneihin **D2**, **D3** ja **D3**. Vakiot kannattaa nimetä niin, että jo nimestä tietää mitä vakio tarkoittaa. Esim. vakiosta rLed tietää että sillä ohjataan punaista LEDiä (red led). Totutusti vakiot määritetään koodin alussa avainsanalla #define.

```
#define gLed 2
#define yLed 3
#define rLed 4
```

setup-funktio

Määritämme pinnit **D2**, **D3** ja **D4** toimimaan digitaalisina lähtöinä ja viittaamme pinneihin niille määritettyjen vakioiden nimillä gLed, yLed ja Rled. Pinnin toimintatila asetetaan komennolla pinMode(<pin>, <mode>).

```
void setup() {
  pinMode(gLed, OUTPUT);
  pinMode(yLed, OUTPUT);
  pinMode(rLed, OUTPUT);
}
```

loop-funktio

Tehdään alkuksi koodi, jolla testataan kytkennän toimivuus. Kytketään jokainen LEDi vuorollaan päälle sekunnin ajaksi, sammutetaan ja siirrytään seuraavaan LEDiin. LEDin sytyttäminen ja sammuttaminen tehdään komennolla digitalWrite(<pin>, <value>). Viive ohjelmoidaan komennolla delay(<kesto ms>). Aloitetaan vihreästä LEDistä.

```
void loop() {
  digitalWrite(gLed, HIGH);
  delay(1000);
  digitalWrite(gLed, LOW);

  digitalWrite(yLed, HIGH);
  delay(1000);
  digitalWrite(yLed, LOW);

  digitalWrite(rLed, HIGH);
  delay(1000);
  digitalWrite(rLed, LOW);
}
```

Valmis koodi

```
//vakioiden määritykset

#define gLed 2  //vihreä LED pinnissä D2
#define yLed 3  //keltainen LED pinnissä D3
#define rLed 4  //punainen LED pinnissä D4


void setup() {
  //määritetään pinnit toimimaan digitaalisina lähtöinä
  pinMode(gLed, OUTPUT);
  pinMode(yLed, OUTPUT);
  pinMode(rLed, OUTPUT);
}

void loop() {
  //sytytetään vihreä LED yhdeksi sekunniksi
  //ja sammutetaan sen jälkeen
  digitalWrite(gLed, HIGH);
  delay(1000);
  digitalWrite(gLed, LOW);

  //sytytetään keltainen LED yhdeksi sekunniksi
  //ja sammutetaan sen jälkeen
  digitalWrite(yLed, HIGH);
  delay(1000);
  digitalWrite(yLed, LOW);

  //sytytetään punainen LED yhdeksi sekunniksi
  //ja sammutetaan sen jälkeen
  digitalWrite(rLed, HIGH);
  delay(1000);
  digitalWrite(rLed, LOW);
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

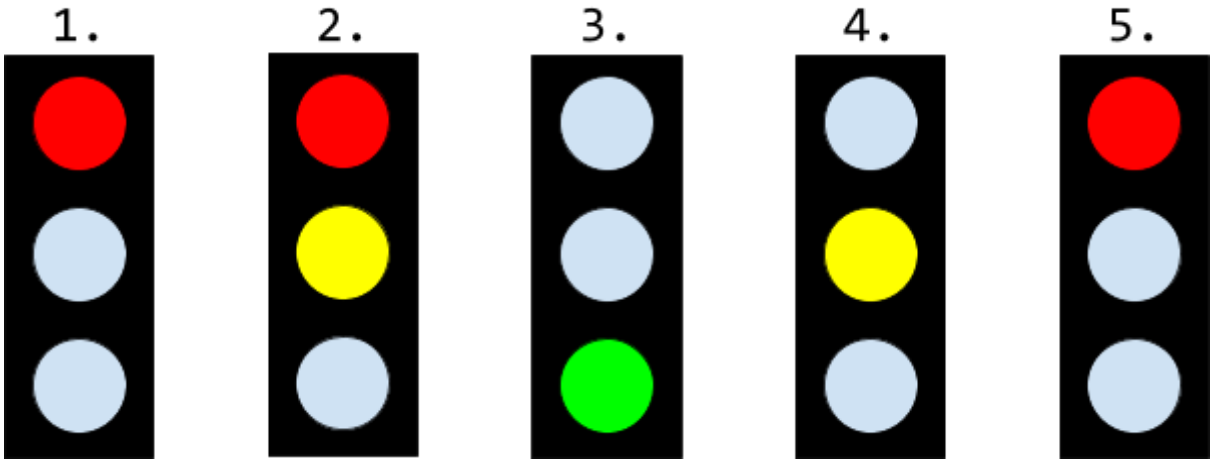
Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta “Lähetä”. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

Jos ohjelman koodissa oli virhe, niin prosessi keskeytyy virhe-ilmoitukseen. Tarkista koodi, tee tarvittavat korjaukset ja lähetä uudelleen.

Jos koodi ja kytkentä on oikein tehty, niin kukin kytkennän LED syttyy vuorollaan yhdeksi sekunniksi ja sammuu sen jälkeen järjestyksessä vihreä, keltainen ja punainen.

Liikennevalon ohjelmointi

Nyt tiedät, miten kytkennän LEDejä ohjataan Arduinon digitaalisella lähdöllä. Ohjelmoi seuraavaksi liikennevalo. Tähän ei anneta valmista koodia, vaan sinun tulee tehdä se itse. Alla olevassa kuvassa on esitetty liikennevalon toimintasekvenssi.



1. Ohjaa punainen LEDi päälle viiden sekunnin ajaksi.
2. Ohjaa keltainen LEDi päälle yhden sekunnin ajaksi.
3. Sammuta punainen ja keltainen LEDi ja ohjaa vihreä LEDi päälle kymmenen sekunnin ajaksi.
4. Sammuta vihreä LEDi ja ohjaa keltainen LEDi päälle kahden sekunnin ajaksi.
5. Sammuta keltainen LEDi.

Tee tarvittava koodi **loop-funktioon**, jossa sitä suoritetaan ikuisesti. Koodia tehdessäsi huomaat, että on paljon helpompaa ohjata oikeaa LEDiä, kun siihen voi viitata kuvaavalla vakion nimellä (esim. yLed → keltainen LEDi).

Ohjelmoinnissa tarvitset vain kahta komentoa oikein parametroituna:

- digitalWrite(pinni, HIGH/LOW)
- delay(<kesto ms>)

Testaa ohjelma kääntämällä se ja lähettämällä se Arduinoon. On yleistä, että ohjelmaa joutuu korjaamaan muutaman kerran ennen kuin se on täysin toimiva.

Painikeohjaus

Työn kuvaus

Digitaalisten lähtöjen lisäksi mikro-ohjaimessa on myös **digitaalisia tuloja**. Yleisesti mitä tahansa digitaalipinniä voi käyttää lähtönä tai tulona. Pinni määritetään lähdöksi / tuloksi ohjelmallisesti (pinMode). Pinneihin voidaan liittää monia erilaisia syöttö- ja tulostuslaitteita sekä toimintoja suorittavia moduuleita.

Tässä työssä rakennamme virtapiirin, jossa on kaksi toiminnan kannalta oleellista komponenttia. Uutena komponenttina käytämme painiketta, joka toimii yksinkertaisena syöttölaitteena (input). Tulostuslaitteena (output) toimii LEDi.

Teemme kytkennän ja ohjelman, joka toimii näin:

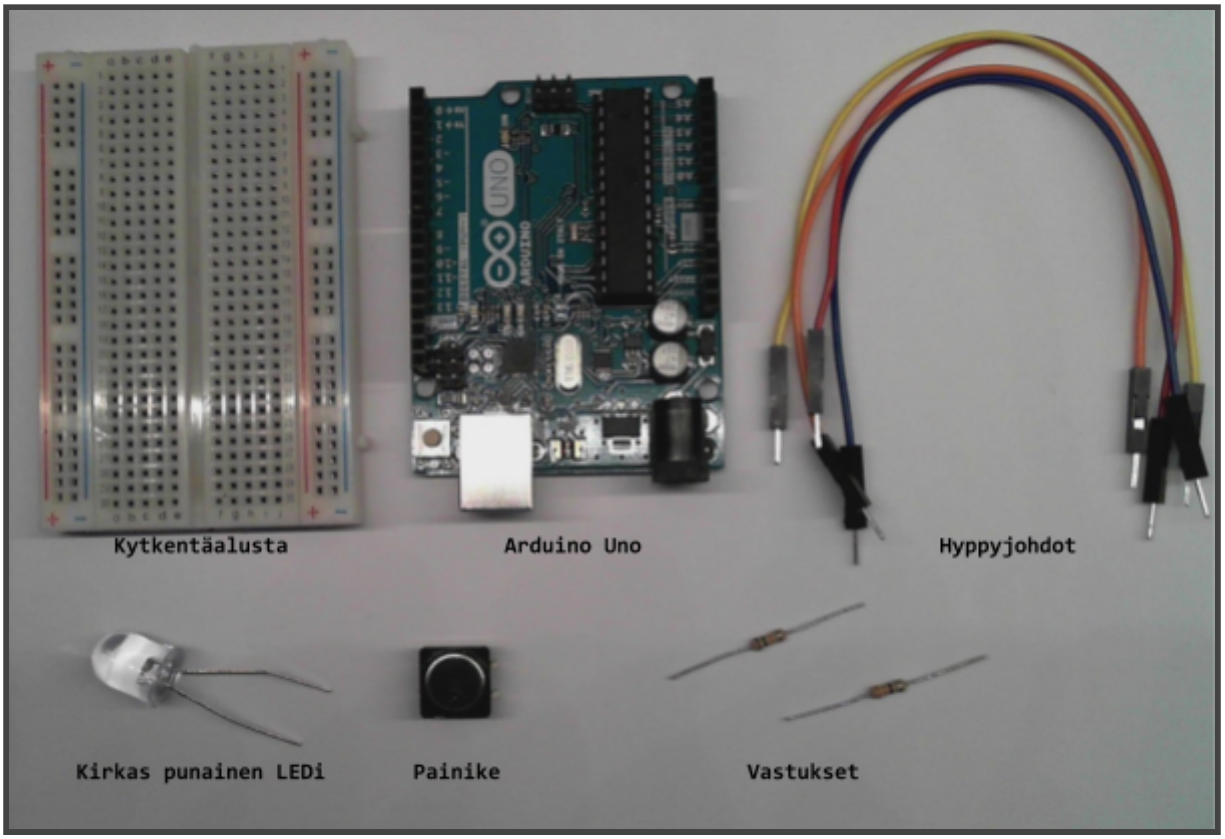
- Jos LEDi on pois päältä ja painiketta painetaan, niin
  - LEDi sytytetään päälle
  - Tietokoneelle lähetetään viesti “LED on”
- Jos LEDi on päällä ja painiketta painetaan, niin
  - LEDi sammutetaan
  - Tietokoneelle lähetetään viesti “LED off”

Viestin lähetys tietokoneelle tehdään USB-yhteyden kautta ja viestit tulevat näkyviin ohjelmointiympäristön sarjamonitoriin.

Tarvittavat komponentit

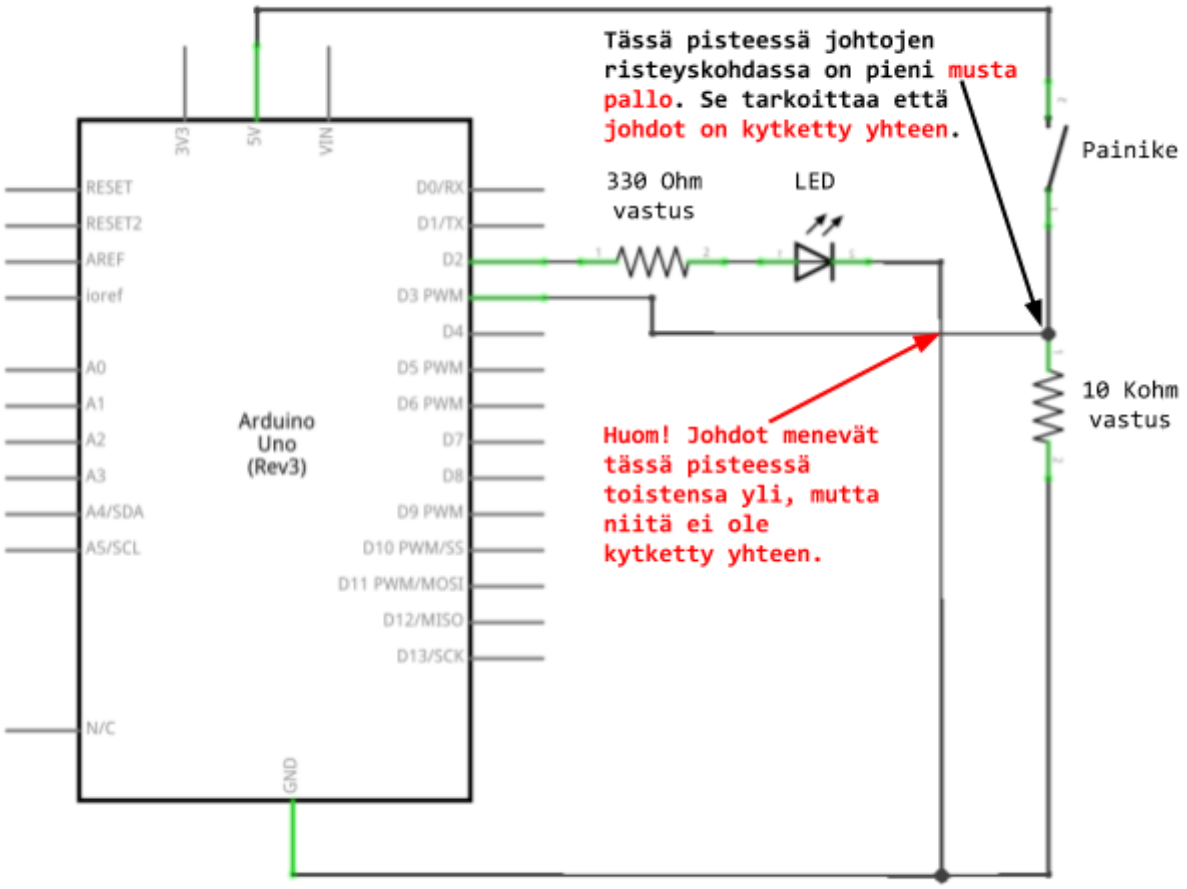
- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- Kirkas punainen LEDi
- Piirilevypainike
- 330 Ohmin vastus
- 10 Kohmin vastus
- Neljä hyppyjohdinta (uros-uros): punainen, sininen, keltainen ja oranssi
- Kytkentäalusta

Seuraavalla sivulla on kuva tarvittavista komponenteista.


Kytkentäalusta
Arduino Uno
Hyppyjohdot
Kirkas punainen LEDi
Painike
Vastukset

Kytkennän rakennus

Alla olevassa kuvassa on esitetty virtapiirin kytkentäkaavio.

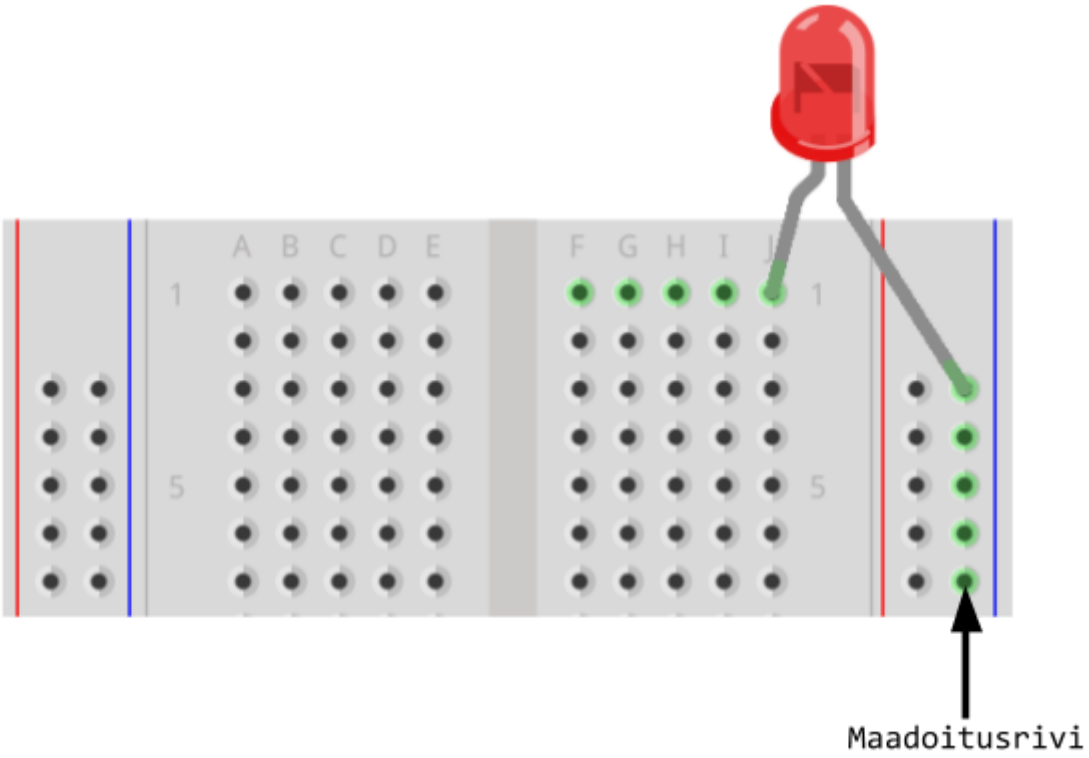


LEDin kytkennässä ei ole mitään uutta. Siinä on vastus ja LEDi sarjassa liitettynä Arduinon **pinnien D3** ja **GND** väliin. Pinniä D3 käytetään siis digitaalisena lähtönä ohjaamaan LEDiä.

Arduinon **pinni D2** on kytketty **10 Kohmin vastuksen kautta maahan** (GND) **sekä painikkeen kautta 5 Voltin jännitteeseen**. Kun painiketta ei paineta, pinnin tila on **LOW**. Kun painiketta painetaan, pääsee 5 Voltin jännite kytkeytymään pinniin ja sen tilaksi muuttuu **HIGH**. Koodissa voimme selvittää onko digitaalisen tulon tila LOW (painiketta ei painettu, ei jännitettä) vaiko HIGH (painiketta painettu, pinniin tulee 5 Voltin jännite). Digitaalinen tulo luetaan komennolla digitalRead().

Voit kokeilla rakentaa virtapiirin suoraan kytkentäkaavion perusteella.
Se onnistuu myös ohjatusti seuraavilla vaiheilla.

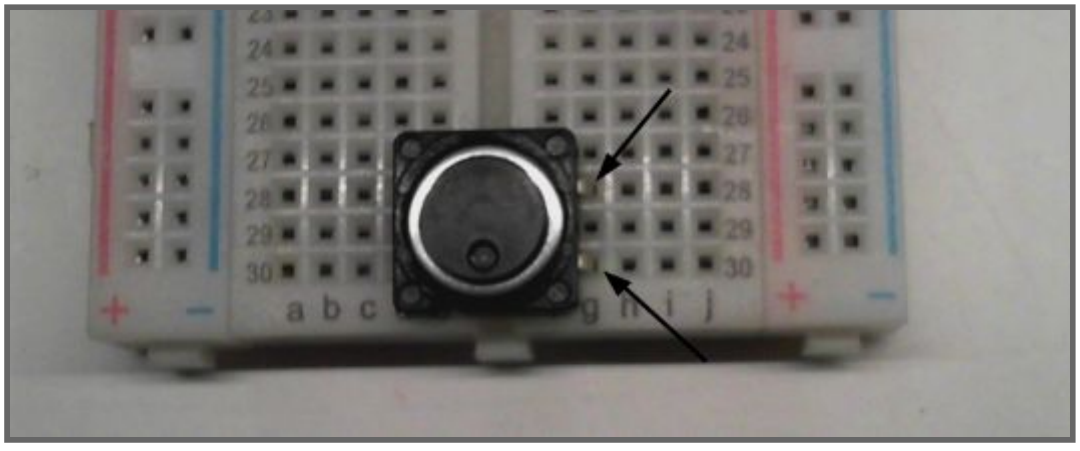
1.
Liitä LEDin pidempi jalka **pisteen J1** ja toinen jalka **maadoitusriviin**.



2.
Kytke 330 Ohmin vastus **pisteisiin D1** ja **F1** (sarjaan LEDin kanssa).

3.
Paina painike kytkentäalustan keskelle niin että painikkeen kaksi jalkaa tulee keskikohdan vasemmalle puolelle ja kaksi oikealle puolelle.

Vasemmanpuoleiset jalat ovat pisteissä D28 ja D30.
Oikeanpuoleiset jalat ovat pisteissä G28 ja G30 (käytetään kytkennässä)



4.
Kytke 10 Kohmin vastus **pisteeseen J30** ja vastuksen toinen jalka johonkin **maadoitusrivin pisteeseen** (ei väliä mihin pisteeseen).

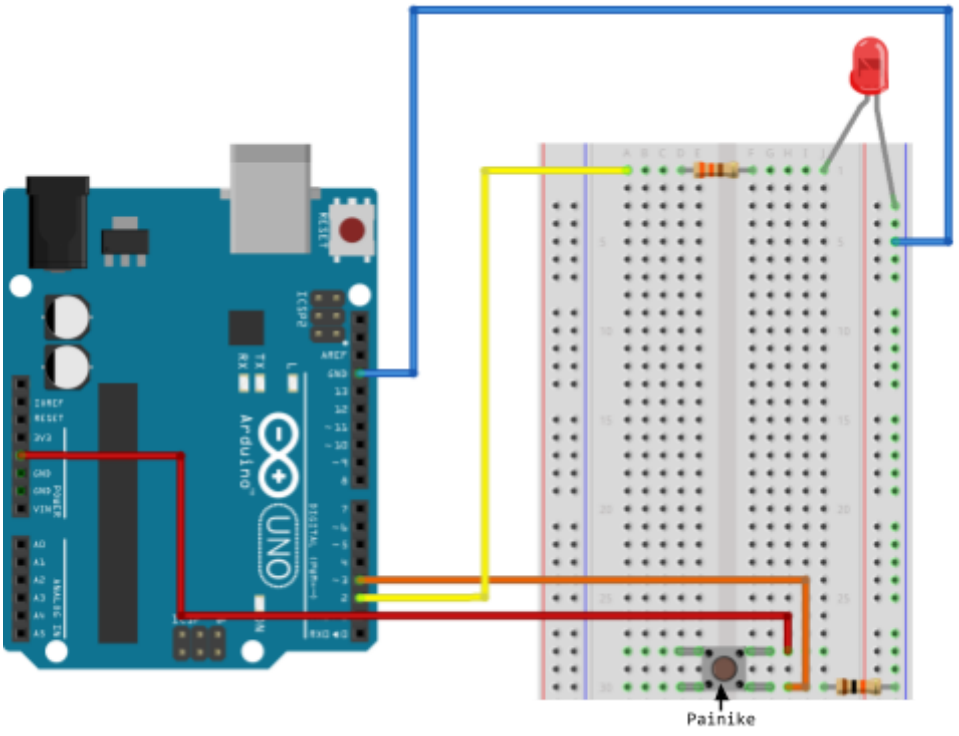
5.
Liitä punainen hyppyjohto Arduinon **pinniin +5V** ja johdon toinen pää **pisteeseen H28**.

6.
Liitä sininen hyppyjohto Arduinon **pinniin GND** (johonkin kolmesta, ei väliä mihin) ja johdon toinen pää johonkin **maadoitusrivin pisteeseen**.

7.
Liitä keltainen hyppyjohto Arduinon **pinniin D2** ja johdon toinen pää **pisteeseen A1** (digitaalinen lähtö ohjaa LEDiä).

8.
Liitä oranssi hyppyjohto Arduinon **pinniin D3** ja johdon toinen pää **pisteeseen H30** (digitaalinen tulo, lukee painikkeen painallukset).

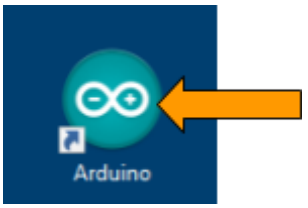
Valmis kytkentä



Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.



Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakiot

Määritetään vakiot, joilla viittaamme ohjelmassa **pinneihin D2** ja **D3**.

```
#define ledPin 2
#define btnPin 3
```

Muuttuja

Muuttuja on ohjelmassa varasto, johon voimme tallentaa tietoa, lukea tietoa siitä ja muuttaa siihen tallennetun tiedon arvoa. Arduino-ohjelmassa kaikki käytettävät muuttujat tulee aina esitellä muodossa **<muuttujan tyyppi> <muuttujan nimi>**. Muuttujan tyyppi määrittää, minkä tyyppistä tietoa muuttujaan voidaan tallentaa (kokonaisluku → int, desimaaliluku → float, merkki → char). Esittelyn yhteydessä muuttujalle voidaan myös antaa alkuarvo, niin kuin tässä on tehty.

```
int tila = 0;
```

setup-funktio
Määritetään **ledPin** (D2) toimimaan digitaalisena lähtönä ja **btnPin** puolestaan digitaalisena tulona (D2). Pinnistä tehdään tulo komennolla pinMode(<pinni>, INPUT).

Lopuksi avataan sarjaliikenneyhteys tietokoneen ja Arduinon välille USB-kaapelin kautta komennolla Serial.begin(<nopeus>). Nopeus on parametri ja sen arvo on bittiä per sekunti (bps). 9600 bps on ohjelmointiympäristön sarjamonitorin nopeuden oletusarvo, joten käytämme sitä. Tiedonsiirtonopeus riittää tähän tarkoitukseen oikein hyvin. Tarvittaessa on mahdollista toki käyttää myös paljon suurempia nopeuksia.

```
void setup() {
  Serial.begin(9600);          //avataan sarjaliikenneyhteys
  pinMode(ledPin, OUTPUT);     //digitaalinen lähtö
  pinMode(btnPin, INPUT);      //digitaalinen tulo
  digitalWrite(ledPin, LOW);   //LED varmuudeksi pois päältä
}
```

loop-funktio
Ohjelman toiminnot tapahtuvat painiketta painamalla. Tutkimme ehtorakenteella, onko pinnin **btnPin** tila **HIGH** (painiketta painettu). Huomaa että **yhtäsuuruus tutkitaan kahdella yhtäsuuruusmerkillä** (==). Yksi yhtäsuuruusmerkki tarkoittaa arvon antamista kohteelle.

```
void loop() {
  if (digitalRead(btnPin) == HIGH) {
  //tähän tulee painikkeen painalluksella
  //suoritettava koodi
  }
}
```

Muuttujassa **tila** pidetään yllä tietoa siitä, onko LEDi päällä vai ei. Sovitaan että kun muuttujan tila arvona on luku nolla, niin LEDi ei ole päällä ja arvo yksi tarkoittaa päällä olevaa LEDiä.

Tutkimme ehtorakenteella, onko muuttujan tila arvo yhtä suuri kuin 0:

- Jos on, sytytämme LEDin.
- Jos ei ole, sammutamme LEDin.

Tässä ehtorakenteessa meillä on mukana vaihtoehtoinen **else-osa**. Se suoritetaan silloin **kun ehto ei ole totta**.

```
void loop() {
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0) {
       //tähän tulee LEDin sytyttävä koodi
    }
    else {
    }
       //tähän tulee LEDin sammuttava koodi
  }
}
```

loop-funktion koodi kokonaisuudessaan

```
void loop() {
  //tutkitaan, onko painiketta painettu
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0) {              //jos LEDi ei ollut päällä
      tila = 1;                   //asetetaan tilan arvoksi luku 1
      digitalWrite(ledPin, HIGH); //sytytetään LEDi
      Serial.println("LED on");   //lähetetään viesti PC:lle
    }
    else {                        //jos LEDi oli päällä
      tila = 0;                   //asetetaan tilan arvoksi luku 0
      digitalWrite(ledPin, LOW);  //sammutetaan LEDi
      Serial.println("LED off");  //lähetetään viesti PC:lle
    }
    delay(200);                   //kytkinvärähtelyn eliminointi
  }
}
```

Muuttujan **tila** arvo vaihdetaan aina päinvastaiseksi (0 → 1 ja 1 → 0). LEDiä ohjataan totutusti komennolla **digitalWrite()**. Viesti tietokoneelle lähetetään komennolla Serial.println(“<viesti>”).

Hetki ennen painikkeen sulkeutumista ja avautumista voi aiheuttaa sarjan kytkinvärähtelyjä. Jos niitä ei eliminoida mikro-ohjain voi rekisteröidä suuren määrän painalluksia, vaikka käyttäjä olisi painanut painiketta vain yhden kerran. Pieni viive (200 ms) ohjelmassa painikkeen painalluksen havaitsemisen jälkeen eliminoi kytkinvärähtelyn vaikutuksen ohjelman toimintaan.

Valmis koodi

```
//vakioiden määritykset
#define ledPin 2
#define btnPin 3

//muuttujan esittely ja alustus
int tila = 0;

void setup() {
  Serial.begin(9600);          //avataan sarjaliikenneyhteys
  pinMode(ledPin, OUTPUT);     //digitaalinen lähtö
  pinMode(btnPin, INPUT);      //digitaalinen tulo
  digitalWrite(ledPin, LOW);   //LED varmuudeksi pois päältä
}

void loop() {
  //tutkitaan, onko painiketta painettu
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0) {               //jos LEDi ei ollut päällä
      tila = 1;                    //asetetaan tilan arvoksi luku 1
      digitalWrite(ledPin, HIGH); //sytytetään LEDi
      Serial.println("LED on");   //lähetetään viesti PC:lle
    }
    else {                         //jos LEDi oli päällä
      tila = 0;                    //asetetaan tilan arvoksi luku 0
      digitalWrite(ledPin, LOW);  //sammutetaan LEDi
      Serial.println("LED off");  //lähetetään viesti PC:lle
    }
    delay(200);                    //kytkinvärähtelyn eliminointi
  }
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

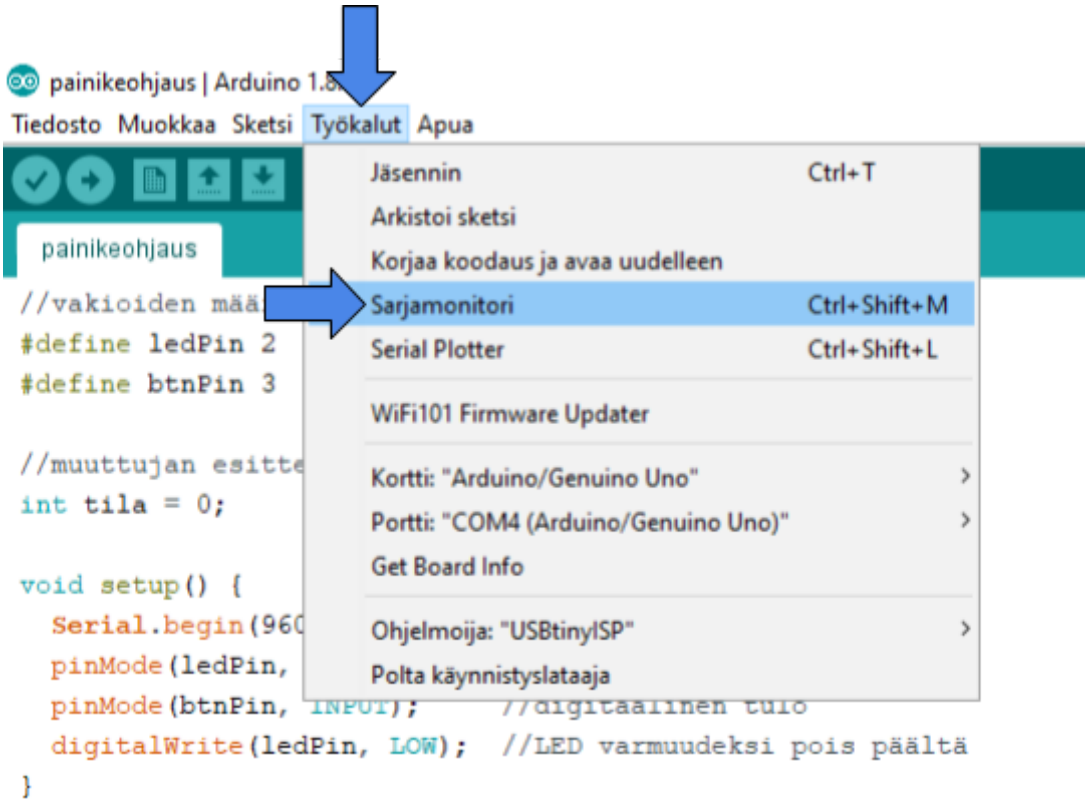
Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **"Lähetä"**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

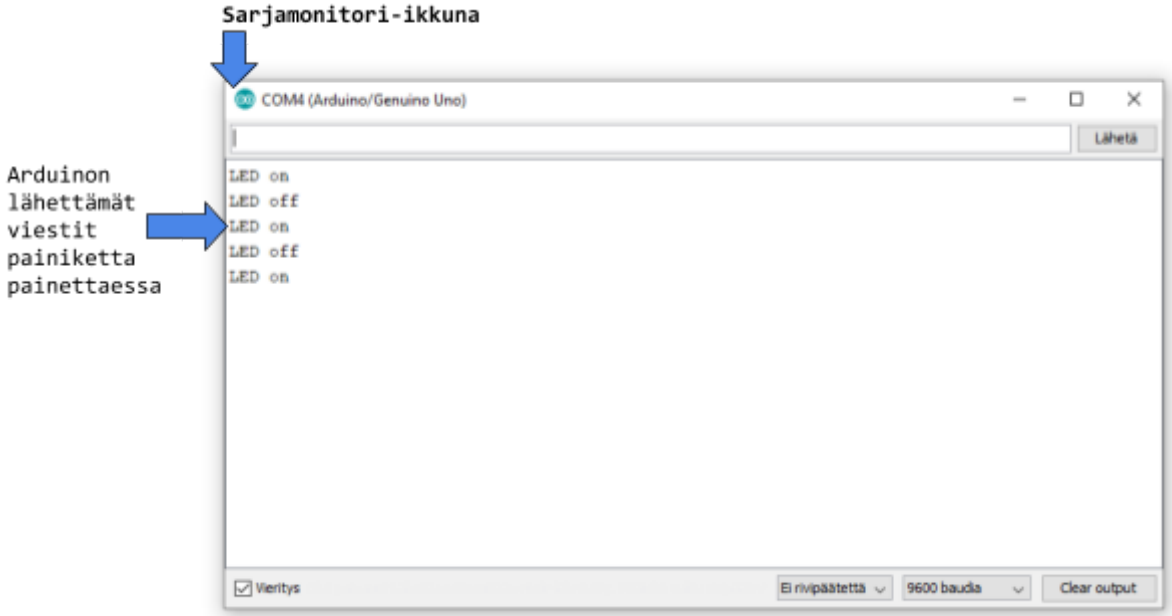
Jos ohjelman koodissa oli virhe, niin prosessi keskeytyy virhe-ilmoitukseen. Tarkista koodi, tee tarvittavat korjaukset ja lähetä uudelleen.

Testaus

Avaamalla sarjamonitorin näet Arduinon tietokoneelle lähettämät viestit. Sarjamonitori avataan valikosta **Työkalut / Sarjamonitori**.



Sarjamonitori aukeaa omaan ikkunaan. Painele kytkennän painiketta niin näet tietokoneelle tulevat viestit.



Käytämme tätä samaa kytkentää myös seuraavassa työssä (Ohjaus tietokoneella), joten säilytä kytkentä seuraavaa kertaa varten.

Ohjaus tietokoneella - Tietokoneella ohjattava laite

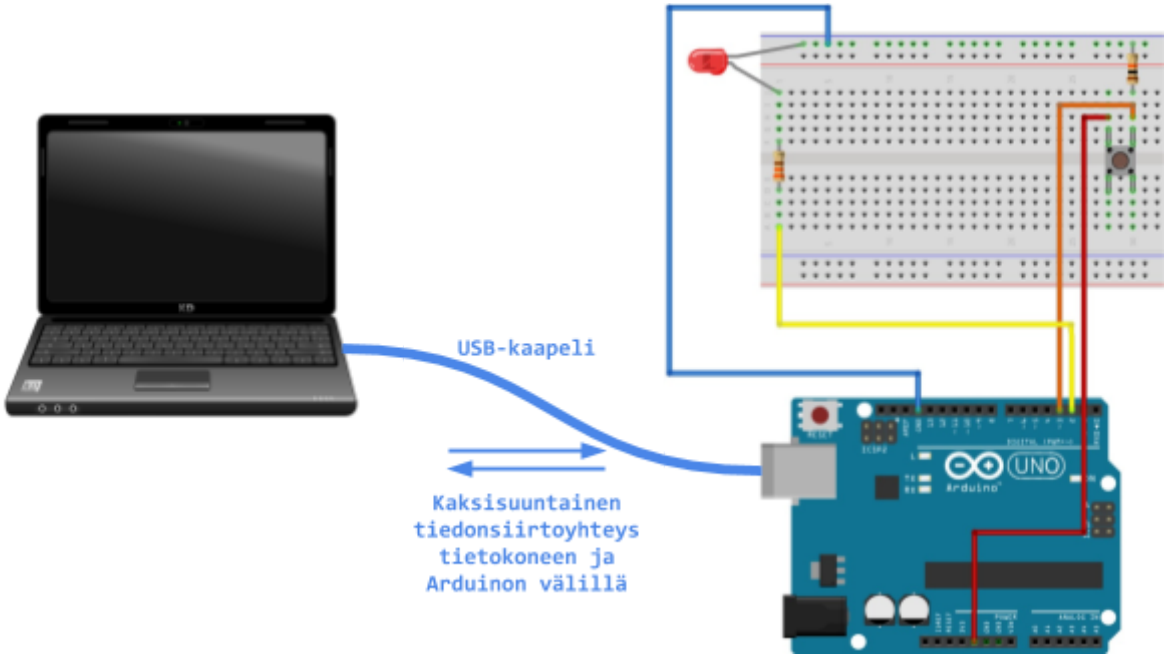
Työn lähtökohdat

Tässä työssä käytetään täysin samaa kytkentää kuin projektissa "Painikeohjaus". Jos sinulla ei ole tätä kytkentää, niin rakenna se ensin linkistä löytyvän työohjeen mukaisesti. Kytkentään ei tarvitse tehdä mitään muutoksia.

Niin ikään käytämme myös ohjelman pohjana "Painikeohjaus"-projektin koodia. Lataa kooditiedosto Arduinon ohjelmointiympäristöön ja tallenna se heti aluksi uudella nimellä, jotta uusi ohjelma ei mene vanhan päälle.

Työn kuvaus

Tässä työssä teemme tietokoneesta ohjauslaitteen, jonka lähettämä viesti kytkee LEDin päälle ja pois päältä. Viestit tietokoneen ja Arduinon välillä kulkevat USB-kaapelin kautta sarjaliikenneyhteydellä. Samanaikaisesti säilytämme myös painikkeen toiminnan niin, että LEDiä voi ohjata sekä tietokoneella että painikkeella.



Aluksi sovimme LEDin ohjauksessa käytettävät merkit (protokolla):

- Kun tietokone lähettää mikro-ohjaimelle merkin **1** (numero yksi), niin **LEDi sytytetään**.
- Kun tietokone lähettää mikro-ohjaimelle merkin **0** (numero nolla), niin **LEDi sammutetaan**.

Ohjelmointi

Aloitus

Aluksi sinulla tulee olla painikeohjaus-työn koodi ohjelmointi-ympäristössä uudella nimellä tallennettuna.

Muuttujat

Lisätään ohjelman alkuun yhden uuden muuttujan esittely. Muuttujan tyyppi on **char** (merkki) ja sen nimi on **cInput**. Tähän muuttujaan tullaan myöhemmin tallentamaan tietokoneen lähettämä viesti yksittäisenä merkkinä.

```
//muuttujan esittely ja alustus
int tila = 0;
char cInput;
```

Omat funktiot

Säilytämme koodissa edelleen mahdollisuuden ohjata LEDiä painikkeella, mutta tämän rinnalle ohjelmoimme toiminnon, joka mahdollistaa LEDin ohjaamisen tietokoneen lähettämällä viestillä. LEDin ohjaaminen on samanlaista molemmissa tapauksissa, sekä painikkeen painalluksella että tietokoneen lähettämällä viestillä. Samaa koodia ei ole mielekästä toistaa ohjelmassa kahteen kertaan, joten teemme kaksi omaa funktiota. Ensimmäinen **led_on()-funktio** sytyttää LEDin ja **led_off()-funktio** sammuttaa LEDin. Painikeohjaus ja ohjaus tietokoneella käyttävät sitten kumpikin näitä samoja funktioita LEDin ohjaamiseen.

Omien funktioiden rungot:

```
//tämä funktio tekee LEDin sytyttämiseen tarvittavat toiminnot
void led_on() {

}

//tämä funktio tekee LEDin sammuttamiseen tarvittavat toiminnot
void led_off() {

}
```

Leikkaa aiemmasta koodista kolme koodiriviä, ehtorakenteesta **if (tila == 0**) ja liitä ne funktioon **led_on()** näin:

```
//tämä funktio tekee LEDin sytyttämiseen tarvittavat toiminnot
void led_on() {
  tila = 1;                     //asetetaan tilan arvoksi luku 1
  digitalWrite(ledPin, HIGH);   //sytytetään LEDi
  Serial.println("LED on");     //lähetetään viesti PC:lle
}
```

Leikkaa saman ehtorakenteen **else-osasta** kolme koodiriviä pois ja liitä ne funktioon **led_off()**.

```
//tämä funktio tekee LEDin sammuttamiseen tarvittavat toiminnot
void led_off() {
  tila = 0;                     //asetetaan tilan arvoksi luku 0
  digitalWrite(ledPin, LOW);    //sammutetaan LEDi
  Serial.println("LED off");    //lähetetään viesti PC:lle
}
```

Nyt LEDin ohjaamiseen tarvittavat funktiot ovat valmiit.

Ohjaus painikkeella

Ehtorakenteesta **if (tila == 0 ... else...** poistettujen koodirivien tilalle kirjoitetaan nyt vain tarvittavat **funktiokutsut**. Myös aaltosulut poistetaan, koska ehtorakenteen **then**- ja **else**-osissa on vain yksi koodirivi (aaltosulut tarvitaan vain, jos niissä on kaksi tai useampi rivi koodia).

Tässä vaiheessa **loop-funktio** on tällainen:

```
void loop() {
  //tutkitaan, onko painiketta painettu
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0)
      led_on();   //funktiokutsu
    else
      led_off();  //funktiokutsu
    delay(200);
  }
}
```

Funktiokutsu siirtää koodin suorituksen sen nimen mukaiseen funktioon. Funktion koodin suorituksen jälkeen ohjelman suoritusta jatketaan **loop-funktiossa** (pääohjelma).

Ohjaus tietokoneella

Lisää loop-funktioon ehtorakenne

```
  if (Serial.available()) {
  }
```

```
void loop() {
  //tutkitaan, onko painiketta painettu
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0)
      led_on();   //funktiokutsu
    else
      led_off();  //funktiokutsu
    delay(200);   //kytkinvärähtelyn eliminointi
  }

  //tutkitaan, onko mikro-ohjaimen sarjaporttiin tullut tietoa
  if (Serial.available()) {
  }
}
```

Serial.available() funktio palauttaa mikro-ohjaimen sarjaporttiin tulleiden tavujen lukumäärän. Jos mitään luettavaa tietoa ei ole tullut, niin funktio palauttaa arvon nolla ja silloin ehtorakenteen **then**-osaa ei suoriteta.

Kun tietokone lähettää ohjaustiedon, niin tieto menee mikro-ohjaimen sarjaporttiin ja yllä oleva funktio palauttaa **arvon 1**. Silloin suoritetaan ehtorakenteen **then**-osa, jossa luetaan tieto sarjaportista ja käsitellään se.

Sarjaportin puskuriin tullut tieto luetaan funktiolla Serial.read() ja sijoitetaan muuttujan **cInput** arvoksi.

```
if (Serial.available()) {
   cInput = Serial.read();  //luetaan tieto muuttujaan cInput
 }
```

Lopuksi lisäämme koodin, joka tutkii oliko vastaanotettu merkki 1 vai 0. Merkin 1 ascii-koodi on 49 ja merkin 0 ascii-koodi on 48 (ascii-taulukko). Ascii-standardissa on määritelty, millä numerolla mikin merkki tiedonsiirrossa koodataan.

Kun tietokone on lähettänyt mikro-ohjaimelle merkin 1, muuttujassa cInput on siis luku 49. Ja kun tietokone on lähettänyt merkin 0, muuttujassa on luku 48.

Nyt voimme ehtorakenteella tutkia:

- Jos **cInput == 49**, niin kutsutaan funktiota **led_on()**
- Jos **cInput == 48**, niin kutsutaan funktiota **led_off()**

```
//tutkitaan, onko mikro-ohjaimen sarjaporttiin tullut tietoa
 if (Serial.available()) {
   cInput = Serial.read();      //luetaan tieto muuttujaan cInput
   if (cInput == 49) led_on();  //funktiokutsu
   if (cInput == 48) led_off(); //funktiokutsu
 }
```

Ohjelma on nyt valmis. Seuraavalla sivulla on esitetty ohjelman koko koodi.

Valmis koodi

```
//vakioiden määritykset
#define ledPin 2
#define btnPin 3


//muuttujien esittely ja alustus
int tila = 0;
char cInput;


void setup() {
  Serial.begin(9600);          //avataan sarjaliikenneyhteys
  pinMode(ledPin, OUTPUT);     //digitaalinen lähtö
  pinMode(btnPin, INPUT);      //digitaalinen tulo
  digitalWrite(ledPin, LOW);  //LED varmuudeksi pois päältä
}


//tämä funktio tekee LEDin sytyttämiseen tarvittavat toiminnot
void led_on() {
  tila = 1;                       //asetetaan tilan arvoksi luku 1
  digitalWrite(ledPin, HIGH);     //sytytetään LEDi
  Serial.println("LED on");       //lähetetään viesti PC:lle
}


//tämä funktio tekee LEDin sammuttamiseen tarvittavat toiminnot
void led_off() {
  tila = 0;                      //asetetaan tilan arvoksi luku 0
  digitalWrite(ledPin, LOW);     //sammutetaan LEDi
  Serial.println("LED off");     //lähetetään viesti PC:lle
}
```

```
void loop() {
  //tutkitaan, onko painiketta painettu
  if (digitalRead(btnPin) == HIGH) {
    if (tila == 0)
      led_on();   //funktiokutsu
    else
      led_off();  //funktiokutsu
    delay(200);
  }

  //tutkitaan, onko mikro-ohjaimen sarjaporttiin tullut tietoa
  if (Serial.available()) {
    cInput = Serial.read();      //luetaan tieto muuttujaan cInput
    if (cInput == 49) led_on();  //funktiokutsu
    if (cInput == 48) led_off(); //funktiokutsu
  }
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

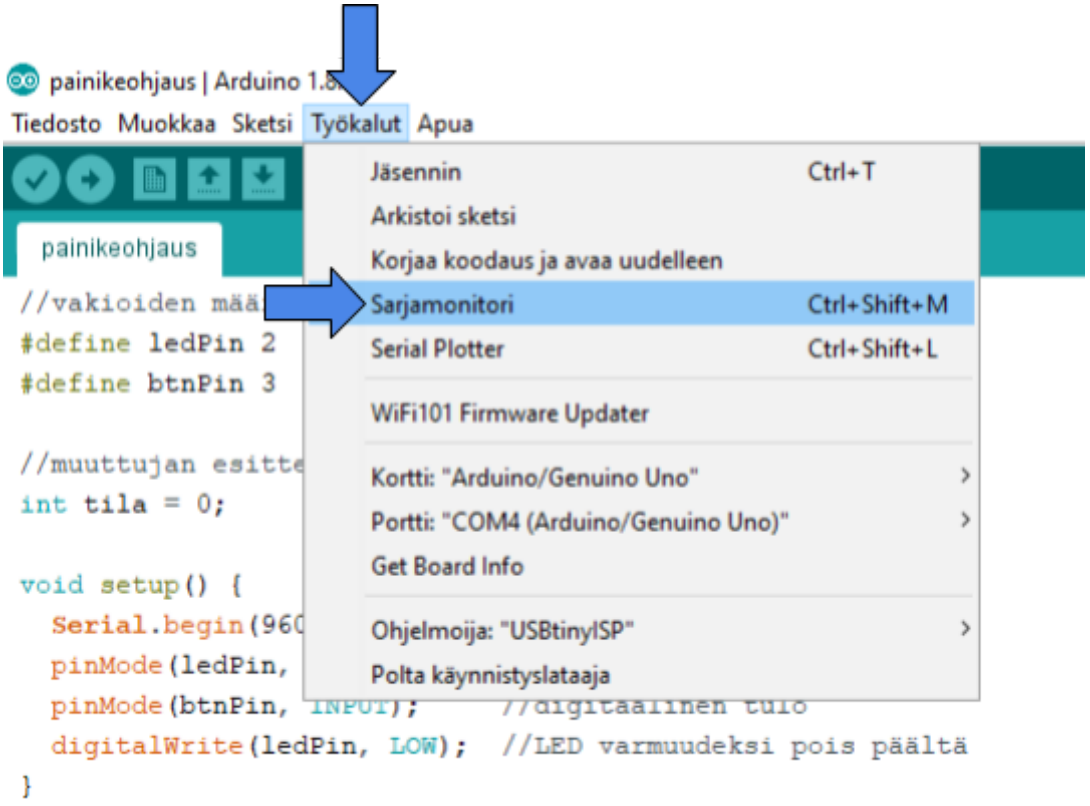
Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **“Lähetä”**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

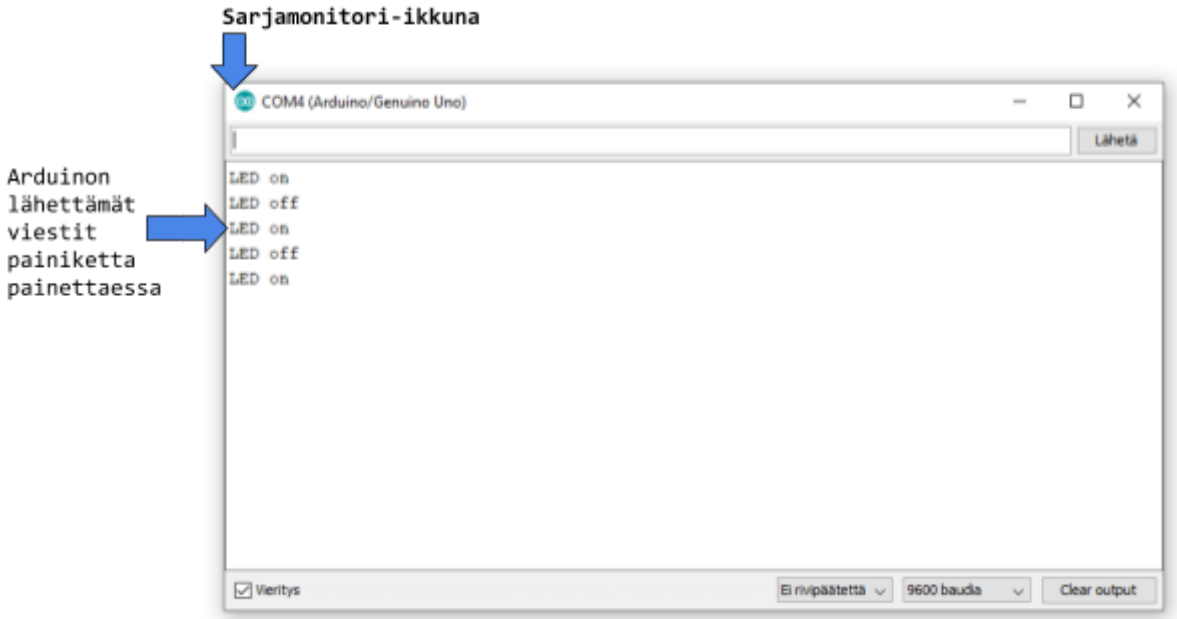
Jos ohjelman koodissa oli virhe, niin prosessi keskeytyy virhe-ilmoitukseen. Tarkista koodi, tee tarvittavat korjaukset ja lähetä uudelleen.

Testaus

Avaa sarjamonitori valikosta **Työkalut / Sarjamonitori**.

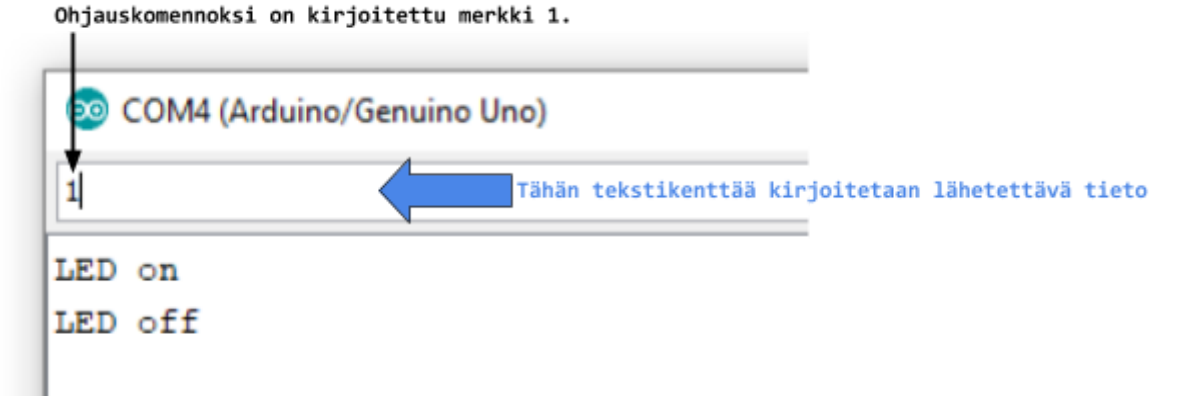


Sarjamonitori aukeaa omaan ikkunaan. Testaa aluksi painikkeen toiminta. LEDin tulee syttyä ja sammua painikkeesta ja sarjamonitoriin tulla viesti LEDin tilasta.



Kokeile sitten ohjausta tietokoneella. Sarjamonitoria voi käyttää myös tiedon lähettämiseen mikro-ohjaimelle.

Kirjoita sarjamonitori-ikkunan yläosassa olevaan tekstikenttään **numero 1** ja paina **Enter**. Sarjamonitori lähettää tiedon mikro-ohjaimelle USB-kaapelia pitkin ja LEDi syttyy. Syttymisen jälkeen sarjamonitoriin tulee viesti "LED on". Kokeile sammuttaa LED lähettämällä **numero 0**. Kokeile ohjata LEDiä sekä painikkeella että tietokoneella.



Yhteenveto

Tätä ohjetta soveltamalla voidaan rakentaa ja ohjelmoida tietokoneella ohjattavia laitteita. Ohjattava laite on Arduino mikro-ohjaimen

yhteyteen rakennettu oma virtapiiri, moduuli tai jokin ulkoinen laite. Ohjaussignaali siirtyy tietokoneelta sarjamuotoisesti USB-kaapelia pitkin mikro-ohjaimeen, jossa se luetaan ja käsitellään ohjauksen suorittamiseksi.

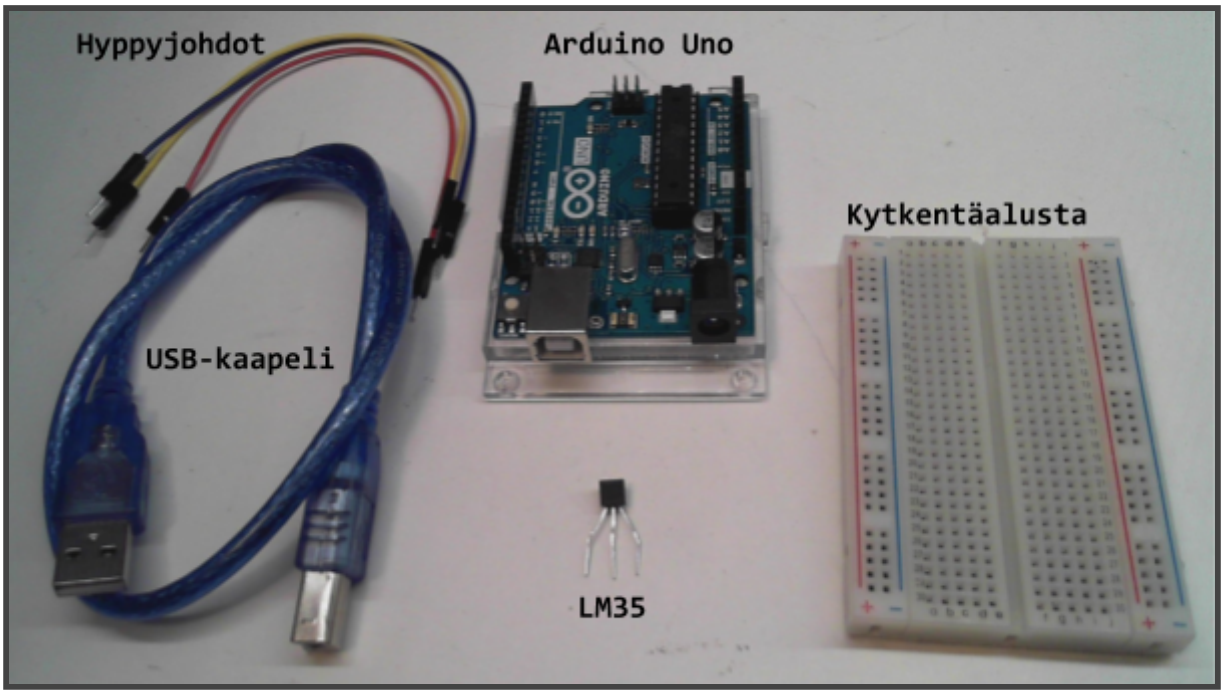
Lämpömittari analogisella anturilla

Työn kuvaus

Työssä rakennetaan ja ohjelmoidaan lämpömittari. Lämpötilan mittaavana anturina käytetään analogista anturia LM35. Arduino ohjelmoidaan lukemaan anturin lähettämä analoginen signaali ja laskemaan siitä signaalin arvoa vastaava lämpötila. Laskemisen jälkeen Arduino lähettää lämpötilan USB-liitännän kautta tietokoneelle. Ohjelmointiympäristön sarjamonitori (Serial monitor) -toiminnon kautta mikro-ohjaimen lähettämät lämpötilat nähdään tietokoneen näytöllä.

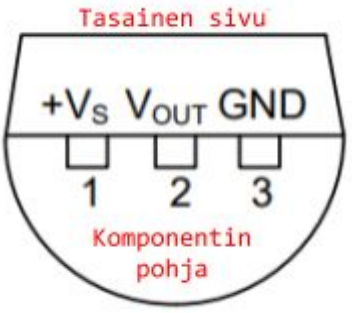
Tarvittavat komponentit

- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- LM35-lämpöanturi
- Kytkentäalusta
- Kolme hyppyjohtoa (uros-uros): punainen, musta ja keltainen



Kytkennän rakennus

Aloitetaan kytkennän rakennus LM35-anturista. Viereinen kuva esittää komponenttia alhaaltapäin katsottuna niin, että komponentin tasainen sivu on ylöspäin.

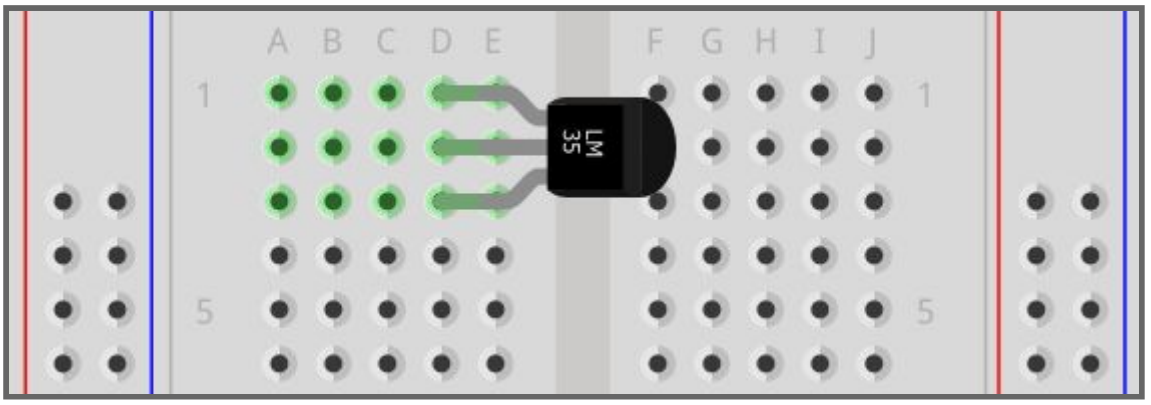


Pinni 1 (+Vs) tarkoittaa anturin **positiivista käyttöjännitettä**. **Pinnistä 2** (Vout) lähtee ulos **anturin analoginen signaali**. **Pinni 3** (GND) on **yhteinen maa** käyttöjännitteelle ja signaalille.

Paina anturi kytkentäalustalle näin:

- **Pinni 1** pisteeseen **D1** (sarake D, rivi 1).
- **Pinni 2** pisteeseen **D2**.
- **Pinni 3** pisteeseen **D3**.

Komponentin tasainen sivu osoittaa vasemmalle.

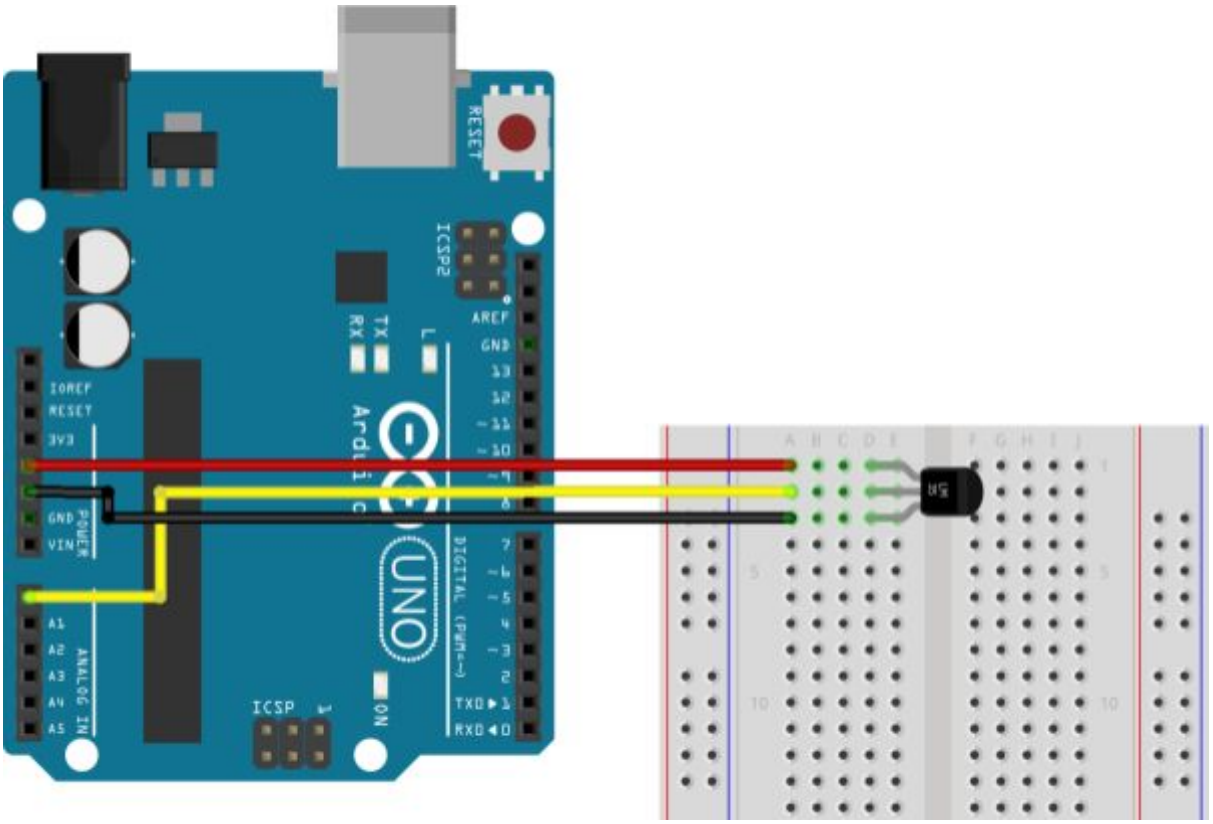


Anturin käyttöjännite
Kytke punainen hyppyjohto kytkentäalusta **pisteeseen A1** ja johdon toinen pää Arduinon **pinniin 5V**. Tätä johdinta pitkin Arduinosta tulee 5 Voltin käyttöjännite anturille.

Anturin signaali
Kytke keltainen hyppyjohto kytkentäalustan **pisteeseen A2** ja johdon toinen pää Arduinon **pinniin A0**. Tätä johdinta pitkin anturi lähettää analogisen signaalin Arduinon analogiseen tuloon A0.

Maajohto (GND)
Kytke musta hyppyjohto kytkentäalustan **pisteeseen A3** ja johdon toinen pää Arduinon **pinniin GND**. Arduino Unossa on kolme GND-pinniä, eikä ole väliä mihin niistä johtimen kytket.

Valmis kytkentä:



Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.

Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Muuttujat

Käytämme ohjelmassa kahta **kokonaislukutyyppistä** muuttujaa. Esitellään ne heti ensimmäiseksi koodin alussa. Muuttujaan **aValue** luemme analogisen signaalin arvon ja muuttujaan **temp** laskemme signaalin arvoa vastaavan lämpötilan ja lähetämme sen tietokoneelle. muuttujatyyppi int

```
int aValue;
```

```
int temp;
```

setup-funktio
Kerran käynnistyksen yhteydessä suoritettavan koodin sijoitamme setup-funktioon. Avataan sarjaliikenneyhteys Arduinon ja tietokoneen välille funktiolla Serial.begin(9600). Parametri 9600 ilmoittaa tiedonsiirtonopeuden arvona **b**its **p**er **s**econd (bittiä sekunnissa).

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

Jatkuvasti toistettavan koodin sijoitamme **loop**-funktioon.

Analogisen signaalin luku
Lämpöanturin signaalijohto on liitetty Arduinon **pinniin A0**. Luetaan tähän pinniin tulevan signaalin (jännitteen) arvo komennolla analogRead(A0) ja tallennetaan arvo muuttujaan **aValue**.

```
aValue = analogRead(A0);
```

Lämpötilan laskenta
Anturin lähettämä analoginen signaali on jännitetaso väliltä 0-5 Volttia. Signaalin jännite muuttuu 10 millivolttia yhtä Celsius-astetta kohti ja analogisen signaalin arvo voi olla mikä tahansa kokonaisluku väliltä 0-1023.

Lämpötila lasketaan kaavalla:

temp = Vs * Vout * 100.0 / 1024

Meidän kytkennässä **Vs** (käyttöjännite) on **5.0 Volttia** ja **Vout** on nyt muuttujan **aValue arvo**. Kirjoitamme kaavan koodiksi näin:

```
temp = (int)5.0 * aValue * 100.0 / 1024;
```

Kaavan alussa oleva (int) tekee tyyppimuunnoksen desimaaliluvusta kokonaisluvuksi. Tämä on tarpeen, sillä kaavassa on mukana jakolasku-operaattori ja se tuottaa tulokseksi desimaaliluvun.

Lämpötilan lähetys tietokoneelle

Seuraavaksi lähetämme mitatun ja lasketun lämpötilan tietokoneelle USB-liitännän kautta. Lähetys tehdään funkioilla Serial.print() ja Serial.println(). Tehdään lähetys kolmessa vaiheessa:

1. Lähetetään merkkijono "**Lämpötila = "**.
2. Lähetetään muuttujan **temp** arvo.
3. Lähetetään merkkijono " **C**" ja rivinvaihto.

```
Serial.print("Lämpötila = ");
Serial.print(temp);
Serial.println(" C");
```

Lopuksi lisäämme vielä yhden sekunnin viiveen funktiolla delay(). Lämpötila mitataan, lasketaan ja lähetetään tietokoneelle yhden

sekunnin välein. Komennon parametri ilmoittaa viiveen pituuden millisekunteinta ja 1000 ms = 1 sekunti.

```
delay(1000);
```

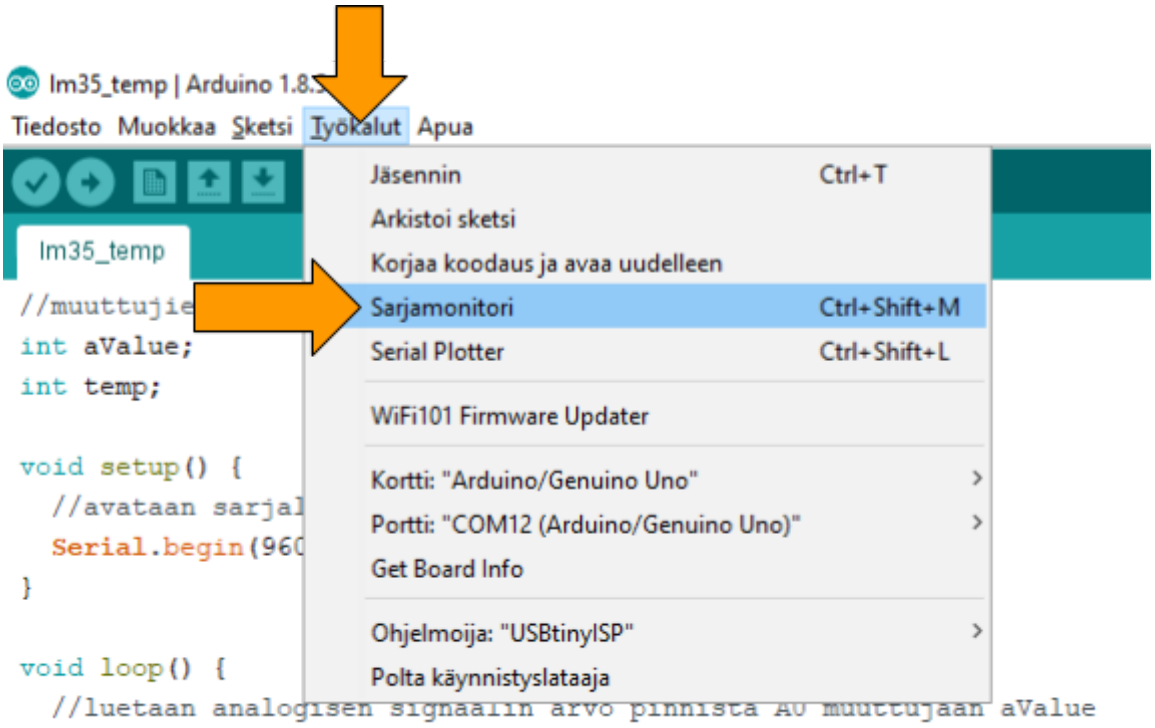
Valmis koodi

```
//muuttujien esittely
int aValue;
int temp;

void setup() {
  //avataan sarjaliikenneyhteys nopeudella 9600 bps
  Serial.begin(9600);
}

void loop() {
  //luetaan analogisen signaalin arvo
  //pinnistä A0 muuttujaan aValue
  aValue = analogRead(A0);

  //lasketaan analogisen signaalin arvoa
  //vastaava lämpötila Celsius-asteina
```

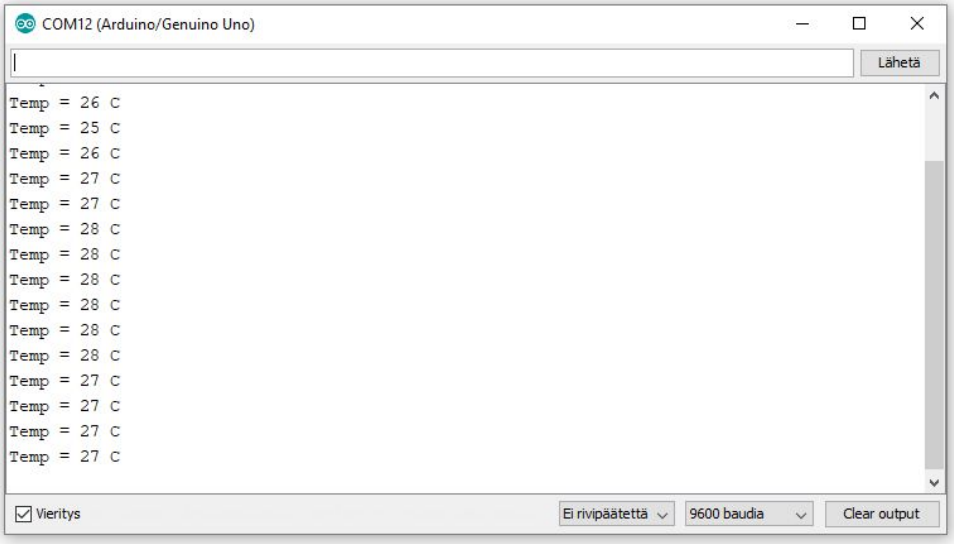
```
  temp = (int)5.0 * aValue * 100.0 / 1024;

  //tulostetaan lämpötila tietokoneelle USB-kaapelin kautta
  Serial.print("Lämpötila = ");
  Serial.print(temp);
  Serial.println(" C");

  //yhden sekunnin viive
  delay(1000);
}
```

Testaus

Liitä Arduino USB-kaapelilla tietokoneeseen ja lataa ohjelma Arduinoon.
Kun lataus on valmis, käynnistä sarjamonitori valikosta **Työkalut / Sarjamonitori.**



Ruutuun aukeaa ikkuna, josta näet Arduinon USB-yhteyden kautta lähettämän datan.



Ota anturista sormilla kiinni, jolloin se lämpenee hitaasti. Lämpötilan nousu näkyy mittaustuloksissa. Päästä irti ja lämpötila alkaa laskea hitaasti.

Lämpötilaplotteri

Plotteri on laite, joka esittää graafisesti analogisen suureen arvon ajan funktiona. Tulostusalue on kaksiulotteinen taso, X-akselilla on aika ja

Y-akselilla suureen arvo. Tulosteesta nähdään nopeasti miten suuren arvo on muuttunut.

Voimme muuttaa lämpömittarimme hyvin helposti lämpötilaa mittaavaksi plotteriksi. Tallenna ohjelma aluksi toisella nimellä, ettei tekemämme muutokset mene aiemman tiedoston päälle.

Muuta lämpötilan lähetys tietokoneelle näin:

```
//tulostetaan lämpötila tietokoneelle USB-kaapelin kautta

//Serial.print("Lämpötila = ");
Serial.println(temp);
//Serial.println(" C");
```

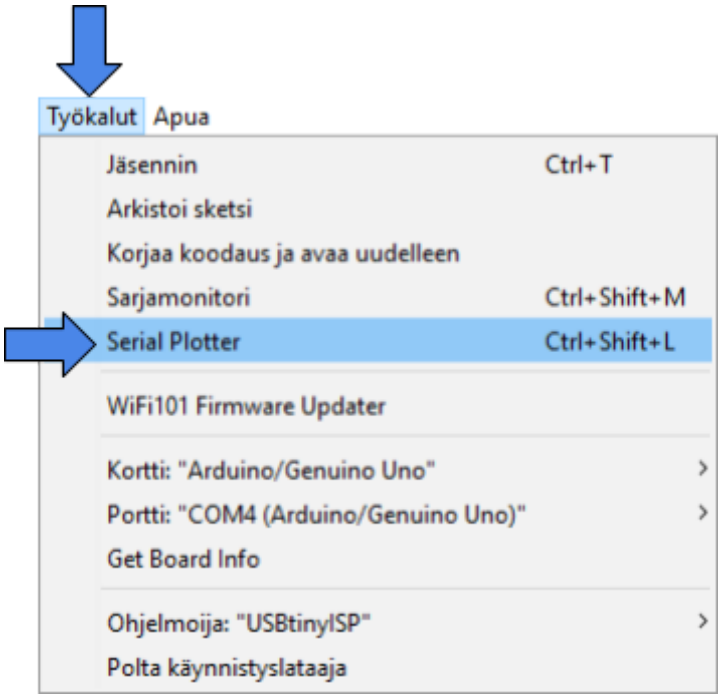
Lähetyksen suorittavasta kolmesta koodirivistä ensimmäinen ja viimeinen rivi on muutettu **kommentiksi** lisäämällä rivien alkuun kaksi kenoviivaa. Keskimmäiseen riviin on **print()-funktion tilalle** vaihdettu funktio **println()**. Se lisää rivinvaihdon lähetyksen loppuun. Näillä muutoksilla lähetämme tietokoneelle siis pelkästään lämpötilan numeerisena arvoja ja rivinvaihdon sen perään.

Plotteri tarvitsee tietoa nopeammin kuin sekunnin välein. Muuta funtion delay() parametriksi luku 100. Lähetämme lämpötilan tietokoneelle 100 millisekunnin välein, eli kymmenen kertaa sekunnissa.

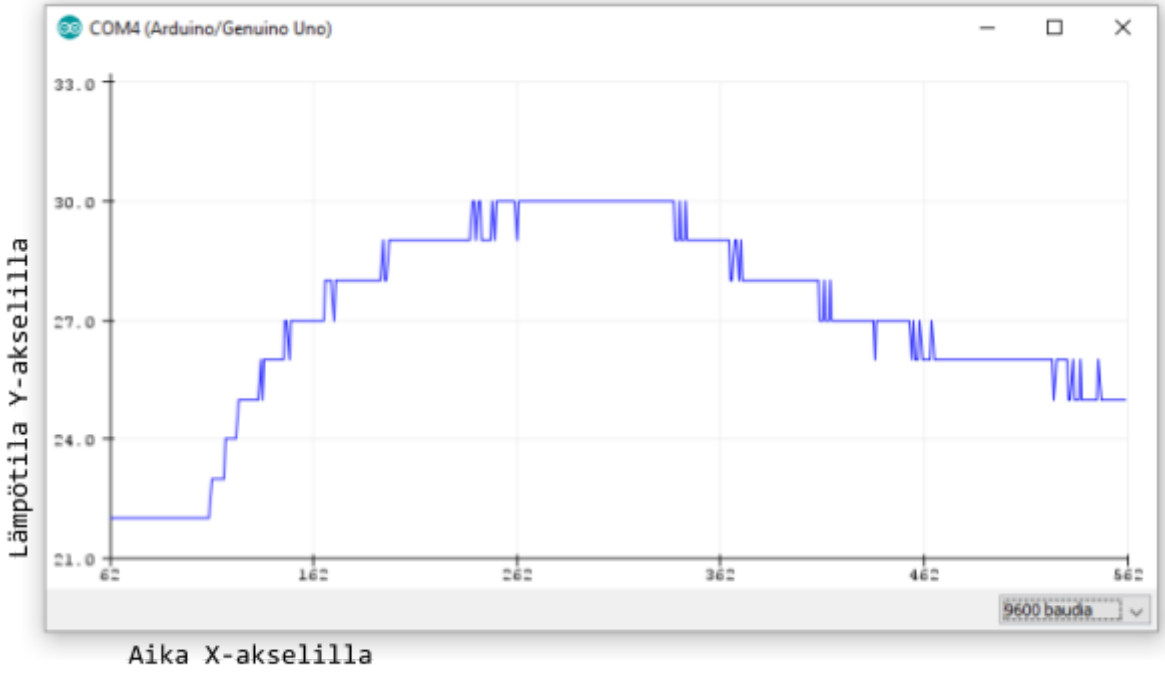
```
//100 ms viive
delay(100);
```

Ohjelma on valmis, lataa se Arduinoon.

Latauksen valmistuttua käynnistä plotteri valikosta **Työkalut / Serial Plotter**.



Plotterin piirtämästä kuvaajasta näet, miten lämpötila on muuttunut ajan funktiona. Kokeile lämmittää anturia ottamalla siitä sormilla kiinni ja päästä hetken päästä irti. Kuvaajassa näet lämpötilan muutokset.



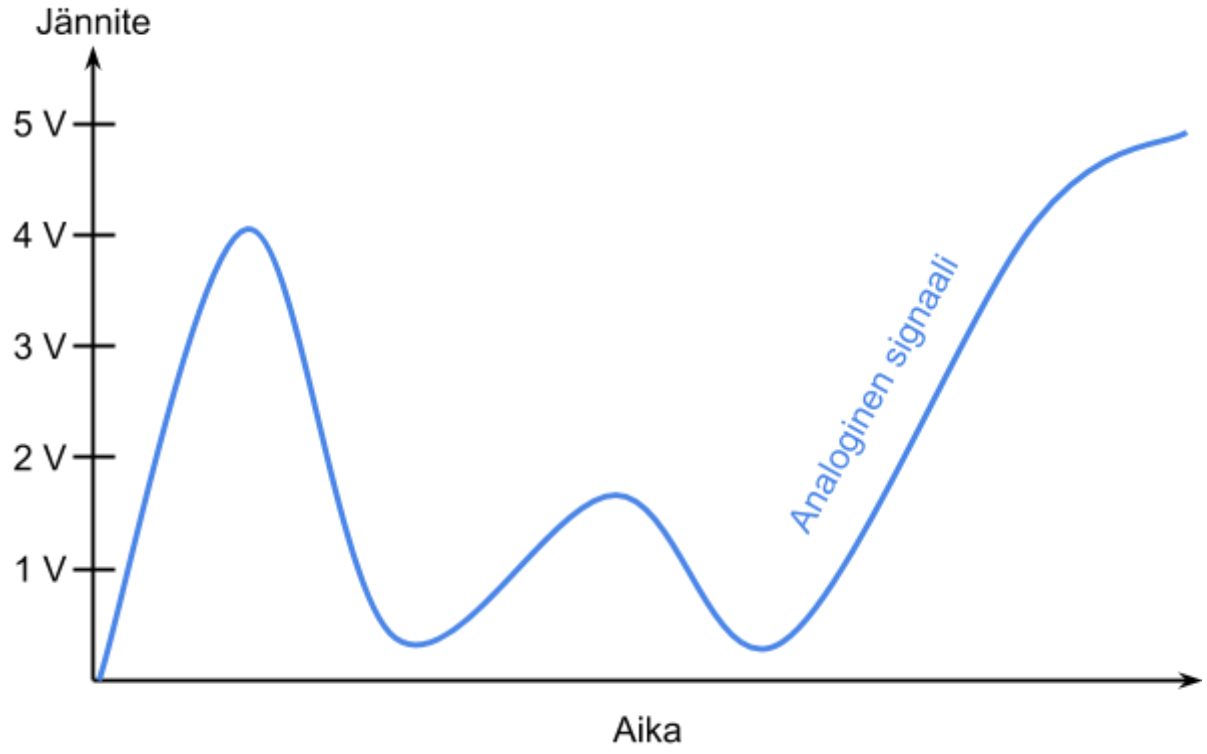
Analoginen signaali ja potentiometri

Työn kuvaus

Tässä työssä perehdymme tarkemmin analogiseen signaaliin ja potentiometriin. Potentiometri on hyvin yleinen analogisen signaalin tuottava komponentti. Opimme kytkemään potentiometrin mikro-ohjaimeen ja ohjelmoimaan sen lukemiseen tarvittavan koodin. Tämä luo pohjan seuraavalle työlle, jossa potentiometriä käytetään säätimenä.

Analoginen signaali

Perehdytään aluksi tarkemmin analogiseen signaaliin. Analogisella signaalilla on äärettömän monta arvoa (digitaalisella signaalilla vain kaksi, 0 ja 1). Mikro-ohjaimen käyttöjännite asettaa ylärajan analogisen signaalin arvolle. Arduino Unon käyttöjännite on 5 Volttia, joten siihen kytketty analoginen signaali voi olla jokin jännitetaso väliltä 0-5 Volttia. Alla oleva kuva esittää analogisen signaalin jännitetason vaihtelun ajan funktiona. X-akselilla on aika ja Y-akselilla jännite Voltteina.



Mikro-ohjain on kuitenkin digitaalinen laite, eikä se pysty suoraan käsittelemään analogista signaalia. Tämän vuoksi mikro-ohjain

muuntaa analogisen signaalin digitaaliseksi käyttäen ns. A/D-muunninta, joka on integroitu mikro-ohjaimen sisälle.

Arduino Unon A/D-muuntimen resoluutio (tarkkuus) on 10 bittiä ja kymmenellä bitillä pystytään esittämään luvut väliltä 0-1023. Niinpä Arduino Unon A/D-muuntimen palauttama lukuarvo on aina jokin luku väliltä 0-2013.

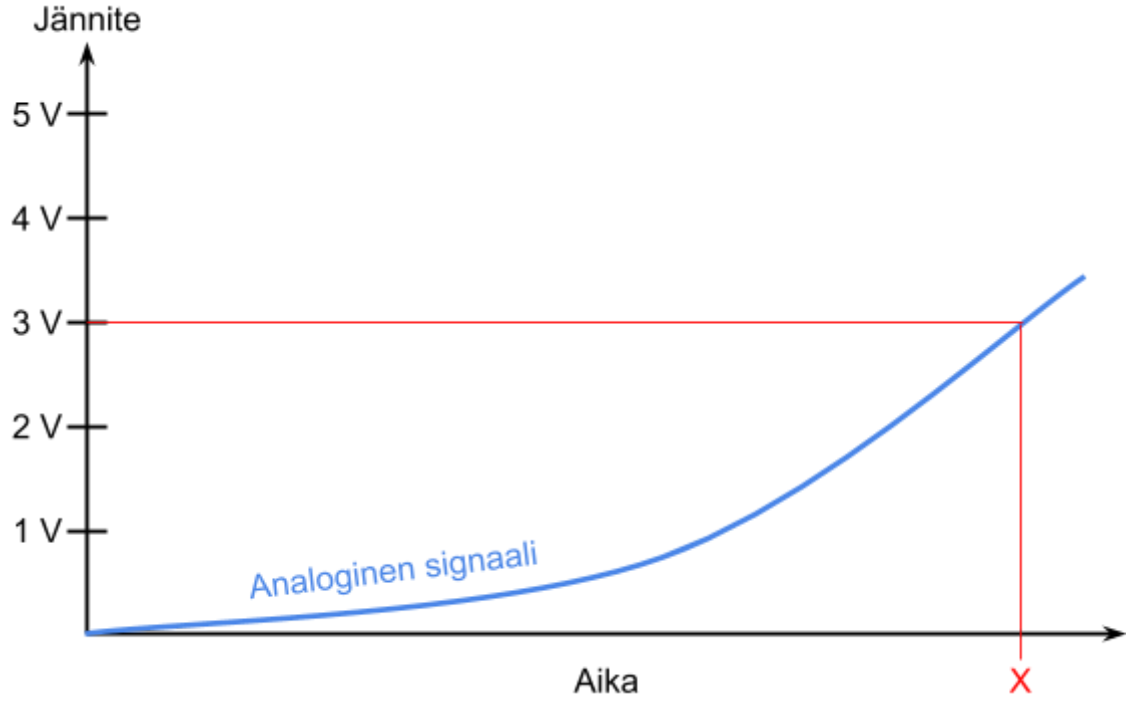
Ajan hetkellä X analogisen signaalin jännite on tasan 3 Volttia, minkä lukuarvon A/D-muunnin palauttaa tästä signaalista? Se voidaan laskea näin:

$$AD_{value} = U_{value} / U_{max} \times AD_{max}$$

Sijoitetaan arvot kaavaan ja lasketaan.
Tiedämme että jännite U_{value} = 3 Volttia, U_{max} = 5 volttia ja AD_{max} = 1023.

$$3\ V / 5\ V \times 1023 = 614$$

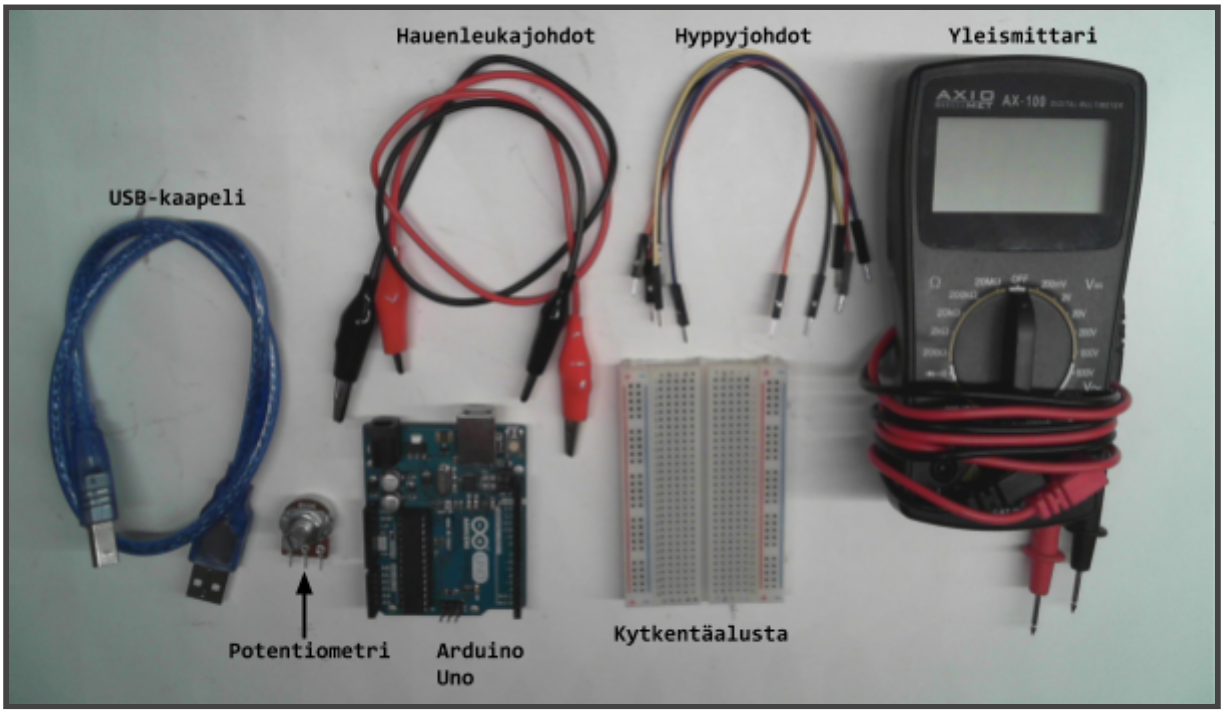
A/D-muuntimen palauttama lukuarvo on aika kokonaisluku, ei koskaan desimaaliluku. Siksi yllä oleva tulos on pyöristynyt arvoon 614.



Testataan tämä teoria käytännössä, työhön tarvitaan.

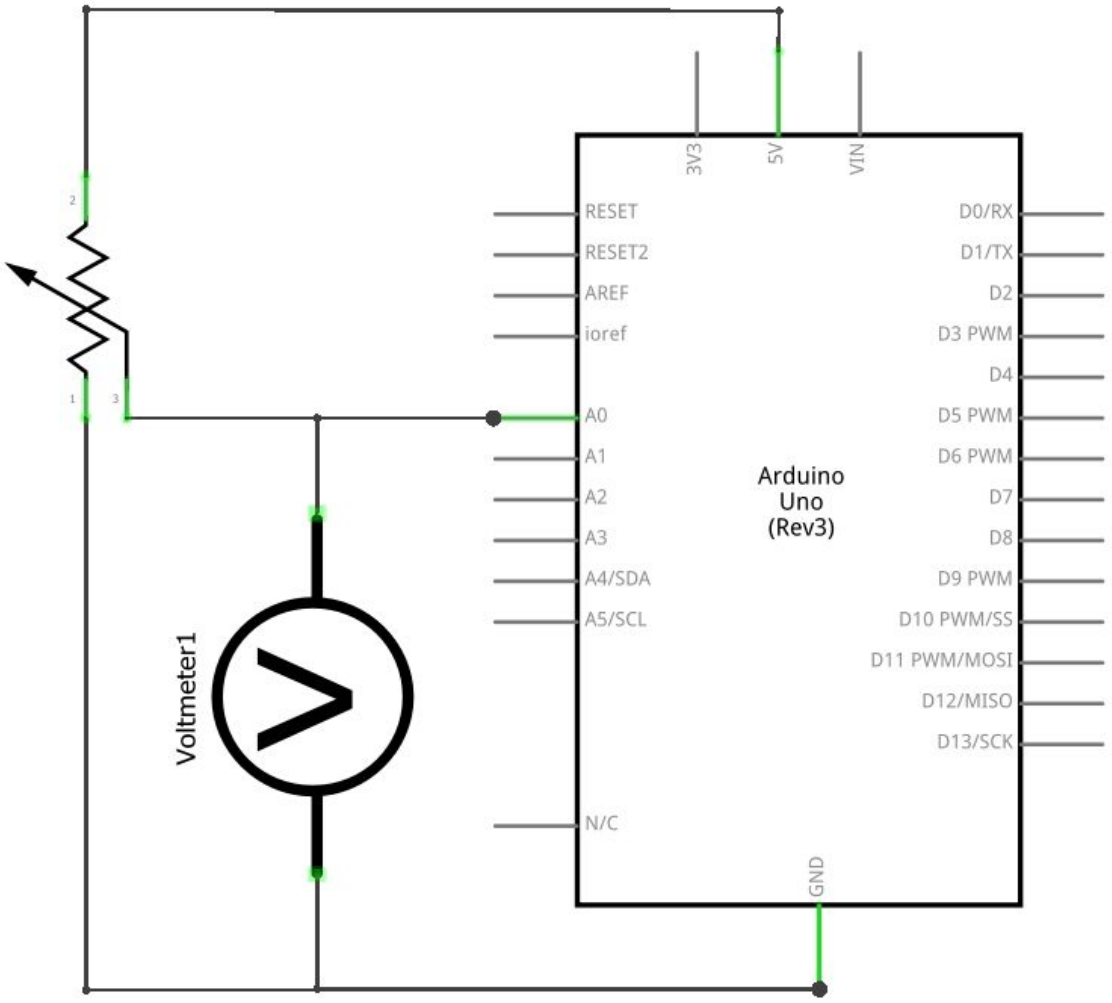
- Arduino Uno mikro-ohjain

- Unon USB-kaapeli
- Potentiometri 10 Kohm
- Viisi hyppyjohdinta (uros-uros):
    - punainen, sininen, keltainen, oranssi ja musta
- Yleismittari
- Kaksi hauenleukajohtoa: punainen ja musta
- Kytkentäalusta

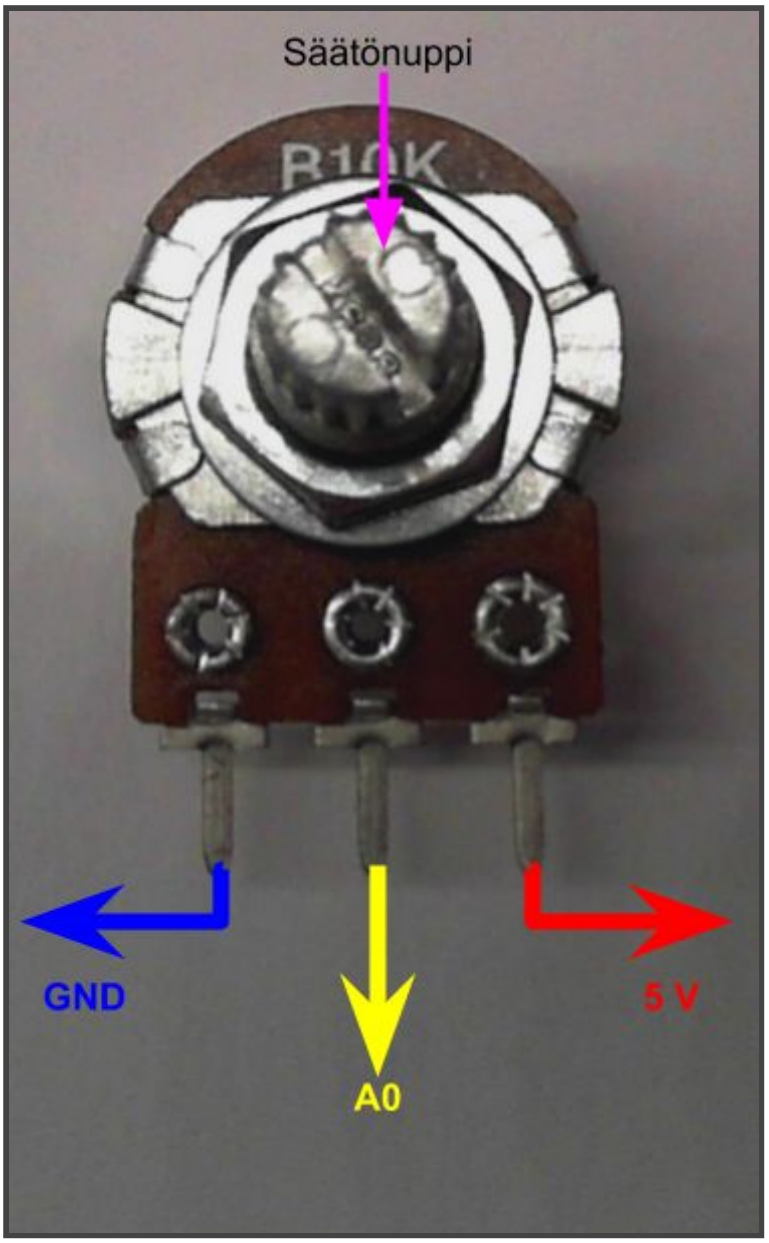


Kytkennän rakennus

Alla olevassa kuvassa on esitetty virtapiirin kytkentäkaavio.



Potentiometrissa on **kolme pinniä**. Oikealla oleva pinni on kytketty **5 Voltin** jännitteeseen ja vasen pinni **maahan (GND)**. Keskimmäinen pinni on kytketty mikro-ohjaimen analogiseen tuloon **A0**.

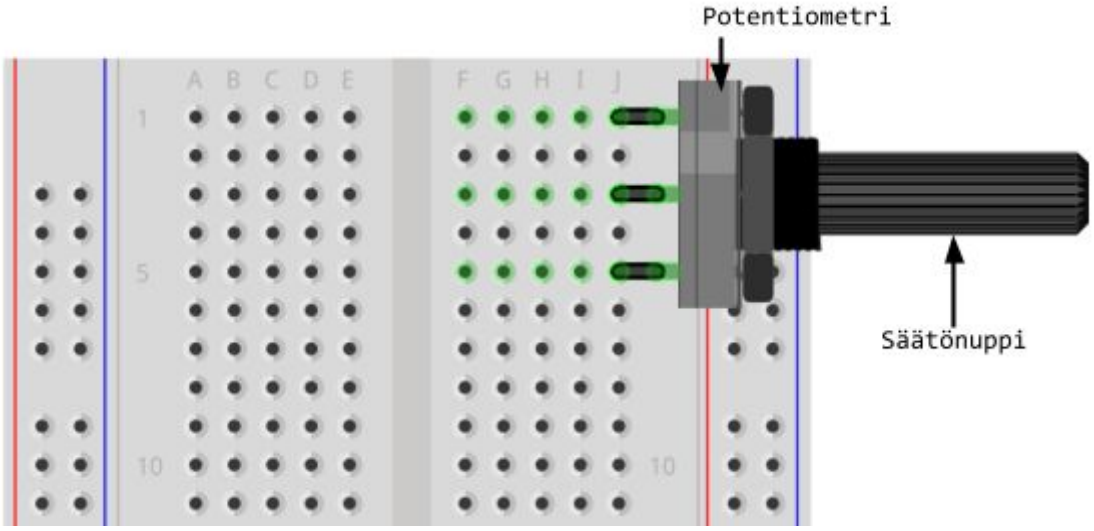


Virtapiirissä potentiometri toimii vastuksena, jonka resistanssi muuttuu kun käyttäjä pyörittää säätönuppia. Tämän potentiometrin resistanssi on 10 kiloOhmia. Voit mitata sen yleismittarilla valitsemalla mittarista resistanssin mittauksen ja kytkemällä mittajohdot potentiometrin äärimmäisiin pinneihin vasemmalla ja oikealla. Muuttuva resistanssi löytyy keskimmäisestä pinnistä. Jos mittajohdot on liitetty keskimmäiseen ja vasempaan tai oikeaan pinniin, niin havaitaan resistanssin muuttuvan välillä 0-10 kOhmia säätönuppia pyöritettäessä. Virtapiirissämme tämä muuttuva resistanssi tuottaa potentiometrin keskimmäiseen pinniin jännitteen, joka vaihtelee välillä 0-5 Volttia säätönuppia pyöritettäessä (=jännitteen jako kahdella vastuksella).

Voit kokeilla rakentaa virtapiirin suoraan kytkentäkaavion perusteella. Se onnistuu myös ohjatusti seuraavilla vaiheilla.

1.
Liitä **potentiometri** kytkentäalustalle näin:

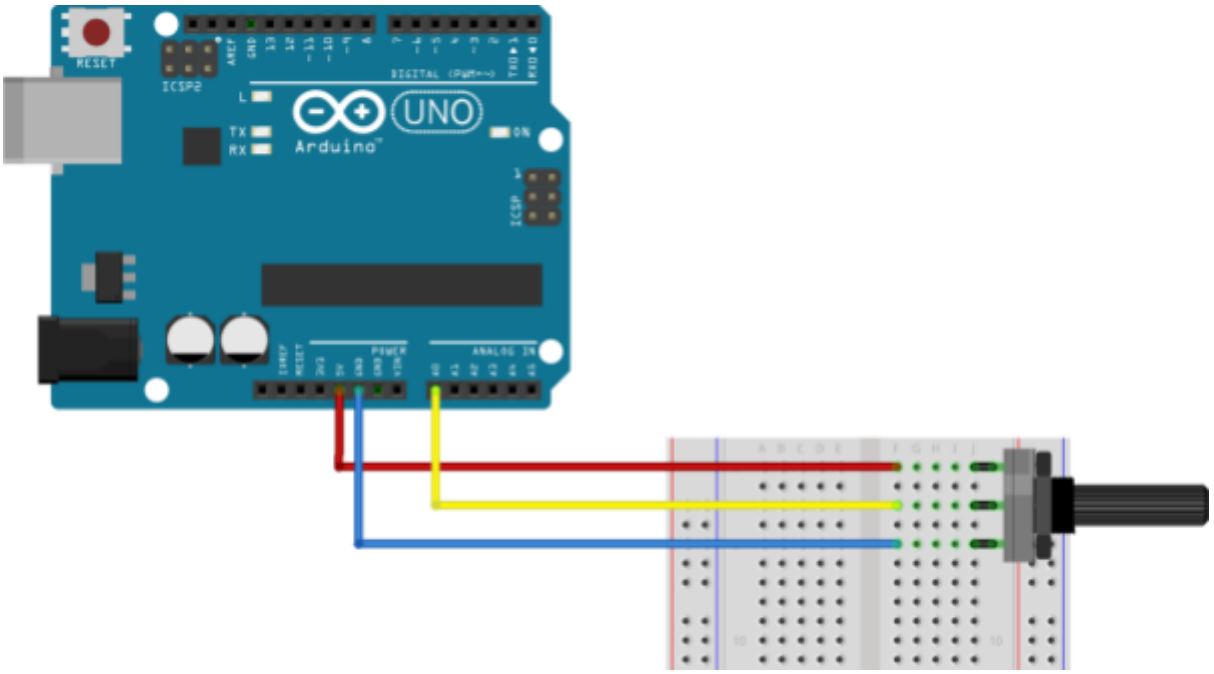
- Säätönuppi osoittaa oikealle
- Ylin jalka tulee **pisteeseen J1**
- Keskimmäinen jalka **pisteeseen J3**
- Alin jalka **pisteeseen J5**



2.
Kytke punainen hyppyjohto Arduinon **pinniin 5V** ja johdon toinen pää **pisteeseen F1**.

3.
Kytke keltainen hyppyjohto Arduinon **pinniin A0** ja johdon toinen pää **pisteeseen F3**.

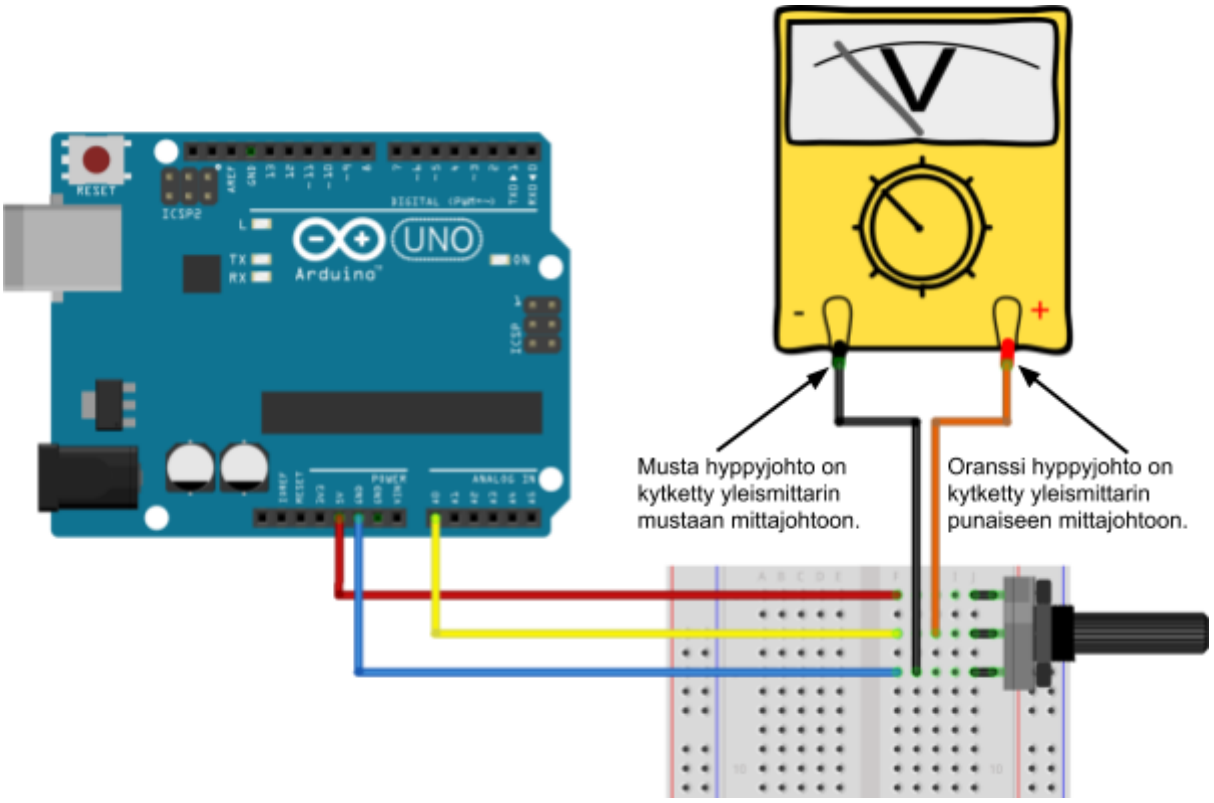
4.
Kytke sininen hyppyjohto Arduinon **pinniin GND** (johonkin kolmesta) ja johdon toinen pää **pisteeseen F5**. Potentiometri on nyt liitetty Arduinoon.



5.
Kytke oranssi hyppyjohto **pisteeseen H3**. Kytke johdon toinen pää punaisella hauenleukajohtimella kiinni **yleismittarin punaiseen mittajohtoon**.

6.
Kytke musta hyppyjohto **pisteeseen G5**. Kytke johdon toinen pää mustalla hauenleukajohtimella kiinni **yleismittarin mustaan mittajohtoon**. Kytkentä on nyt valmis.

Valmis kytkentä



Ohjelmointi

Aloitus
Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.

Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakio
Määritetään vakio **potPin**, jolla viittaamme koodissa **analogiseen tuloon A0**. Vakion nimi kertoo, että kyseiseen pinniin on kytketty potentiometri. Ohjelman toiminnan kannalta tämä ei ole tarpeen, mutta ohjelmoijalle se selventää koodin toimintaa.

```
//vakion määritys
#define potPin A0
```

Muuttuja
Esitellään kokonaislukutyyppinen (int) muuttuja **aValue**. Tallennamme siihen myöhemmin analogisen signaalin arvon

```
//muuttujan esittely
int aValue;
```

setup-funktio
Avataan sarjaliikenneyhteys tietokoneen ja Arduinon välille USB-kaapelin kautta funktiolla Serial.begin(9600).

```
void setup() {
  //sarjaliikenteen avaus
  Serial.begin(9600);
}
```

loop-funktio
Toistamme loop-funktiossa ikuisesti näitä kolmea tehtävää:

1. Luetaan analogisen signaalin arvo pinnistä **potPin** (A0) ja tallennetaan se muuttuja **aValue** arvoksi.
2. Lähetetään muuttujan **aValue** arvo tietokoneelle sarjaliikenne-yhteyden kautta.
3. Odotetaan 100 millisekuntia.

Arduinon analogiseen tuloon tuleva signaali luetaan funktiolla analogRead(<pinni>). Funktio palauttaa arvon väliltä **0 - 1023**. Luettu arvo vastaa pinniin tulevan jännitteen suuruutta.

```
void loop() {
  //luetaan analogisen signaalin arvo pinnistä potPin (A0)
  //ja tallennetaan se muuttujaan aValue
  aValue = analogRead(potPin);

  //lähetetään muuttujan aValue arvo tietokoneelle
  Serial.println(aValue);

  //odotetaan 100 millisekuntia
  delay(100);
}
```

Valmis koodi

```
//vakion määritys
#define potPin A0

//muuttujan esittely
int aValue;

void setup() {
  //sarjaliikenteen avaus
  Serial.begin(9600);
}

void loop() {
  //luetaan analogisen signaalin arvo pinnistä potPin (A0)
  //ja tallennetaan se muuttujaan aValue
  aValue = analogRead(potPin);

  //lähetetään muuttujan aValue arvo tietokoneelle
  Serial.println(aValue);

  //odotetaan 100 millisekuntia
  delay(100);
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

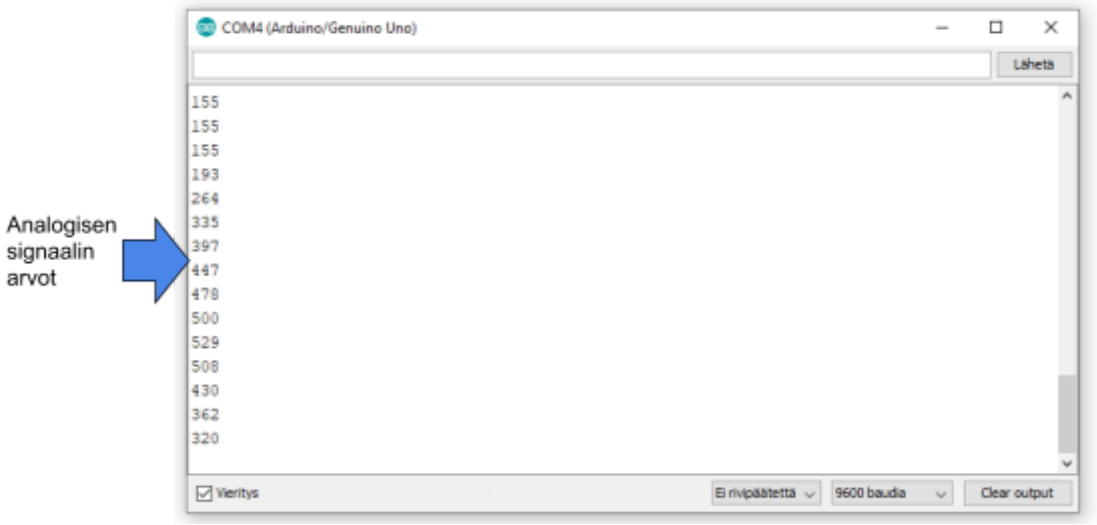
Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **“Lähetä”**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

Testaus

Avaamalla sarjamonitorin näet Arduinon tietokoneelle lähettämät analogisen signaalin arvot. Sarjamonitori avataan valikosta **Työkalut / Sarjamonitori**. Sarjamonitori aukeaa omaan ikkunaan. Pyöritä potentiometrin säätönuppia niin näet miten arvot muuttuvat mukana ja vaihtelevat välillä 0 -1023.



Sulje sarjamonitorin ikkuna ja avaa sarjaplotteri (**Työkalut / Serial Plotter**). Pyörittele säätönuppia ja näet graafisesti miten analogisen signaalin arvot muuttuvat ajan funktiona.

Jännite → Signaalin arvo

Olemme jo aiemmin oppineet, että analogisen signaalin arvo vastaa mikro-ohjaimen analogiseen tuloon tulevaa jännitettä. Kun jännite on 0 Volttia, signaalin arvo on myös nolla. Kun jännite on 5 Volttia, signaalin arvo on suurin mahdollinen, eli 1023. Kun jännite on 2,5 Volttia (puolet maksimijännitteestä), signaalin arvo on 512 (puolet maksimiarvosta).

Tarkistetaan tämä vielä yleismittarilla. Käännä yleismittarin kiertokytkin **tasajännitteen mittausalueelle kohtaan 20V** (mittaa enintään 20 Voltin tasajännitteen).

Säädä potentiometrin säätönupista yleismittarin näyttämä jännite alla olevan taulukon mukaisesti. Kun yleismittari näyttää haluttua jännitettä, niin tarkista sarjamonitorista analogisen signaalin arvo.

Yleismittarin näyttämä jännite Voltteina	Signaalin arvo sarjamonitorissa
0	0
2,5	512
5	1023

Kaikkien mittalaitteiden tuloksissa esiintyy aina pieni mittausvirhe. Jos työssäsi lukemat eivät ihan täysin vastaa esitettyjä, niin kyse voi olla mittalaitteiden mittausvirheistä.

Jännitteen laskenta

Ohjelmassa emme suoraan tiedä, minkä suuruinen jännite analogiseen tuloon tulee, tiedämme vain signaalin arvon väliltä 0-1023. Tästä arvosta voimme kuitenkin helposti laskea pinniin tulevan jännitteen kaavalla:

$$U_{in} = AnalogValue / 1023 \times 5\ V$$

Tallenna ohjelma toisella nimellä. Käytetään koodia pohjana ja tehdään siihen jännitteen laskentaan tarvittavat muutokset.

Esitellään uusi muuttuja **uIn**. Muuttujan tyyppi on float ja tämän tyyppiseen muuttujaan voidaan tallentaa desimaaliluku.

```
float uIn; //tähän lasketaan jännitteen arvo
```

Lasketaan analogiseen tuloon tulevan jännitteen suuruus käyttäen aiempaa kaavaa ja muuttujan aValue arvoa. Kaavassa **muuttuja aValue** muunnetaan **float-tyyppiseksi** tyyppimuunnoksella `(float)aValue`, jotta laskutoimitus tuottaa halutunlaisen desimaaliluvun. Sitten vain lähetetään tietokoneelle laskettu jännite muuttujasta **uIn**.

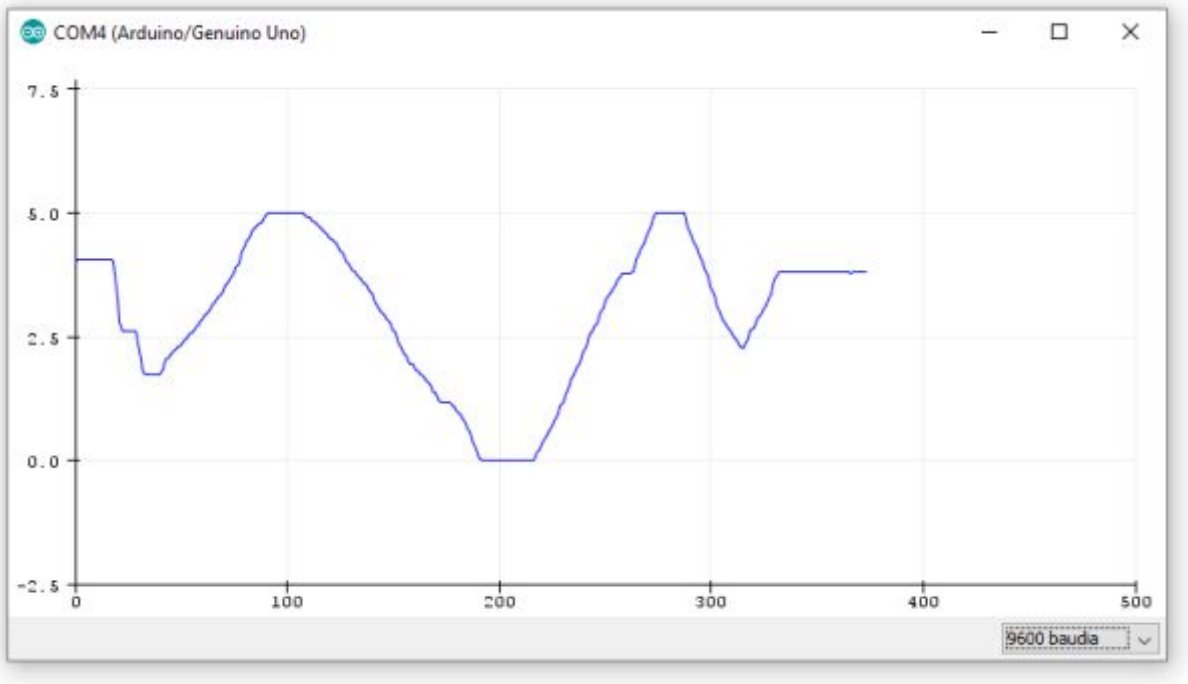
```
 //jännitteen laskenta
  uIn = ((float)aValue / 1023) * 5;

  //lähetetään laskettu jännite tietokoneelle
  Serial.println(uIn);
```

Koko ohjelman muutettu koodi:

```
//vakion määritys
#define potPin A0

//muuttujien esittely
int aValue;
float uIn; //tähän lasketaan jännitteen arvo

void setup() {
  //sarjaliikenteen avaus
  Serial.begin(9600);
}

void loop() {
 //luetaan analogisen signaalin arvo pinnistä potPin (A0)
  //ja tallennetaan se muuttujaan aValue
  aValue = analogRead(potPin);

  //jännitteen laskenta
  uIn = ((float)aValue / 1023) * 5;

  //lähetetään laskettu jännite tietokoneelle
  Serial.println(uIn);

  //odotetaan 100 millisekuntia
  delay(100);
}
```

Käännä ohjelma ja lähetä se Arduinoon. Avaa sarjamonitori ja pyöritä potentiometrin säätönuppia. Vertaa sarjamonitorissa näkyviä arvoja yleismittarin näytöllä oleviin jännitearvoihin. Niiden tulisi olla hyvin lähellä toisiaan.

Ja huomasitko, rakensit juuri **tietokoneeseen liitettävän jännitemittarin**. Se mittaa ja laskee jännitteen sekä lähettää jännitteen arvon tietokoneelle. Tällä mittarilla voimme tosin mitata vain jännitteitä alueelta 0-5 Volttia, mutta mikro-ohjaimien ja niiden antureiden kanssa tämä on aivan riittävä mittausalue.



Automaattinen kytkin valovastuksella

Työn kuvaus

Työssä rakennetaan ja ohjelmoidaan laite, joka kytkee automaattisesti LEDin päälle ja sammuttaa sen ympäristön valoisuuden mukaan. Hämärässä LEDi syttyy ja valoisassa ympäristössä se sammuu. Ympäristön valon määrän tutkimiseen käytetään valovastusta (LDR, Light Dependent Resistor).

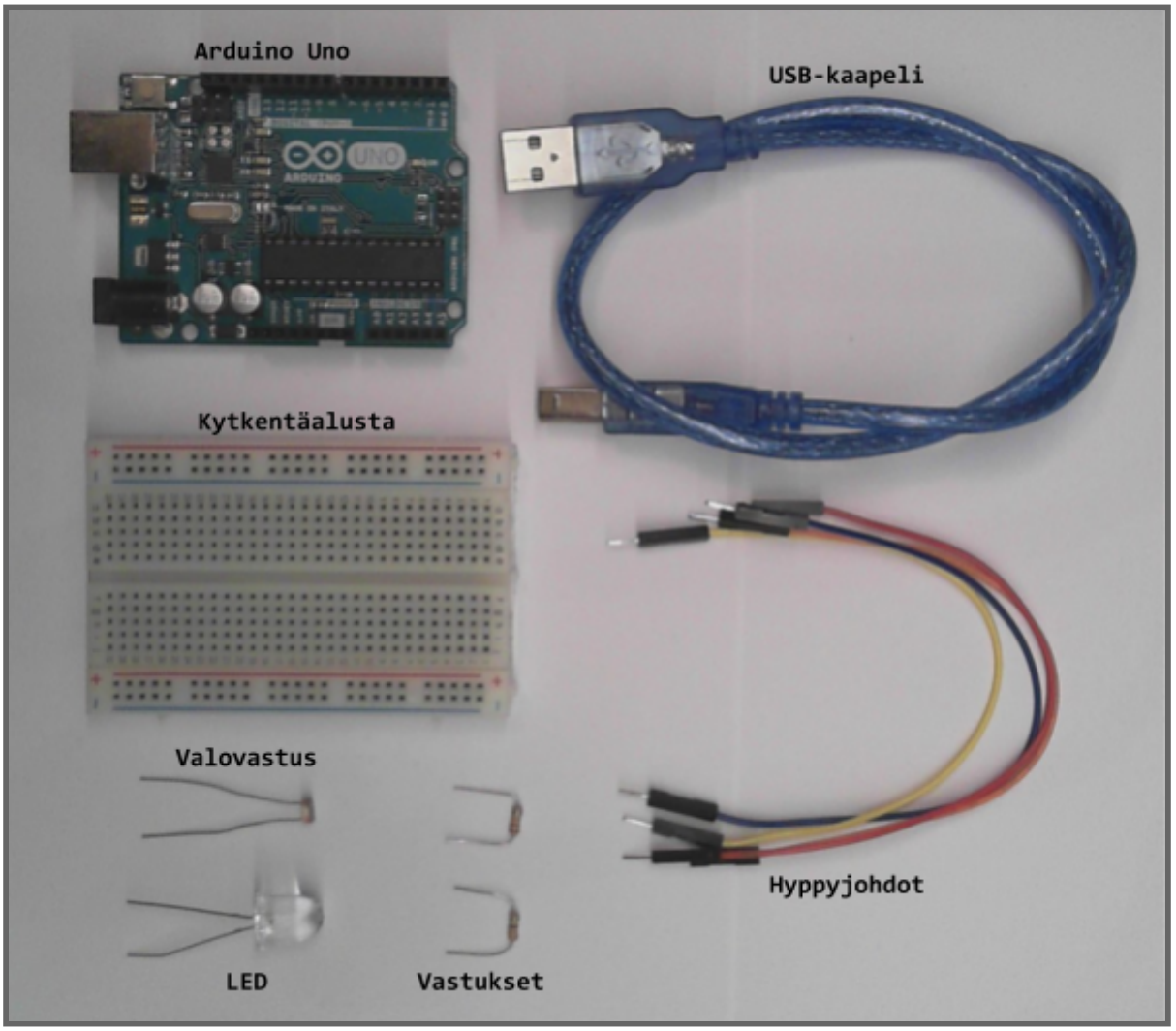
Työssä harjoitellaan automaation perusperiaatetta:

- Mitataan ympäristöstä haluttu suure anturia käyttäen.
- Tutkitaan mittaustulos ja tehdään tarvittaessa ohjaustoimenpide mittaustuloksen perusteella.

Oikeassa automaatiossa suoritettava toimenpide on yleensä sellainen, että se vaikuttaa jatkossa tehtäviin mittaustuloksiin. Esimerkiksi jos mitattu lämpötila ylittää raja-arvon, käynnistetään tuuletin jonka toiminta aiheuttaa lämpötilan laskun. Kun lämpötilan on pudonnut tarpeeksi, tuuletin pysäytetään.

Tarvittavat komponentit

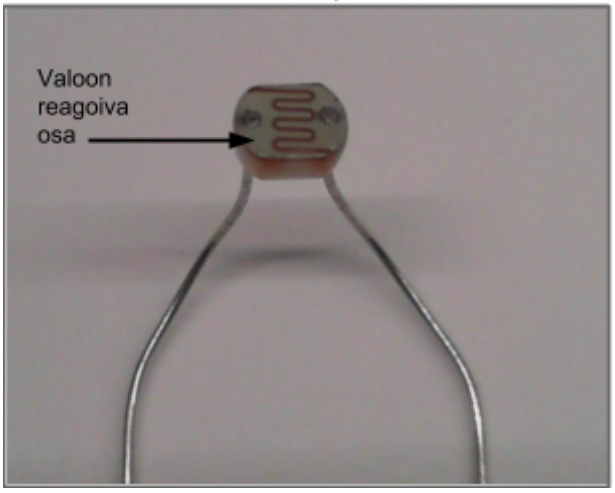
- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- Valovastus
- Kirkas punainen LED
- Kaksi vastusta: 330 Ohmia ja 22 Kohmia
- Kytkentäalusta
- 4 hyppyjohtoa (uros-uros): punainen, sininen, oranssi ja keltainen



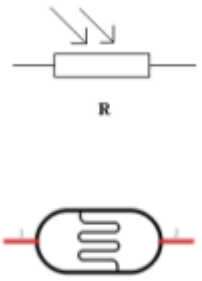
Valovastus

Valovastus on nimensä mukaisesti vastus, mutta sen resistanssi muuttuu siihen osuvan valon voimakkuuden mukaisesti. Kirkkaassa valossa valovastuksen resistanssi pienenee ja hämärässä se kasvaa.

Valovastus komponenttina

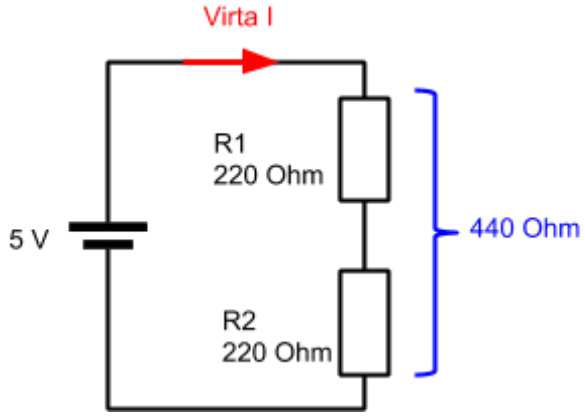


Valovastuksen piirrosmerkkejä



Kahdella sarjaan kytketyllä vastuksella voidaan toteuttaa jännitteen jako. Katsotaan ensin esimerkki, jossa kaksi yhtä suurta vastusta. Piirissä on kytketty kaksi 220 Ohmin vastusta sarjaan. Vastusten sarjakytkennän kokonaisresistanssi on yksittäisten resistanssien summa, eli laskemme kaikki resistanssit yhteen.

$$R_{tot} = R_1 + R_2 \rightarrow R_{tot} = 220\,\Omega + 220\,\Omega = 440\,\Omega$$



Virtapiiriin kytketty jännite on 5 Volttia, minkä suuruinen virta piirissä kulkee? Voimme laskea virran Ohmin lailla, joka liittää yhteen jännitteen, virran ja resistanssin. Ohmin laki on helppo muistaa muistikolmiosta. Peitä kolmiosta suure, jonka haluat laskea niin näet sen laskentakaavan.

Nyt haluamme laskea virran I, joten peitämme sen. Kolmio kertoo, että virta I lasketaan jakamalla jännite U resistanssilla R.

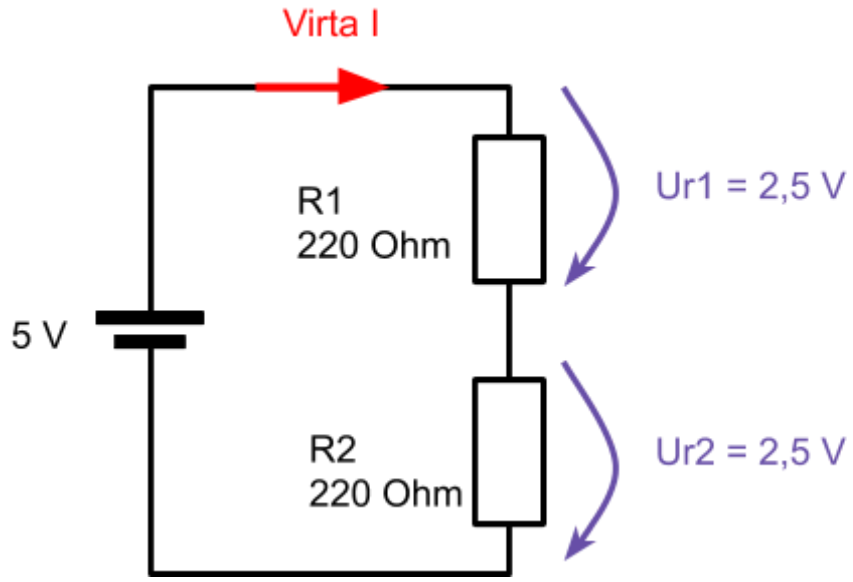
$$I = U / R \rightarrow I = 5V / 440\Omega = 11,4\, mA$$

Seuraavaksi kysymme, kuinka suuri jännite vaikuttaa vastusten R1 ja R2 yli? Senkin voimme laskea Ohmin lailla. Tiedämme, että piirissä kulkeva virta on 11,4 mA ja tiedämme vastusten resistanssit.

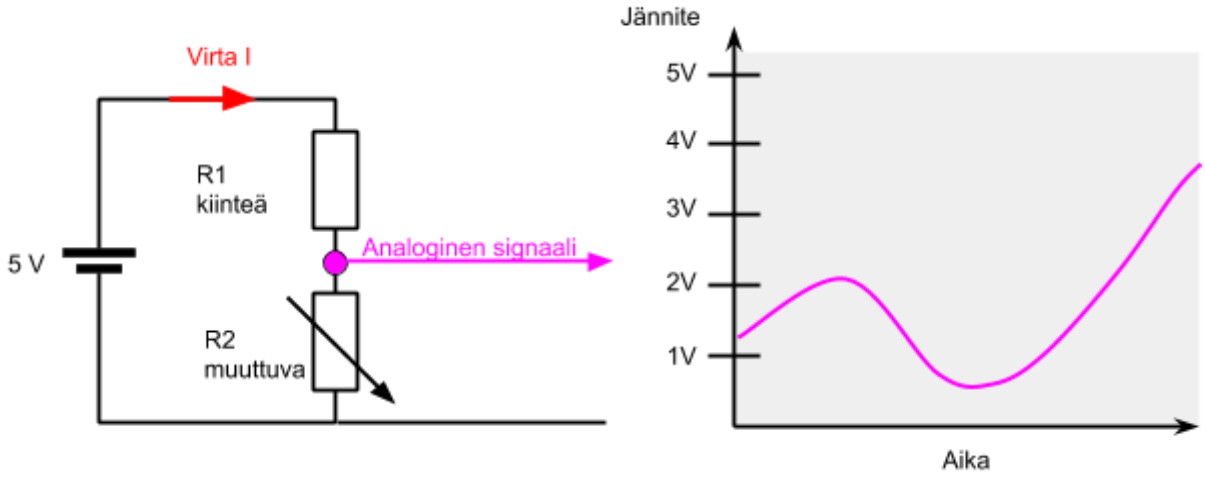
$$U_{R1} = R_1 \times I = 220\Omega \times 0,0114\, A = 2,5\, V$$

$$U_{R2} = R_2 \times I = 220\Omega \times 0,0114\, A = 2,5\, V$$

Koska vastusten R1 ja R2 resistanssit olivat yhtä suuret, niin ovat myös niiden yli vaikuttavat jännitteet yhtä suuret.



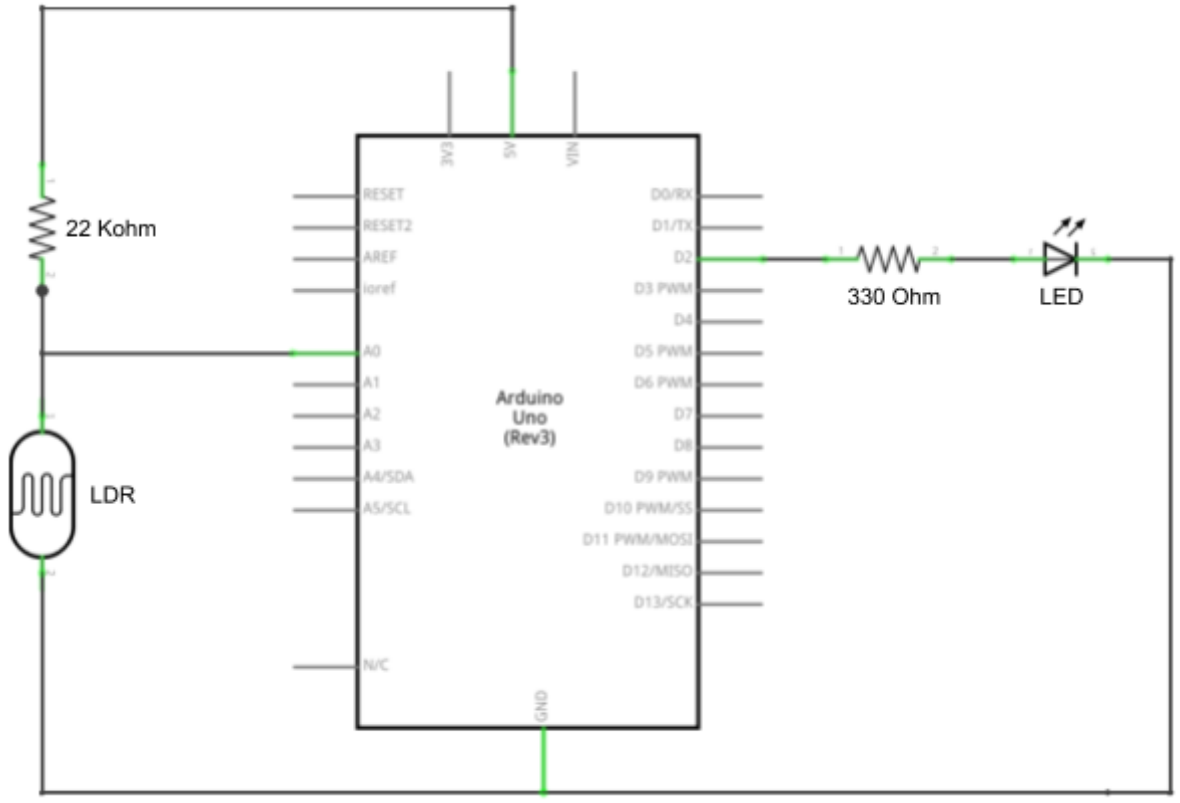
Tilanteessa, jossa **vastuksen R1 resistanssi on kiinteä** ja **vastuksen R2 resistanssi on muuttuva**, saamme **pisteeseen A analogisen signaalin**, jonka jännite muuttuu vastuksen R2 resistanssin mukana.



Juuri näin kytkemme valovastuksen virtapiiriimme. Vastus R1 on kiinteä ja **vastus R2 on valovastus**. Niiden välistä saamme analogisen signaalin, jonka kytkemme Arduinon **analogiseen tuloon A0**.

Kytkennän rakennus

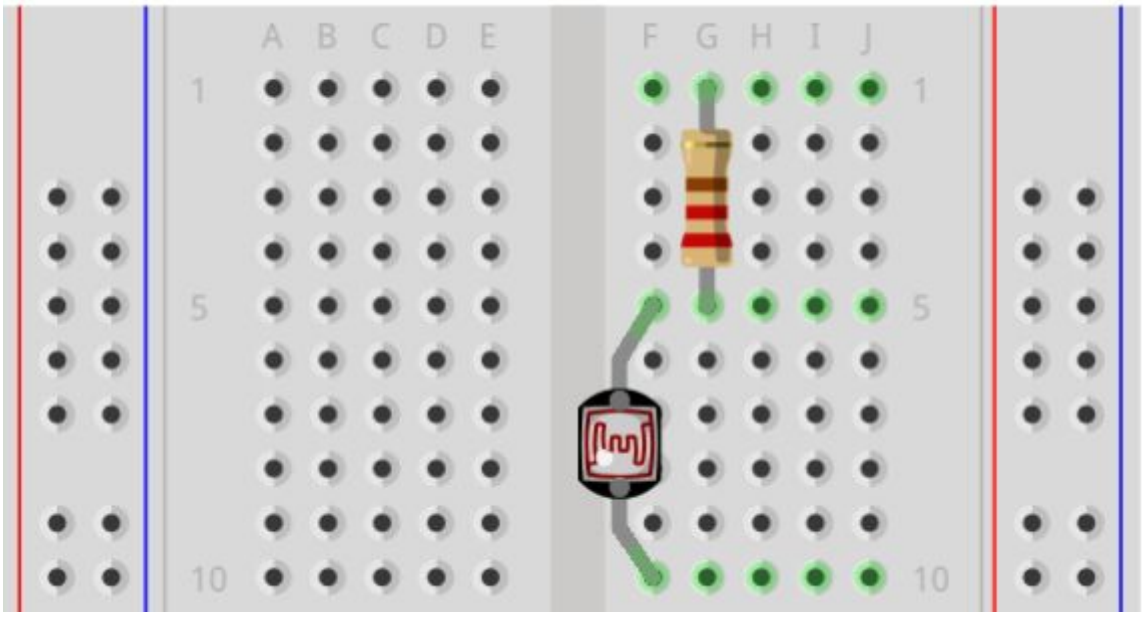
Valovastuskytkennän lisäksi käytämme virtapiirissä myös sinulle jo tuttua LED-kytkentää. Alla on esitetty virtapiirin kytkentäkaavio.



Koeta rakentaa virtapiirin suoraan kytkentäkaavion perusteella. Jos se osoittautuu haastavaksi, voit tehdä sen myös seuraavilla ohjeilla.

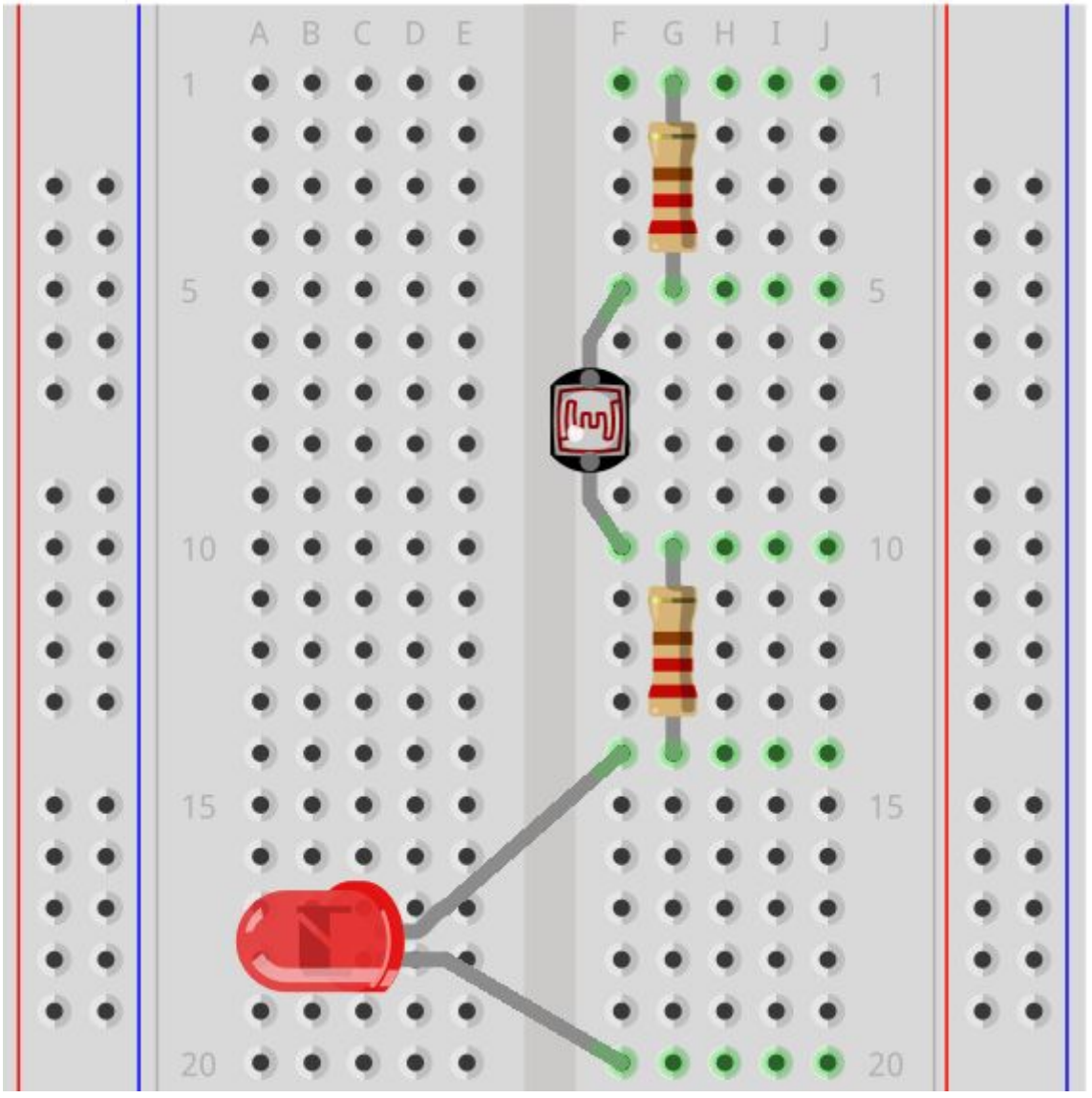
1.
Kytke 22 Kohmin vastus pisteisiin **G1** ja **G5**.

2.
Kytke LDR-vastus pisteisiin **F5** ja **F10.**



3.
Kytke 330 Ohmin vastus pisteisiin **G10** ja **G14**.

4.
Kytke LEDin **pitempi positiivinen jalka** pisteeseen **F20** ja lyhempi jalka pisteeseen **F14**..



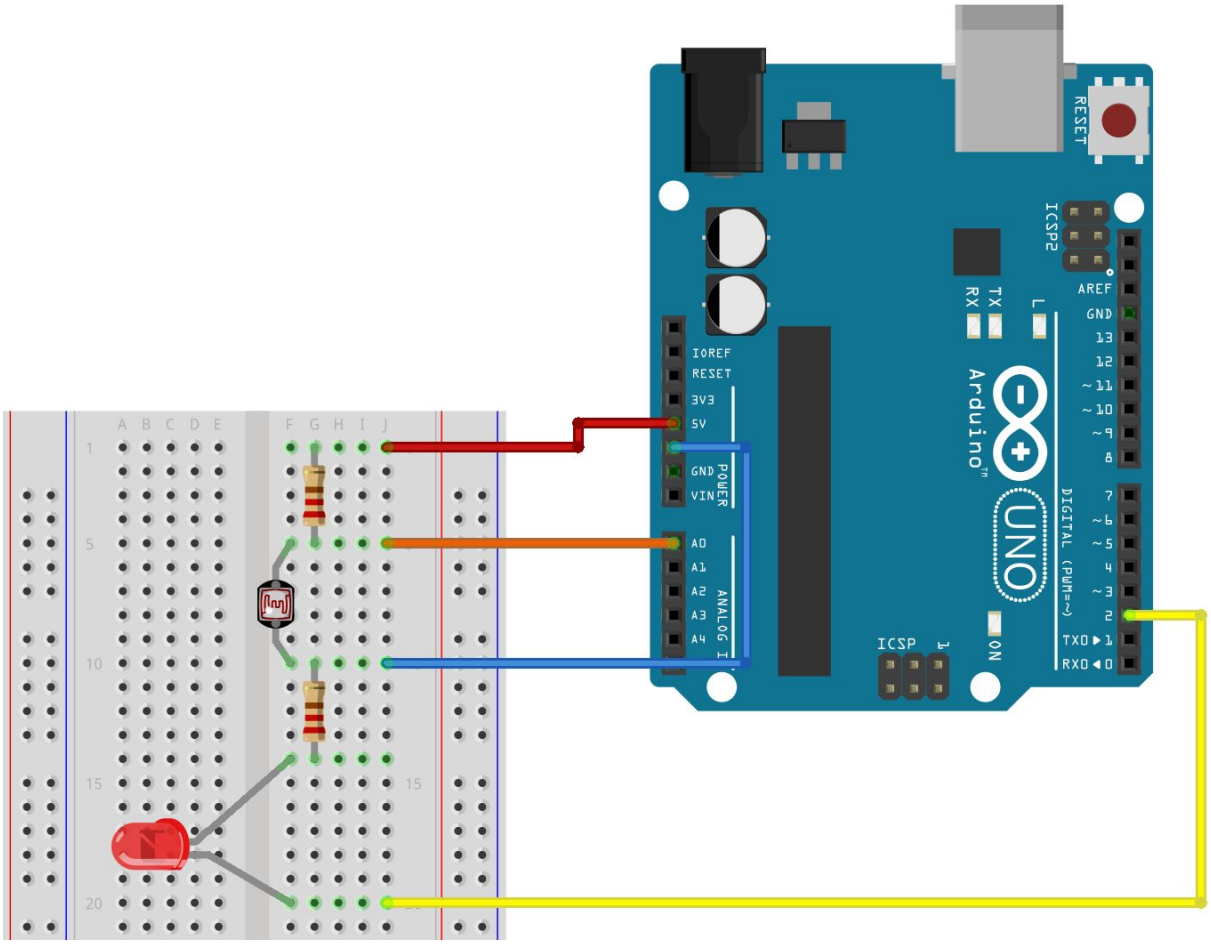
5.
Kytke punainen hyppyjohto Arduinon **pinniin 5V** ja johdon toinen pää pisteeseen **J1**. Tätä johtoa pitkin kulkee jännite vastuksilla muodostettuun jännitteenjako-kytkentään.

6.
Kytke oranssi hyppyjohto Arduinon **pinniin A0** ja johdon toinen pää pisteeseen **J5**. Tätä johtoa pitkin kulkee analoginen signaali jännitteenjako-kytkennästä Arduinon analogiseen tuloon.

7. Kytke sininen hyppyjohto Arduinon **pinniin GND** ja johdon toinen pää pisteeseen **J10**. Tämä toimii maa-johtona jännitteenjako-kytkennälle sekä LEDille.

8. Kytke keltainen hyppyjohto Arduinon **pinniin D2** ja johdon toinen pää pisteeseen **J20**. Tätä johtoa pitkin kulkee LEDin ohjaussignaali.

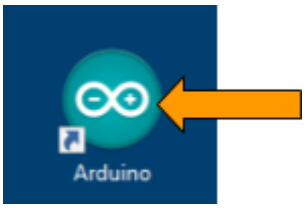
Kytkentä on nyt valmis.



Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.



Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakiot

Määritetään kaksi vakioita. Ensimmäisellä viittaamme analogiseen tuloon A0 (ldrPin) ja toisella digitaaliseen lähtöön D2 (ledPin).

```
//vakioiden määritys
#define ldrPin A0
#define ledPin 2
```

Muuttujat

Analogisen signaalin arvon tallentamista varten tehdään yksi kokonaislukutyyppinen muuttuja (int).

```
//muuttujan esittely
int aValue;
```

setup-funktio

Määritetään pinni ledPin (D2) toimimaan digitaalisena lähtönä.

```
void setup() {
  pinMode(ledPin, OUTPUT);
}
```

loop-funktio

Ensimmäiseksi luetaan funktiolla analogRead(ldrPin) jännitteenjako-kytkennästä tulevan analogisen signaalin arvo ja tallennetaan se muuttujaan **aValue**.

```
aValue = analogRead(ldrPin);
```

Sitten tutkitaan ehtorakennetta käyttäen, onko muuttujan aValue arvo suurempi kuin 700 vai pienempi. Jos arvo on suurempi kuin 700 (hämärää), niin sytytetään LED funktiolla digitalWrite(ledPin, HIGH). Muuten sammutetaan LED (valoista).

```
if (aValue > 700)
   digitalWrite(ledPin, HIGH);
 else
   digitalWrite(ledPin, LOW);
```

Valmis koodi

```
//vakioiden määritys
#define ldrPin A0
#define ledPin 2

//muuttujan esittely
int aValue;

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

void loop() {
  //luetaan analogisen signaalin arvo pinnistä ldrPin (A0)
  //ja tallennetaan se muuttujaan aValue
  aValue = analogRead(ldrPin);

  //jos analogisen signaalin arvo > 700,
  //niin sytytetään LED, muuten sammutetaan LED
  if (aValue > 700)
    digitalWrite(ledPin, HIGH);
  else
    digitalWrite(ledPin, LOW);
}
```

Ohjelma on valmis, lataa se Arduinoon.

Testaus

Liitä Arduino USB-kaapelilla tietokoneeseen ja lataa ohjelma Arduinoon. Normaalissa huoneenvalossa LEDin ei tulisi palaa. Peitä valovastus kädellä tai sammuta huoneesta valot. Mikro-ohjaimen tulisi nyt ohjata LEDi päälle. Poista käsi valovastuksen päältä tai kytke valot, LEDin tulisi nyt sammua.

Ympäristön valoisuudesta tai valovastuksen ominaisuuksista johtuen voit joutua muuttamaan koodissa käytettyä raja-arvoa 700. Saadaksesi selville sopivan raja-arvon voit lisätä ohjelmaan testausta varten toiminnon, joka lähettää signaalin arvon tietokoneesi sarjamonitoriin.

Ota sarjaliikenne käyttöön **setup**-funktiossa:

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

Lisää analogisen signaalin arvon lähetys ja viive **loop**-funktioon:

```
  Serial.println(aValue);
  delay(500);
```

Käännä ja lähetä ohjelma Arduinoon. Avaa sarjamonitori ja katso, mitä arvoja Arduino lähettää valoissa ja hämärässä. Valitse raja-arvosi sen mukaan. Kirjoita haluamasi raja-arvo luvun 700 tilalle. Tämän jälkeen voit muuttuu äsken lisäämäsi koodirivit kommenteiksi, kääntää ja lähettää ohjelman uudelleen Arduinoon. Testaa toiminta uudella raja-arvolla.

Kommentiksi muutettu koodi:

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

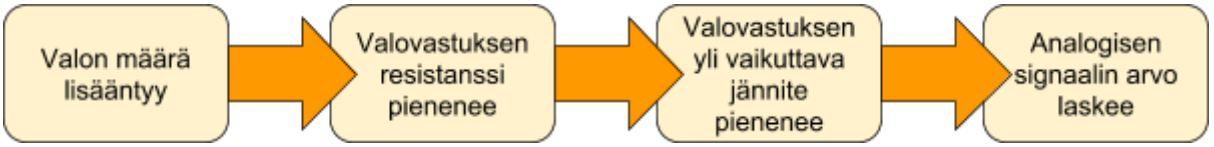
ja

```
  //Serial.println(aValue);
  //delay(500);
```

Kirkkaassa valossa valovastuksen resistanssi pienenee ja samalla pienenee sen yli mitattava jännite, sillä

$$U = R \times I$$

Valovastuksen yli mitattava jännite on Arduinon lukema analoginen signaali. Valon määrän lisääntyessä tapahtuu kuvan mukainen tapahtumaketju.



Valon määrän vähentyessä tapahtuu päinvastoin, ja silloin tapahtumaketjun lopuksi analogisen signaalin arvo kasvaa.

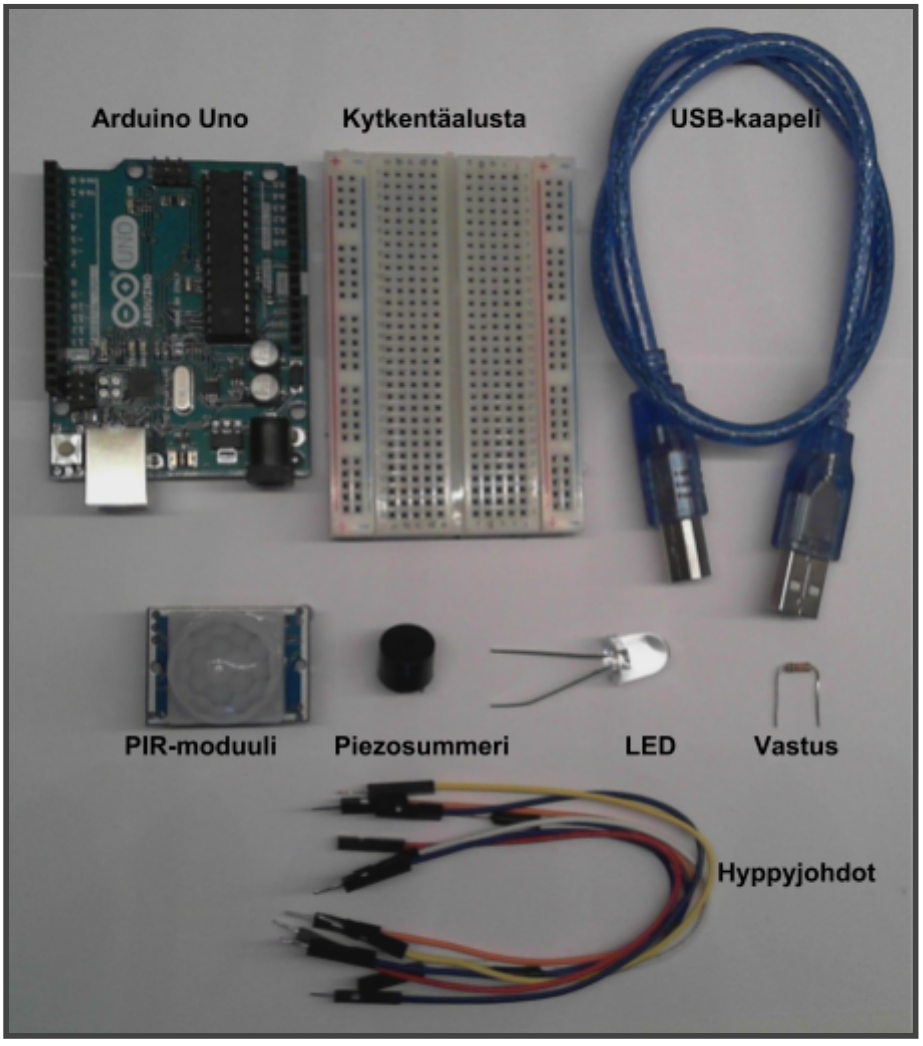
Varashälytin

Työn kuvaus

Työssa rakennetaan ja ohjelmoidaan varashälytin. Liiketunnistimena käytetään PIR-moduulia ja hälytys tehdään vilkkuvalla LEDillä sekä piippaavalla piezosummerilla.

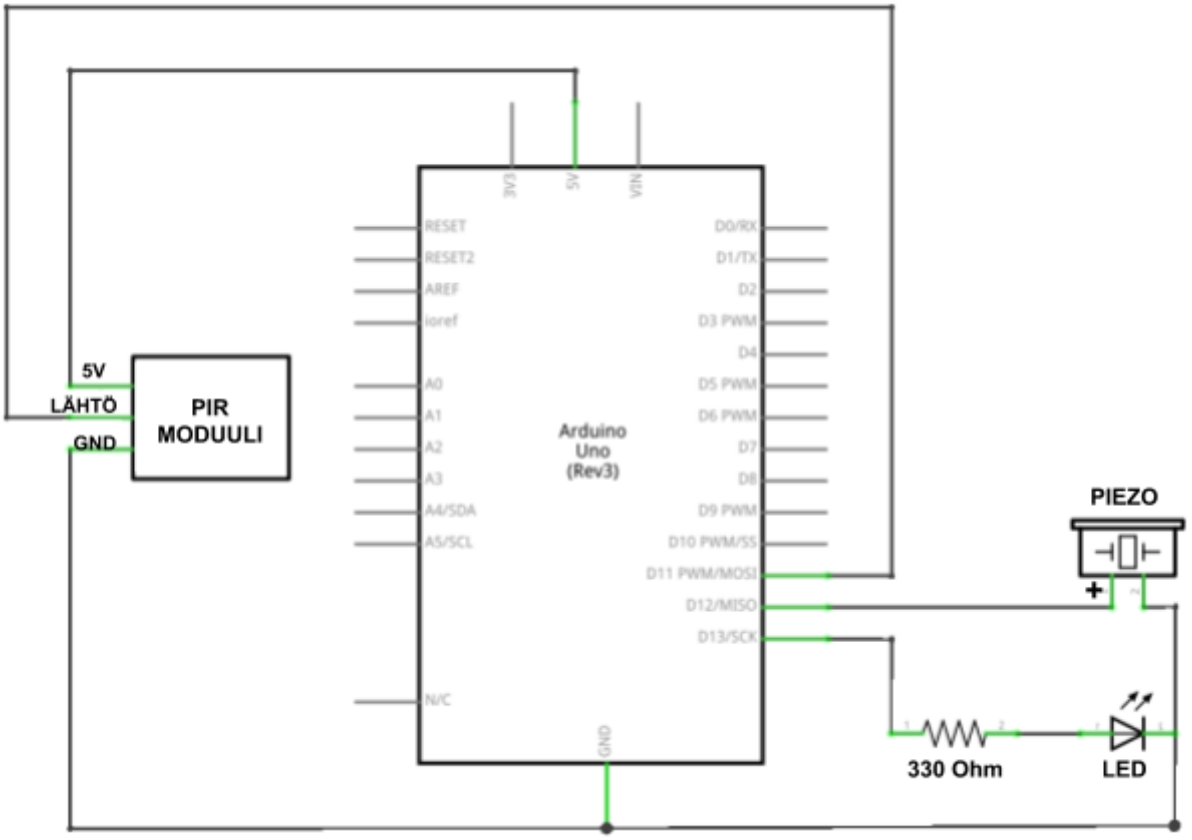
Tarvittavat komponentit

- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- Kirkas punainen LEDi
- 330 Ohmin vastus
- Piezosummeri
- PIR-moduuli (liiketunnistin)
- 3 hyppyjohdinta naaras-uros -liittimillä
    - 1 punainen, 1 keltainen ja 1 sininen
- 2 hyppyjohtoa uros-uros -liittimillä
    - 1 valkoinen, 1 oranssi ja 2 sinistä
- Kytkentäalusta



Kytkennän rakennus

Alla olevassa kuvassa on esitetty virtapiirin kytkentäkaavio.

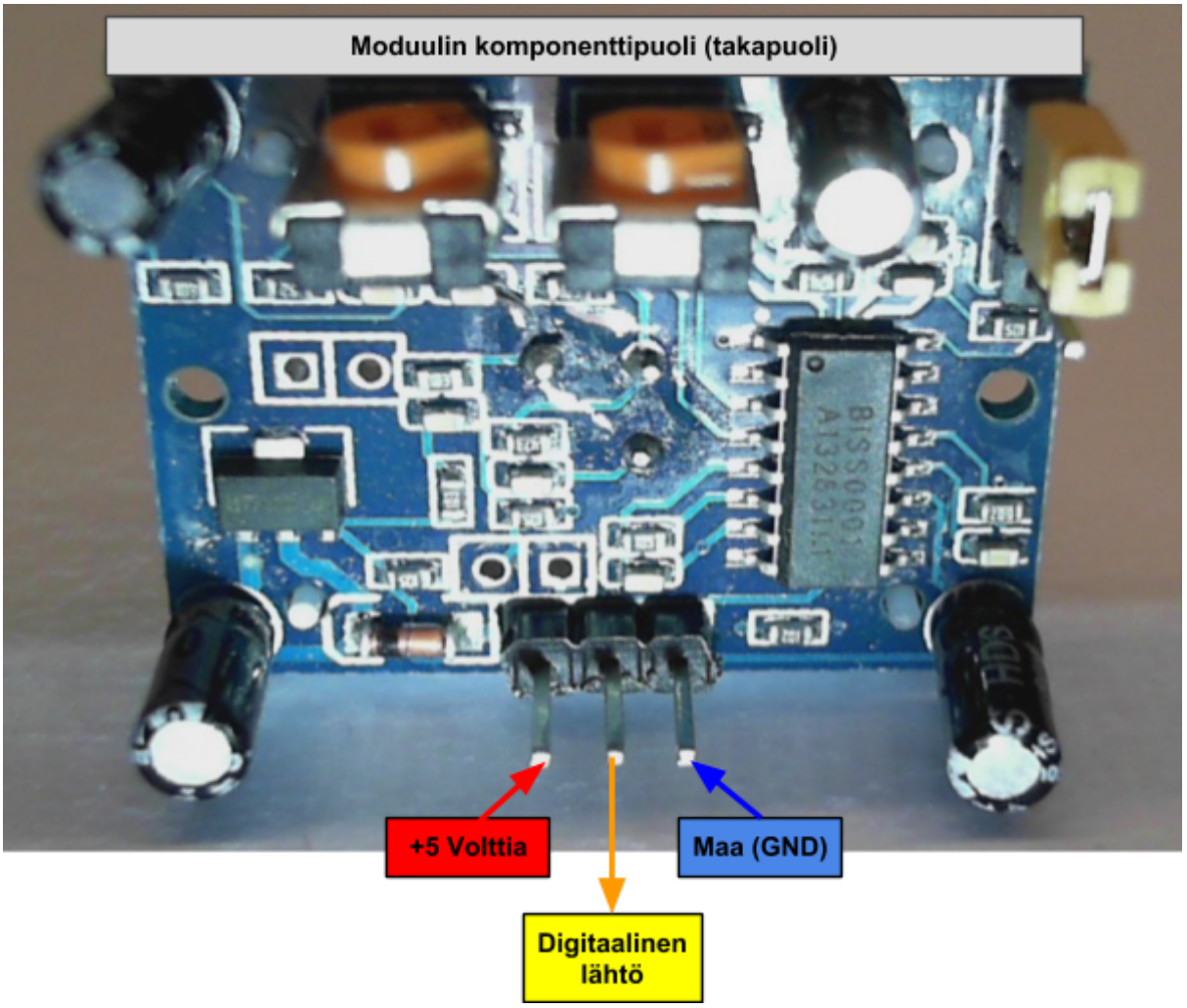


PIR-moduuli

Kytkennässä käytetään liiketunnistimena PIR-moduulia. Moduulissa on kolme pinniä:

- 5 Voltin käyttöjännite
- Digitaalinen lähtö
- Maa (GND)

Urospinnit sijaitsevat moduulin takana ja niihin liitetään hyppyjohdon **naarasliitin**. Käännä moduulin komponenttipuoli (takaosa) itseäsi kohti niin, että urospinnit ovat moduulin alareunassa.



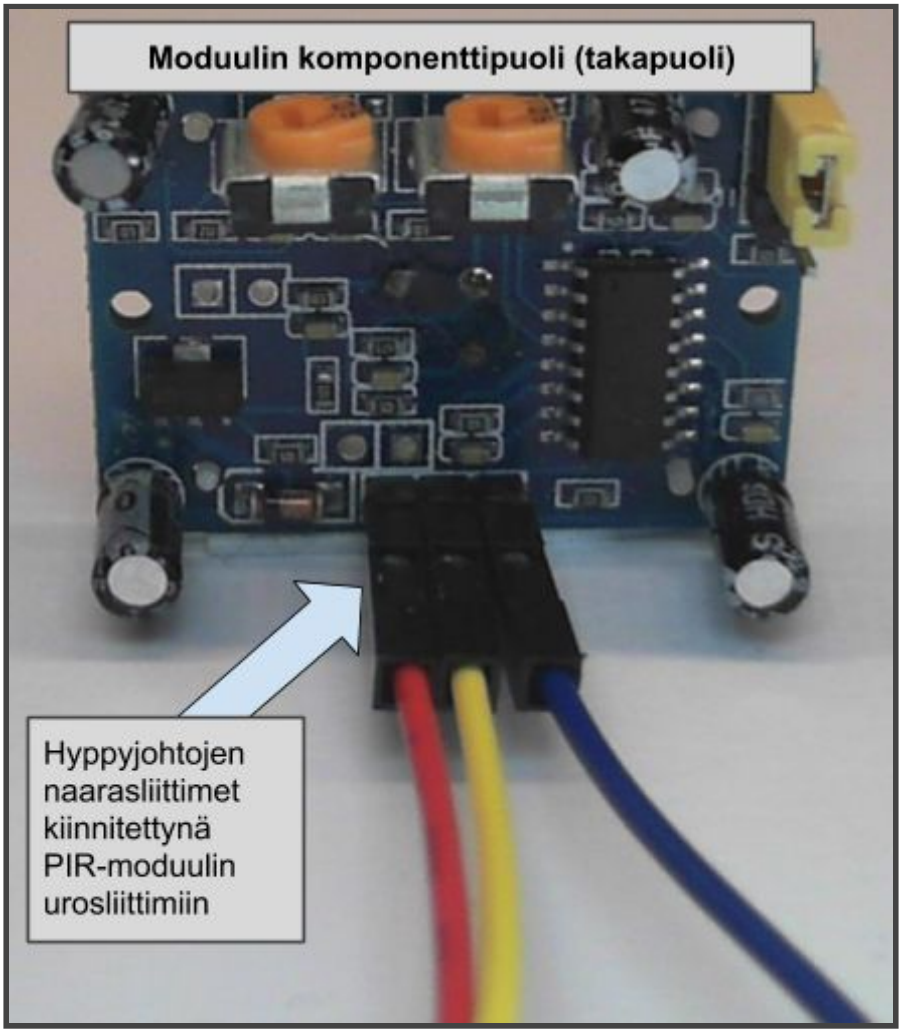
1. PIR-moduulin kytkentä

Liitä hyppyjohdot PIR-moduulin ja Arduinon väliin näin:

- Punaisen hyppyjohdon naarasliitin **PIR-moduulin pinniin +5 Volttia** ja johdon toisessa päässä oleva urosliitin **Arduinon pinniin 5 V**.
    - Tätä johtoa pitkin Arduinosta tulee käyttöjännite moduuliin.

- Keltaisen hyppyjohdon naarasliitin **PIR-moduulin "Digitaalinen lähtö"-pinniin** ja johdon toisessa päässä oleva urosliitin **Arduinon pinniin D11**.
    - Tätä johtoa pitkin PIR-moduuli ilmoittaa Arduinolla havaitsemastaan liikkeestä digitaalisella signaalilla.

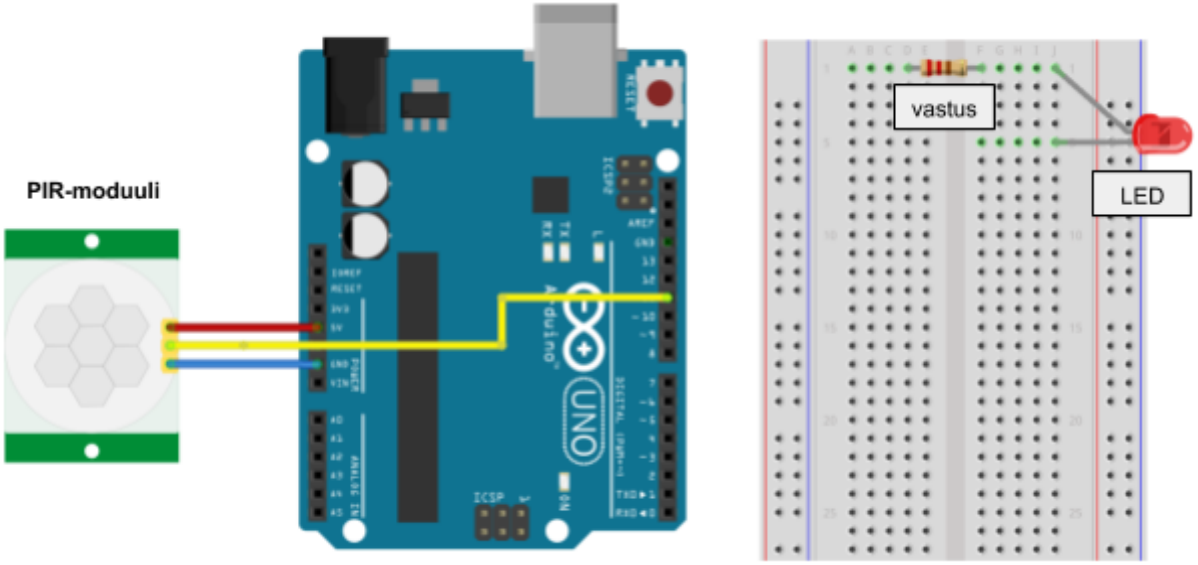
- Sinisen hyppyjohdon naarasliitin **PIR-moduulin pinniin Maa (GND)** ja johdon toisessa päässä oleva urosliitin **Arduinon pinniin GND**.
    - Tämä johto on käyttöjännitteen ja signaalin yhteinen maajohto.



Jatkossa käytämme hyppyjohtoja, joiden molemmissa päissä on urosliittimet.

2.
Kytke 330 Ohmin vastus kytkentäalustalle pisteisiin **D1** ja **F1.**

3.
Kytke LEDin **pidempi positiivinen jalka pisteeseen J1** ja lyhyempi negatiivinen jalka pisteeseen **J5**.



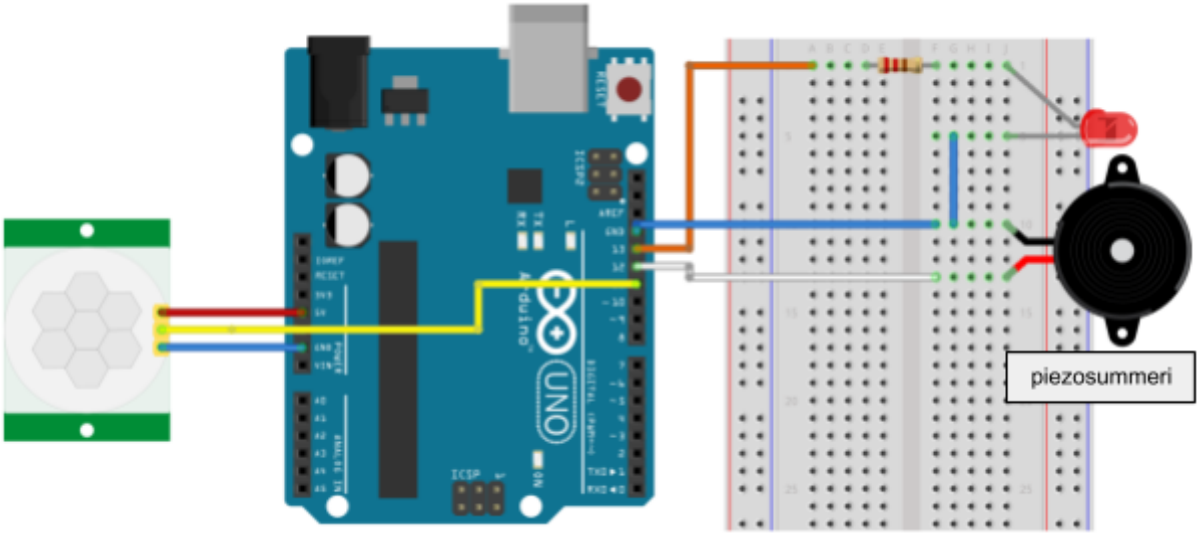
4.
Kytke piezosummerin negatiivinen jalka pisteeseen **J10** ja positiivinen jalka pisteeseen **J13**.

5.
Kytke sininen hyppyjohto pisteisiin **G5** ja **G10**. Johto yhdistää LEDin ja piezosummerin maa-pisteet.

6.
Kytke valkoinen hyppyjohto Arduinon **pinniin D12** ja johdon toinen pää **pisteeseen F13**. Tätä johtoa pitkin tuleva signaali laittee piezosummerin piippaamaan.

7.
Kytke oranssi hyppyjohto Arduinon **pinniin D13** ja johdon toinen pää **pisteeseen A1**. Tätä johtoa pitkin kulkeva signaali vilkuttaa LEDiä.

8.
Kytke sininen hyppyjohto Arduinon **pinniin GND** ja johdon toinen pää **pisteeseen F10**. Johto on LEDin ja piezosummerin yhteinen maajohdin.



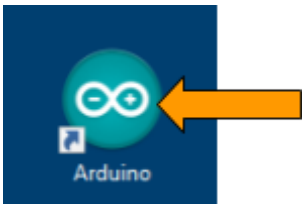
Huom! Käytössäsi oleva piezosummeri voi poiketa kuvassa esitetystä mallista. Jotkin mallit painetaan suoraan kytkentäalustalle niin, etteivät johtimet jää näkyviin.

Kytkentä on nyt valmis.

Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.



Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakiot

Määritetään vakiot, joilla viittaamme ohjelmassa **pinneihin D11** (PIR-moduulu), **D12** (piezosummeri) ja **D13** (LEDi).

```
//vakioiden määritys
#define pirPin 11
#define piezoPin 12
#define ledPin 13
```

Muuttuja

Ohjelmassa käytetään vain yhtä muuttujaa laskemaan [for-silmukan](#) toistojen lukumäärä.

```
//muuttujan esittely
int i;
```

setup-funktio

Määritetään pirPin toimimaan digitaalisena tulona (input) ja piezoPin sekä ledPin digitaalisina lähtöinä (output).

```
void setup() {
  pinMode(pirPin, INPUT);
  pinMode(piezoPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
}
```

loop-funktio

Liiketunnistimena toimivan PIR-moduulin digitaalisen lähdön jännite nousee 5 Volttiin kun se havaitsee liikkeen. Tämä tarkoittaa, että digitaalisen signaalin arvo on tällöin **HIGH**.

Sijoitamme loop-funktioon ehtorakenteen, jossa tutkimme onko PIR-moduulista tulevan signaalin arvo HIGH. Jos on, tehdään hälytys.

```
void loop() {
  //onko PIR-moduuli havainnut liikkeen
  if (digitalRead(pirPin) == HIGH) {

  }
}
```

Ehtorakenteen sisällä sijoitamme hälytyksen tuottavan koodin, jolla vilkutamme LEDiä ja laitamme piezosummerin tuottamaan hälytysäänen. Ohjelmoidaan tämä näin:

- Toistetaan viisi kertaa for-silmukalla:
    - Sytytetään LEDi
    - Ohjataan piezosummeri tuottamaan 1000 Hz taajuista ääntä
    - Odotetaan 250 ms
    - Sammutetaan LEDi
    - Sammutetaan piezosummeri
    - Odotetaan 250 ms
- Hälytyksen jälkeen odotetaan vielä kaksi sekuntia, jottei sama liike aiheuta heti perään uutta hälytystä.

Äänen tuottamiseen piezosummerilla käytetään funktiota tone(<pinni>, <taajuus>). Sillä voidaan tuottaa halutun taajuinen digitaalinen signaali 50 prosentin pulssisuhteella parametrillä ilmoitettuun pinniin. Signaali jatkuu niin kauan, kunnes se sammutetaan funktiolla noTone(<pinni>).

loop-funktio kokonaisuudessaan

```
void loop() {
  //onko PIR-moduuli havainnut liikkeen
  if (digitalRead(pirPin) == HIGH)
  {
    //hälytys, toistetaan 5 kertaa
    for (i=0; i<5; i++) {
      digitalWrite(ledPin, HIGH); //LEDi päälle
      tone(piezoPin,1000);        //1000 Hz ääni piezolla
      delay(250);
      digitalWrite(ledPin, LOW);  //LEDi pois päältä
      noTone(piezoPin);           //piezo pois päältä
      delay(250);
    }
    delay(2000);  //viive hälytyksen jälkeen
  }
}
```

Valmis koodi

```
//vakioiden määritys
#define pirPin 11
#define piezoPin 12
#define ledPin 13

//muuttujan esittely
int i;

void setup() {
  pinMode(pirPin, INPUT);
  pinMode(piezoPin, OUTPUT);
  pinMode(ledPin, OUTPUT);
}

void loop() {
  //onko PIR-moduuli havainnut liikkeen
  if (digitalRead(pirPin) == HIGH)
  {
    //hälytys
    //toistetaan 5 kertaa
    for (i=0; i<5; i++) {
      digitalWrite(ledPin, HIGH); //LEDi päälle
      tone(piezoPin,1000);        //1000 Hz ääni piezolla
      delay(250);
      digitalWrite(ledPin, LOW);  //LEDi pois päältä
      noTone(piezoPin);           //piezo pois päältä
      delay(250);
    }
    delay(2000);  //viive hälytyksen jälkeen
  }
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **"Lähetä"**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

Jos ohjelman koodissa oli virhe, niin prosessi keskeytyy virhe-ilmoitukseen. Tarkista koodi, tee tarvittavat korjaukset ja lähetä uudelleen.

Testaus

Teippaa PIR-moduuli pöytään kiinni niin, että sen linssi (muovikupu) osoittaa valvottavaan suuntaan. Kävele valvottavalla alueella ja laite hälyttää havaittuaan liikettä.

Tasavirtamoottorin ohjaus PWM-signaalilla

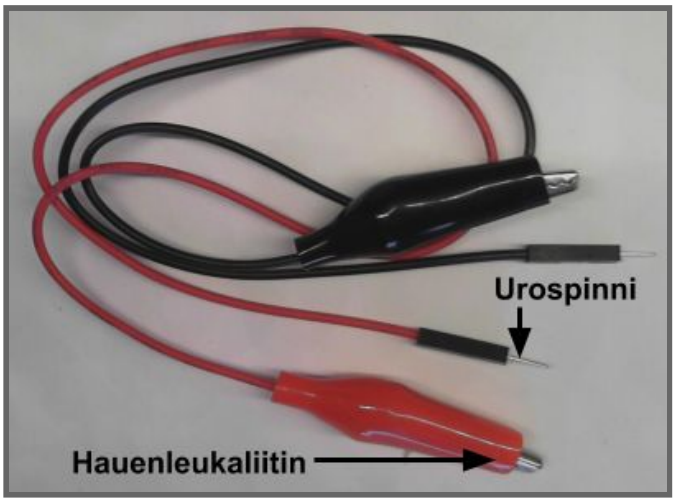
Työn kuvaus

PWM-ohjaus on menetelmä, jolla laitteen tehoa voidaan säätää. Laite voi olla esim LED, hehkulamppu tai sähkömoottori. Tässä työssä käytetään pientä sähkömoottoria (tasavirta), jonka pyörimisnopeutta ohjataan tietokoneella lähettämällä sarjamonitorista plus- ja miinus-merkkejö (plus kasvattaa nopeutta, miinus vähentää nopeutta).

Tasavirtamoottorin lisäksi uusina komponentteina työssä käytetään **FET-transistoria** sekä **diodia** ja perehdytään niiden toimintaan.

Tarvittavat komponentit

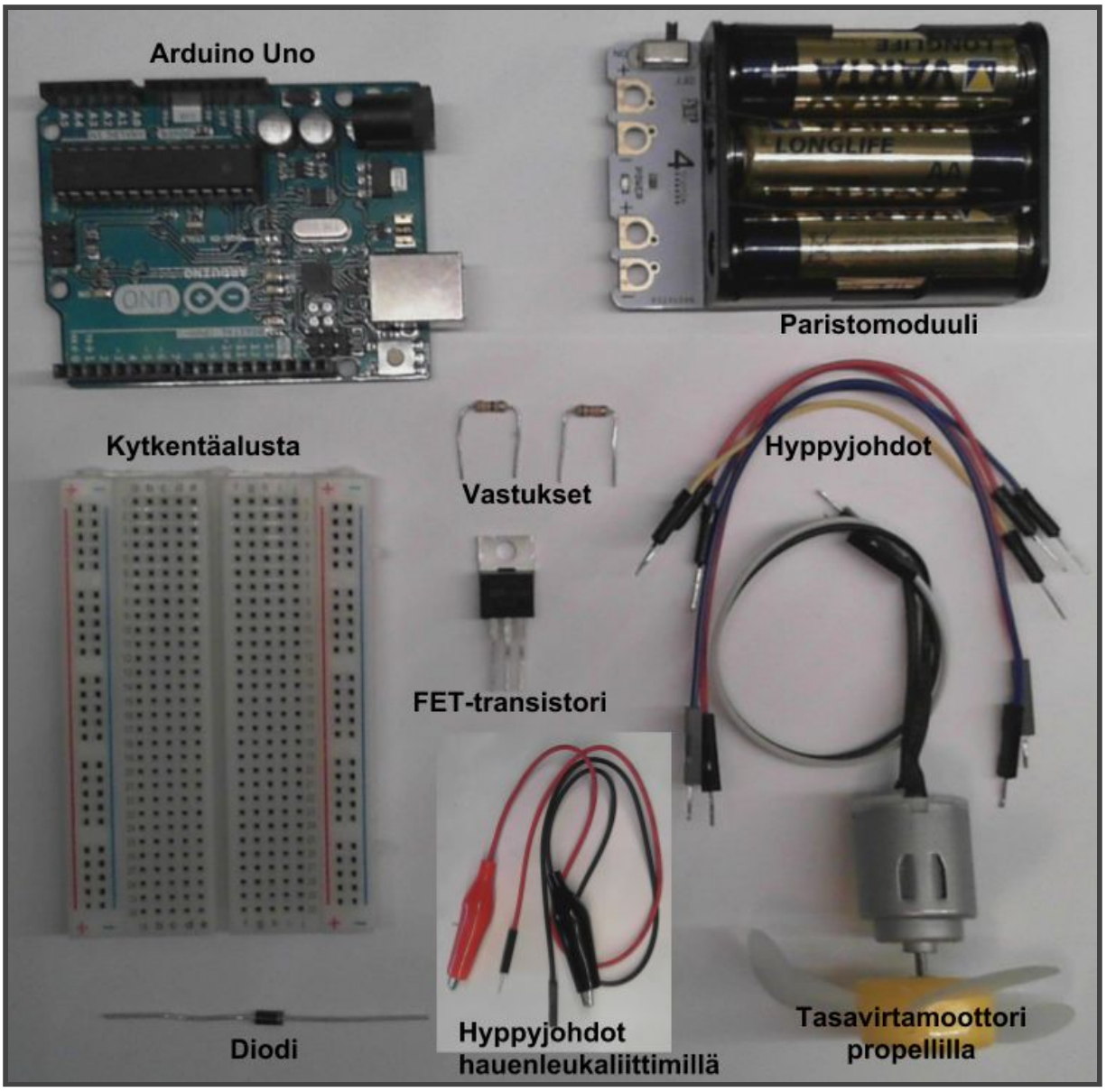
- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- FET-transistori (IRLZ44N)
- Diodi
- 330 Ohmin ja 10 Kohmin vastukset
- Tasavirtamoottori propellilla ja johdoilla
- 4,5 Voltin paristomoduuli
- Kytkentäalusta
- Kolme hyppyjohtoa (uros-uros): punainen, keltainen ja sininen
- Kaksi hyppyjohtoa, jonka toisessa päässä on **hauenleukaliitin** ja toisessa päässä **urospinni** (punainen ja musta)



Tarvittavat työkalut

- Pienikärkiset pihdit

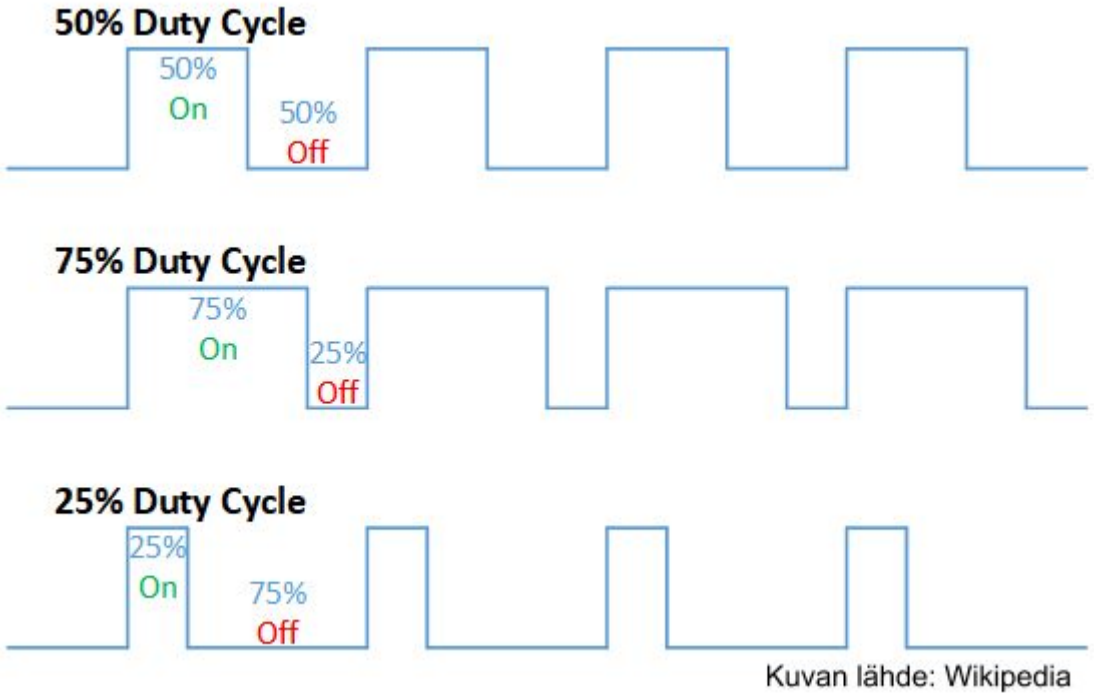
Työhön tarvittavat komponentit:



PWM-ohjaus

Lyhenne PWM tulee sanoista Pulse Width Modulation, eli suomeksi pulssinleveysmodulaatio. PWM-signaali on digitaalinen signaali, eli sillä voi olla vain kaksi mahdollista tilaa, joko ON tai OFF. ON ja OFF tilojen välistä suhdetta muuttamalla voidaan muuttaa ohjattavan laitteen

tehoa. Alla olevassa kuvassa on esitetty PWM-signaali pulssisuhteilla 50, 75 ja 25 prosenttia.



Kun pulssisuhde on 50% signaalin ON ja OFF aika on yhtä suuri. 75% signaalissa ON aika kestää 75% ja OFF aika 25% signaalin jaksonajasta (joka riippuu signaalin taajuudesta).

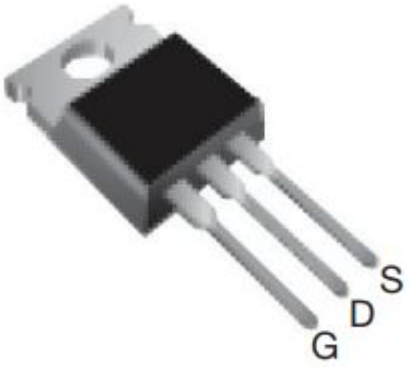
Mitä suurempi pulssisuhde on, eli mitä pidempi ON-aika on suhteessa OFF-aikaan, sitä enemmän tehoa menee ohjattavaan laitteeseen. Esimerkiksi sähkömoottorin tapauksessa 75% PWM-signaali ohjaa sähkömoottorin pyörimään paljon nopeammin kuin 25% signaali. Video: What is PWM?

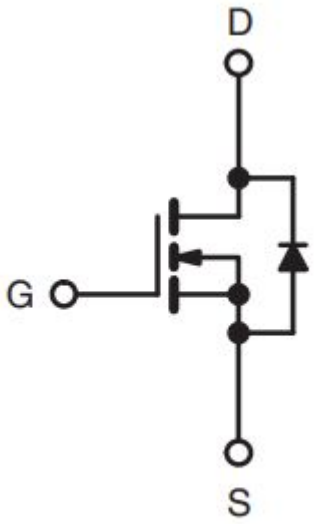
Arduino Unon digitaalisia lähtöjä **D3**, **D5**, **D6**, **D9**, **D10** ja **D11** voidaan käyttää PWM-signaalin tuottamiseen.

FET-transistori

Arduinon digitaalisen lähdön virta voi olla enintään 20 milliAmpeeria. Sen suuremmalla virralla lähtöjä ei saa kuormittaa. Sähkömoottori tarvitsee kuitenkin paljon suuremman virran. Ratkaisemme tämän käyttämällä ulkoista 4,5 Voltin paristoa ja FET-transistoria.

Transistori on komponentti joka voi hyvin pienellä teholla (virralla / jännitteellä) ohjata hyvin suurta tehoa. Tässä kytkennässä käytämme FET-transistoria kytkimenä ja ohjaamme sitä PWM-signaalilla. FET-transistoria ohjataan jännitteellä, eli sen ohjaamiseen ei tarvita tehoa juuri ollenkaan.





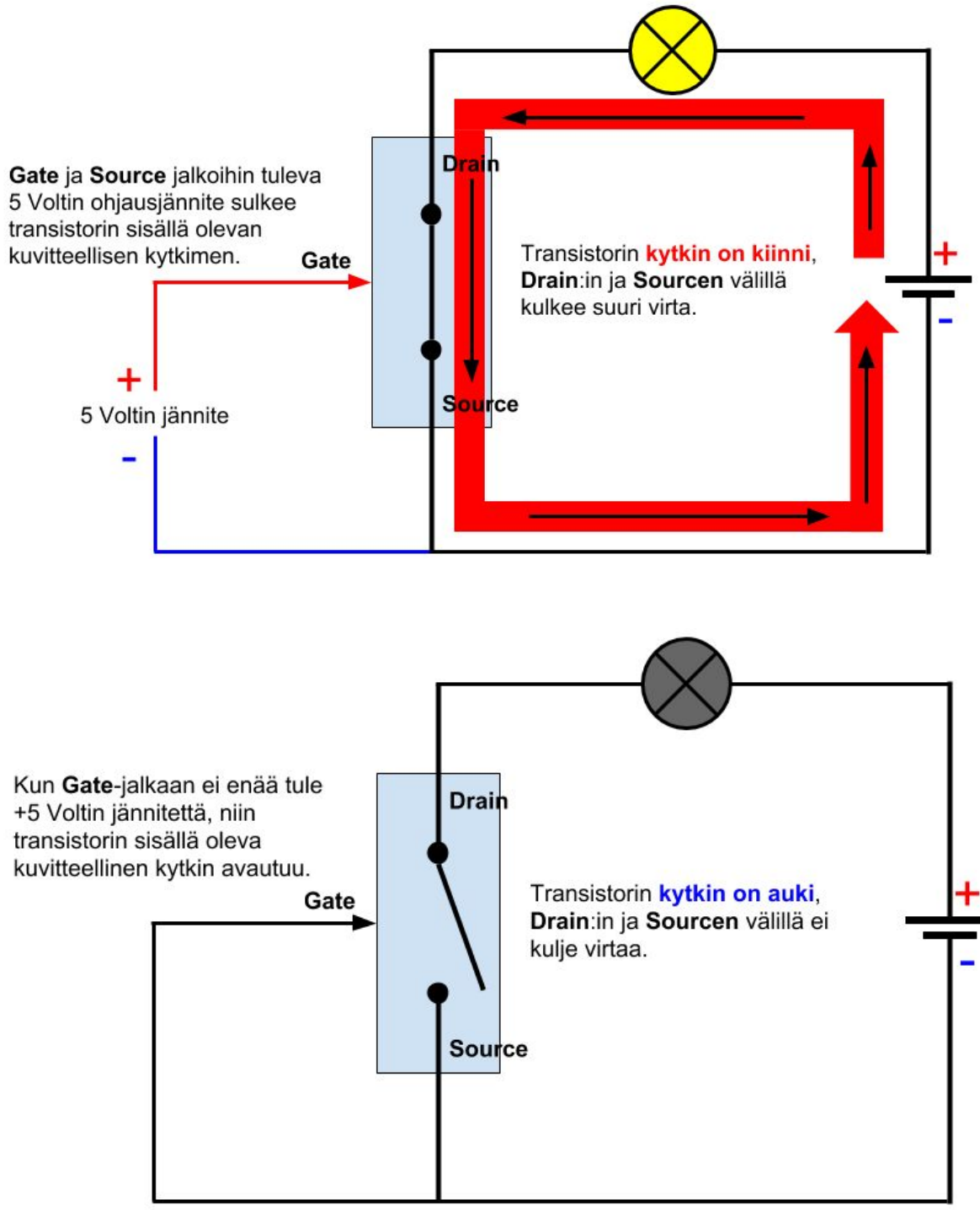
FET-transistorissa on kolme jalkaa:

- G = Gate, hila
- D = Drain, nieli
- S = Source, lähde

Eri transistorimalleissa jalkojen järjestys voi vaihdella, mutta tässä mallissa se on kuvan mukainen Gate, Drain ja Source vasemmalta oikealle, komponentin edestä katsottuna.

FET-transistori kytkimenä

Kun käytämme transistoria kytkimenä, voimme yksinkertaistaa sen toiminnan alla olevien kuvien mukaisesti.

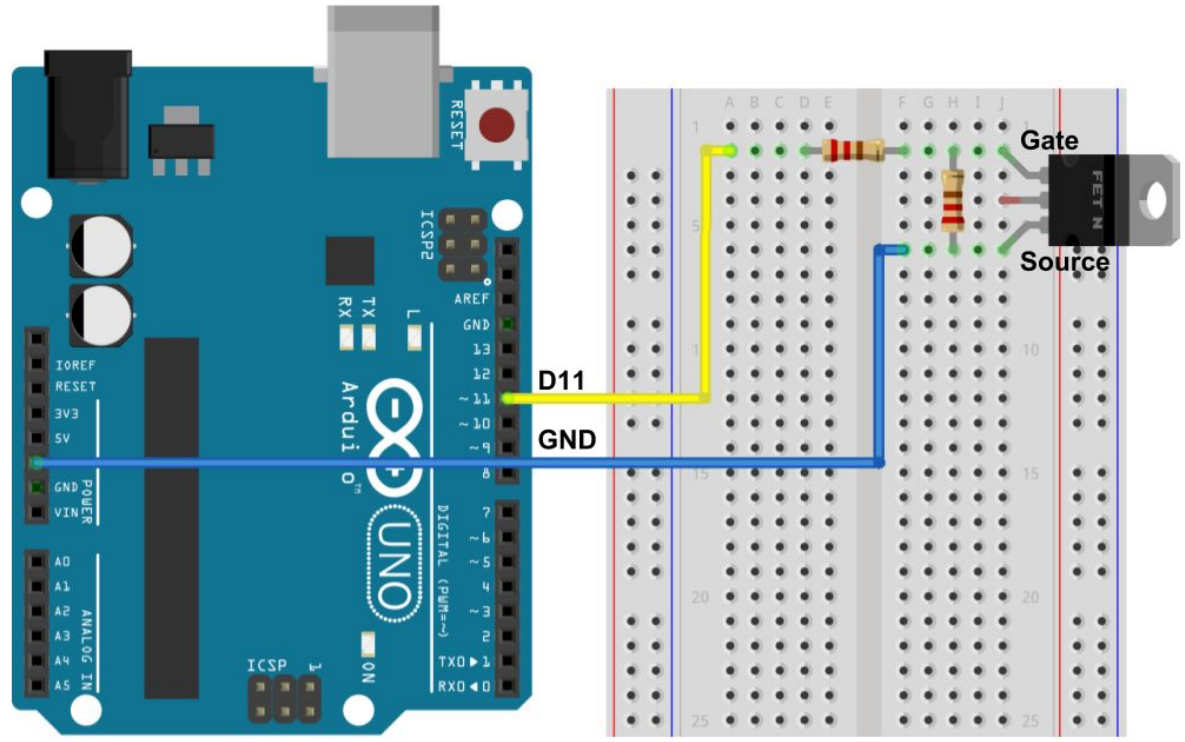


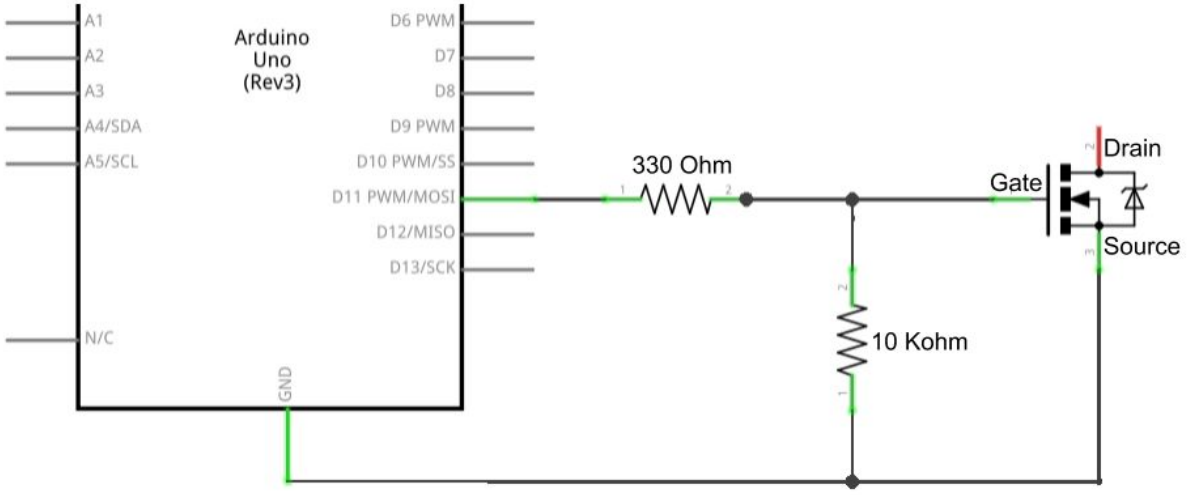
Transistorin datalehdestä näemme, että se pystyy kytkemään jopa 50 Ampeerin virran (I_D), mutta kestää enintään vain 60 Voltin jännitettä.

FET-transistorin kytkentä Arduinoon

+5 Voltin ohjausjännite transistorin **Gate**-jalkaan saadaan Arduinon digitaalisesta lähdöstä. Transistorin **Source**-jalka liitetään Arduinon

GND-pinniin. Alla olevassa kuvassa on esitetty pelkästään transistorin ohjaamiseen tarvittava yleinen kytkentä. Siinä ei ole mukana ohjattavaa virtapiiriä. Tässä työssä transistori kytketään vähän eri kohtaan kytkentäalustalle, joten älä tee kytkentää vielä tämän kuvan perusteella.

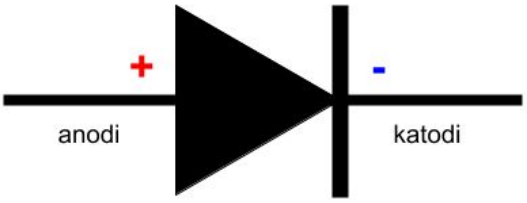




Arduinon lähdön **D11** ja transistorin **Gate**-jalan välissä on 330 Ohmin vastus. Lisäksi Transistorin **Gate** on kytketty Arduinon **GND-pinniin** 10 Kohmin vastuksella (varmistaa transistorin siirtymisen OFF-tilaan).

Diodi

Diodi on komponentti, joka johtaa virtaa vain yhteen suuntaan. Oikeastaan olet jo käyttänyt diodia, sillä LED on valoa tuottava diodi (Light Emitting Diode). Tavallinen diodi ei tuota valoa, se ainoastaan johtaa virtaa yhteen suuntaan. Diodeja käytetään yleisesti mm. tasasuuntaamaan vaihtojännite tasajännitteeksi (diodisilta, 4 diodia).

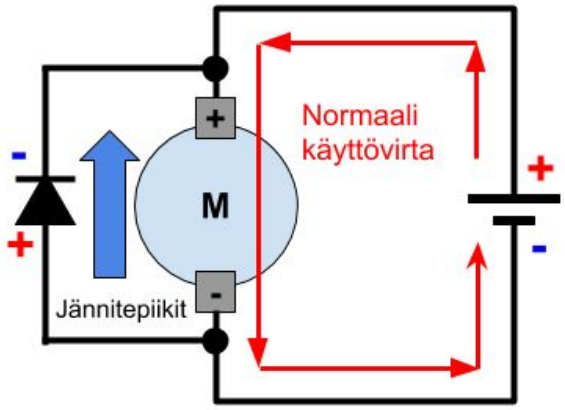


Diodin piirrosmerkki

Diodi komponenttina

Käytämme kytkennässä sähkömoottoria ja se on ns. induktiivinen komponentti. Moottorin sisällä oleva ohut kuparilanka muodostaa induktiivisen kelan. Ja kun tällaisesta induktiivisesta laitteesta katkaistaan äkillisesti jännite (FET-kytkin avautuu), niin seurauksena on suuria jännitepiikkejä. Induktiivisen laitteen aiheuttamat jännitepiikit voivat helposti rikkoa FET-transistorin, joten se on suojattava niiltä.

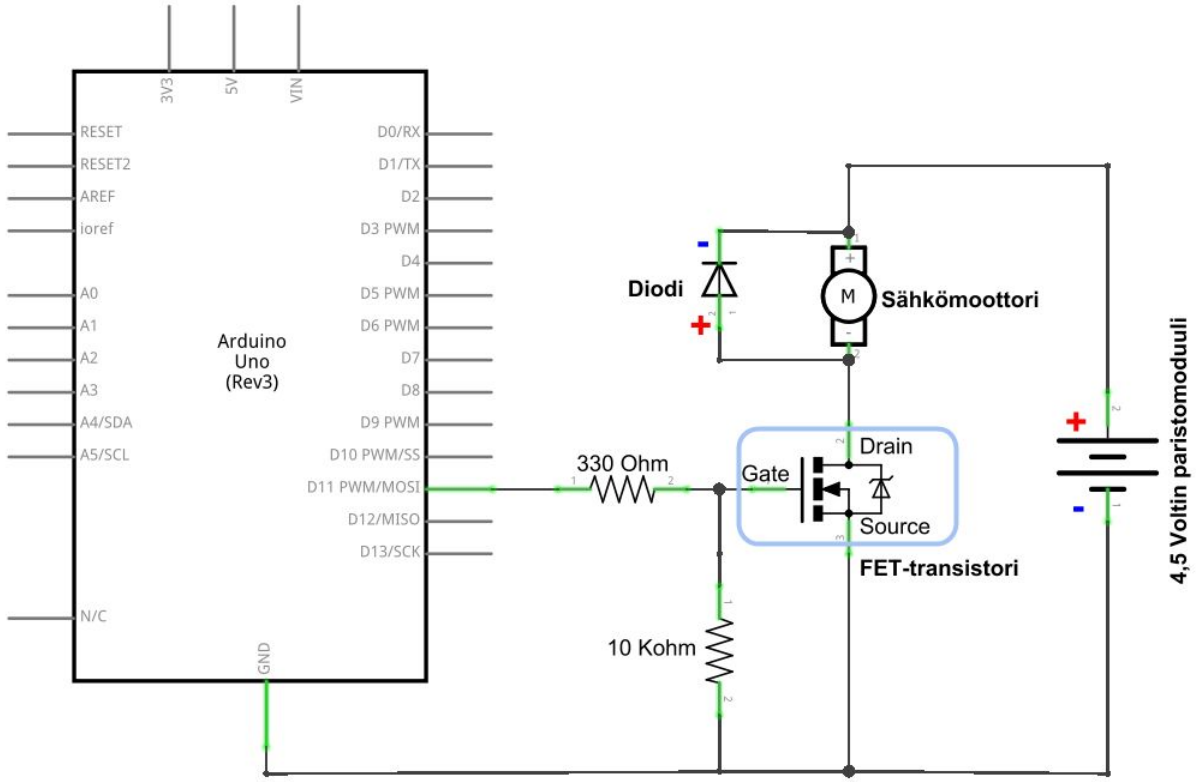
Jännitepiikit voidaan eliminoida kytkemällä diodi estosuuntaan sähkömoottorin rinnalle. Estosuuntaan kytkeminen tarkoittaa, että diodin läpi ei kulje normaali ulkoisen pariston tuottama virta (se kulkee ainoastaan moottorin läpi). Diodi kytketään estosuuntaan moottorin rinnalle kuvan mukaisesti.



Sähkömoottorin lisäksi toinen yleinen induktiivinen laite on rele. Diodi kytketään releen kelan rinnalle ihan samalla tavalla.

Kytkennän rakennus

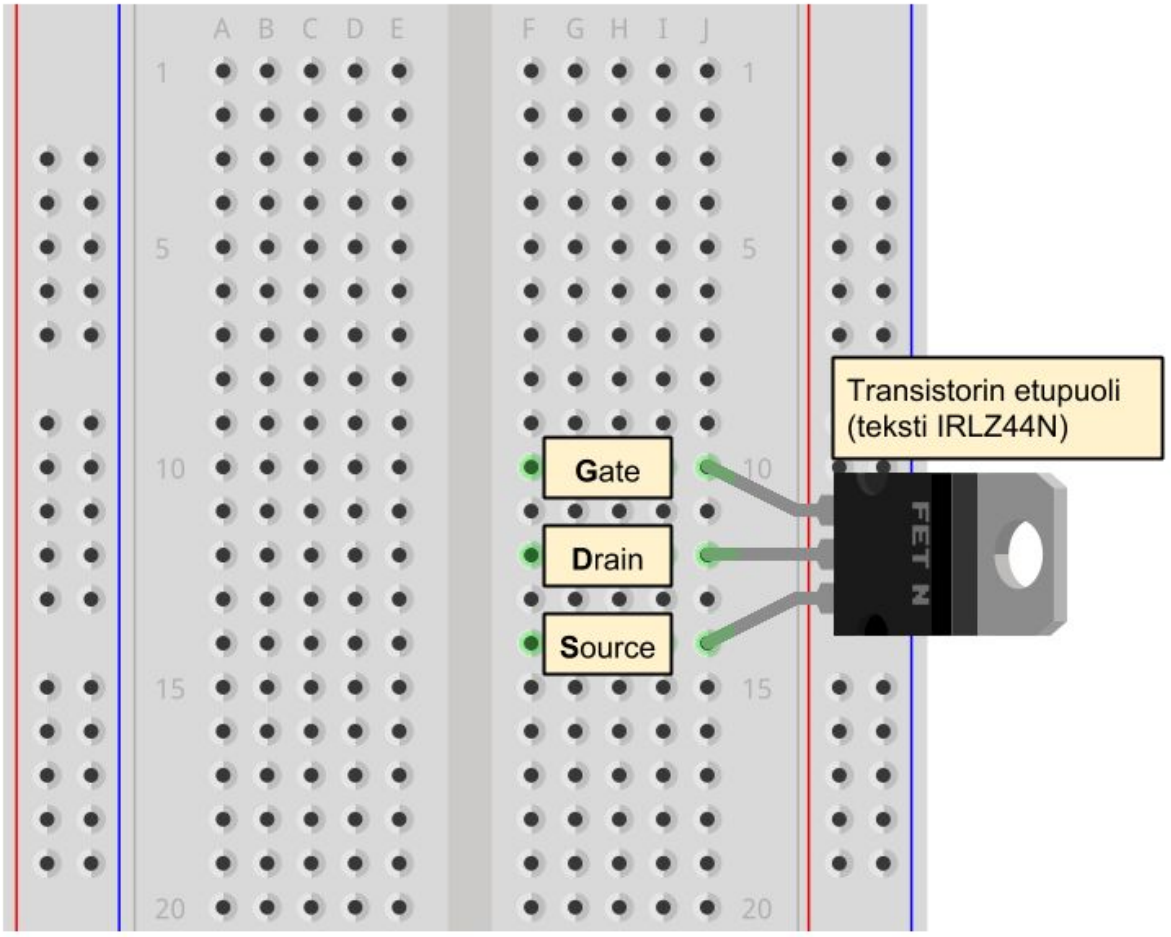
Virtapiirin kytkentäkaavio



1.
Taivuta pihdeillä transistorin jalkoja niin, että kytkentäalustalla kahden vierekkäisen jalan väliin jää yksi tyhjä pisterivi.

Paina transistori kiinni kytkentäalustaan näin:

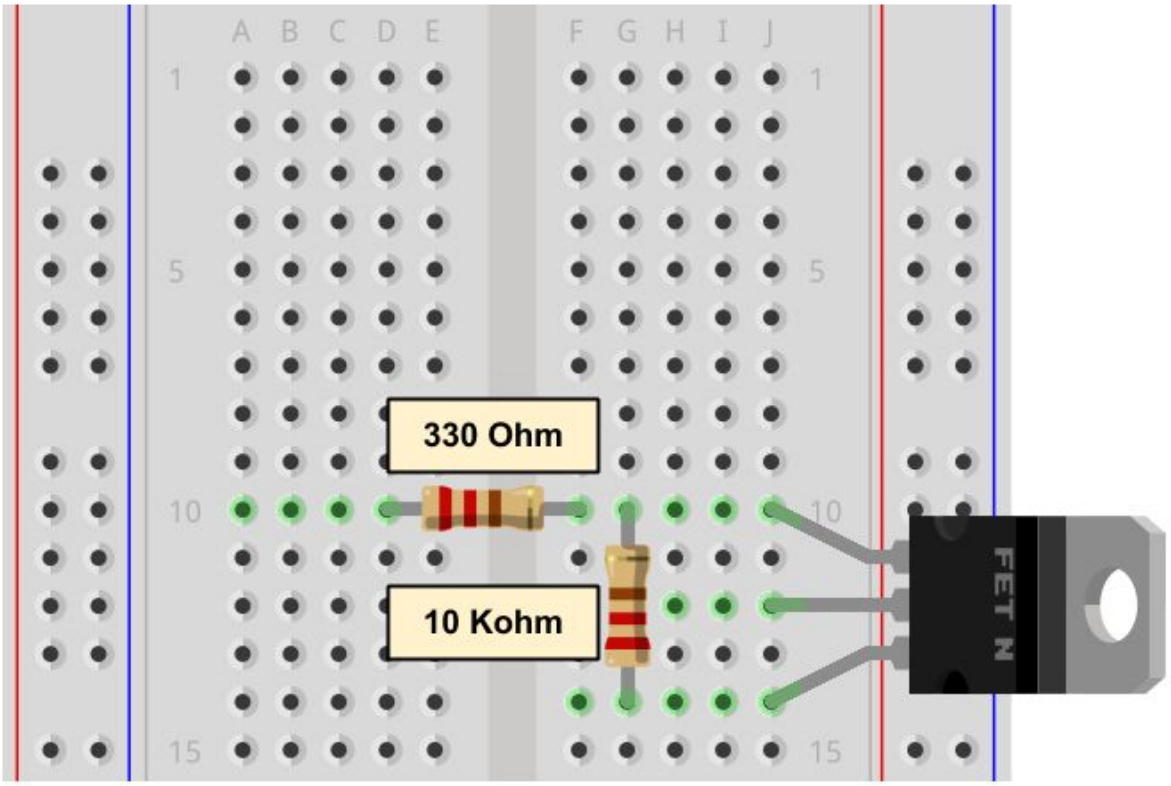
- Transistorin tekstipuoli (etupuoli) osoittaa vasemmalle
- **Gate**-jalka tulee pisteesee **J10**
- **Drain**-jalka tulee pisteeseen **J12**
- **Source**-jalka tulee pisteeseen **J14**



2.

Paina **330 Ohmin** vastus kiinni pisteisiin **D10** ja **F10**.

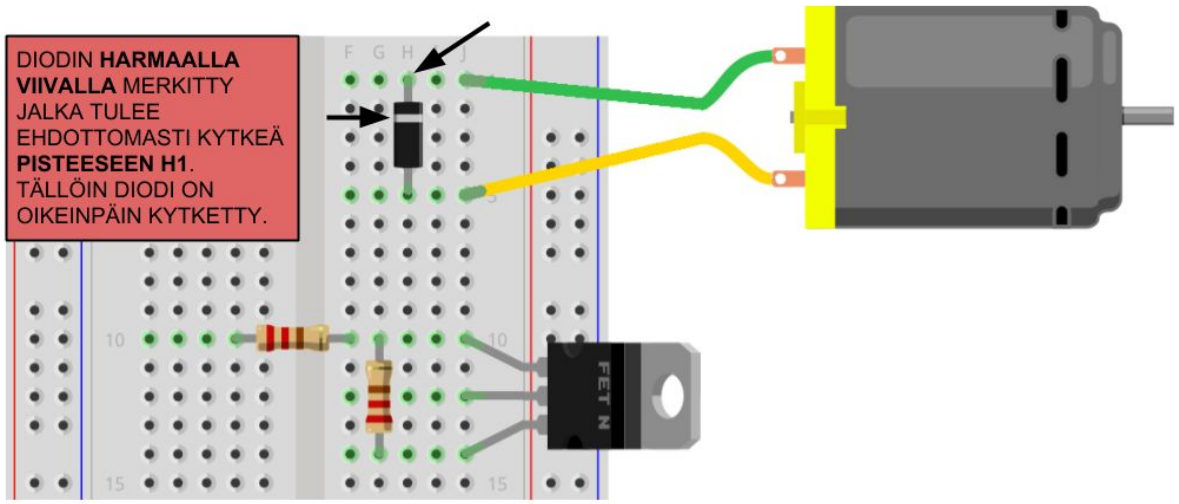
3.
Paina **10 Kohmin** vastus kiinni pisteisiin **G10** ja **G14**.



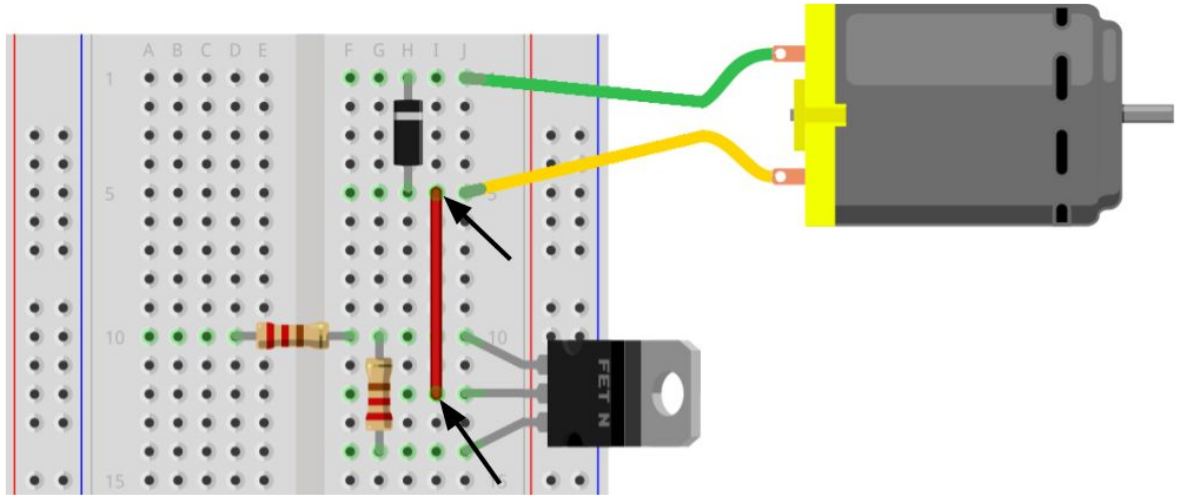
4.
Kytke tasavirtamoottorin johdot pisteisiin **J1** ja **J5**. Ei ole väliä miten päin johdot kytket, järjestys vaikuttaa ainoastaan moottorin pyörimissuuntaan.

5.
Kytke diodi pisteisiin **H1** ja **H5**. Diodin toinen jalka on merkitty komponentin runkoon harmaalla viivalla. KYTKE DIODI EHDOTTOMASTI

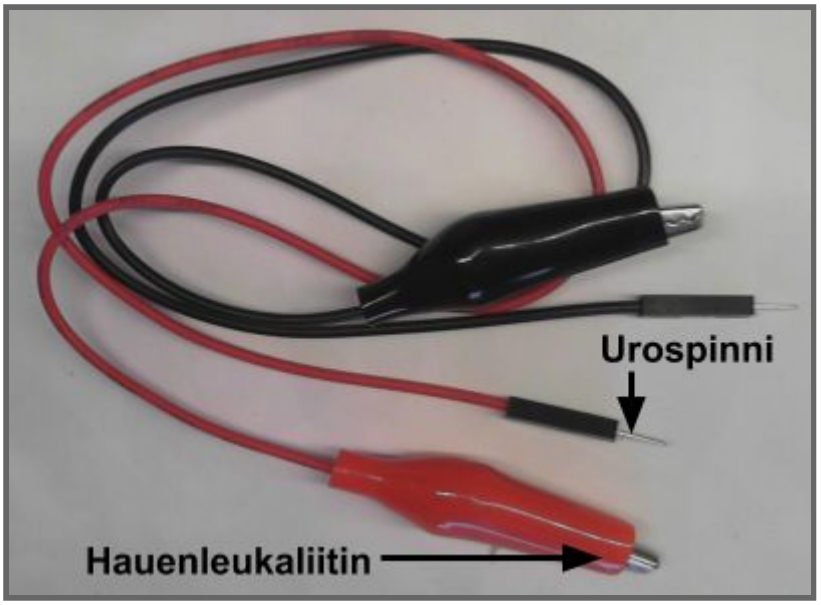
NIIN PÄIN, ETTÄ HARMAALLA VIIVALLA MERKITTY JALKA TULEE PISTEESEEN H1.



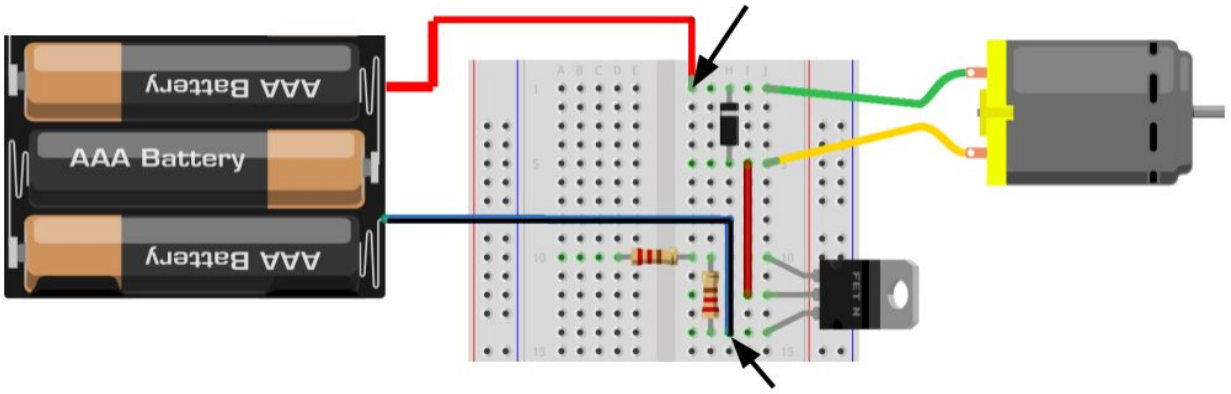
6.
Kytke punainen hyppyjohto pisteisiin **i5** ja **i12** (transistorin **Drain**-jalka).



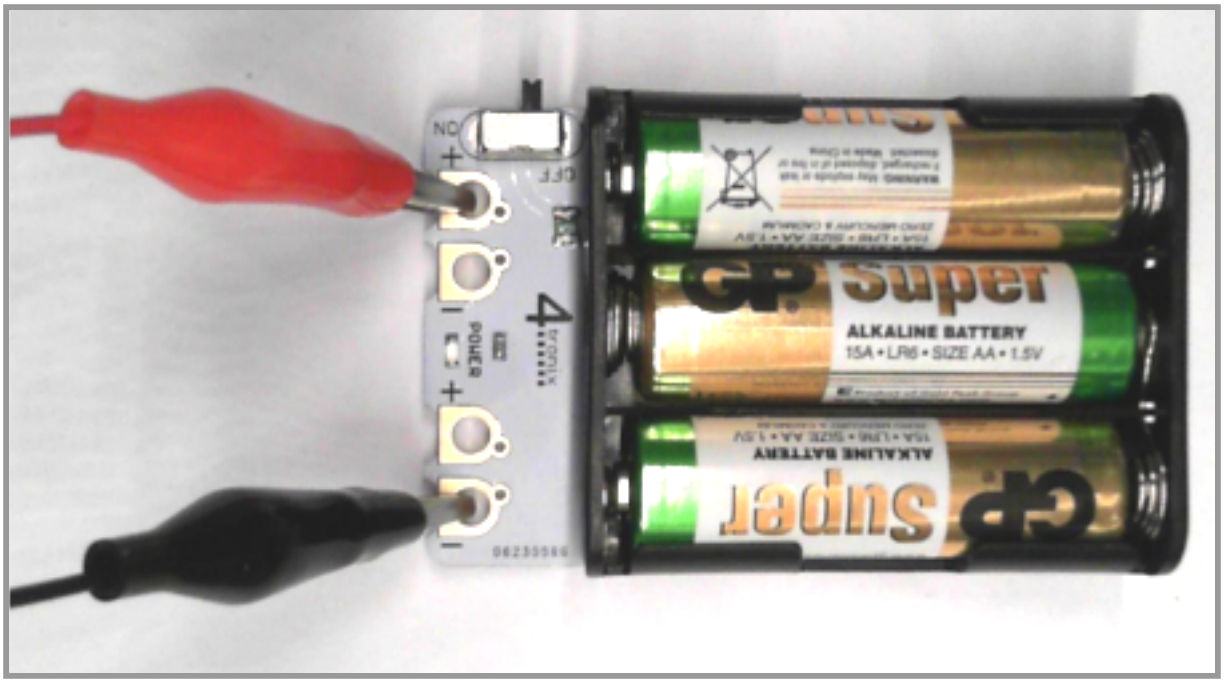
7.
Seuraavaksi liitetään virtalähteenä toimiva paristomoduuli kytkentään käyttäen ao. kuvassa esitettyjä johtoja.



8.
Kytke punaisen hyppyjohdon urospinni pisteeseen **F1**.
Kytke mustan hyppyjohdon urospinni pisteeseen **H14**.



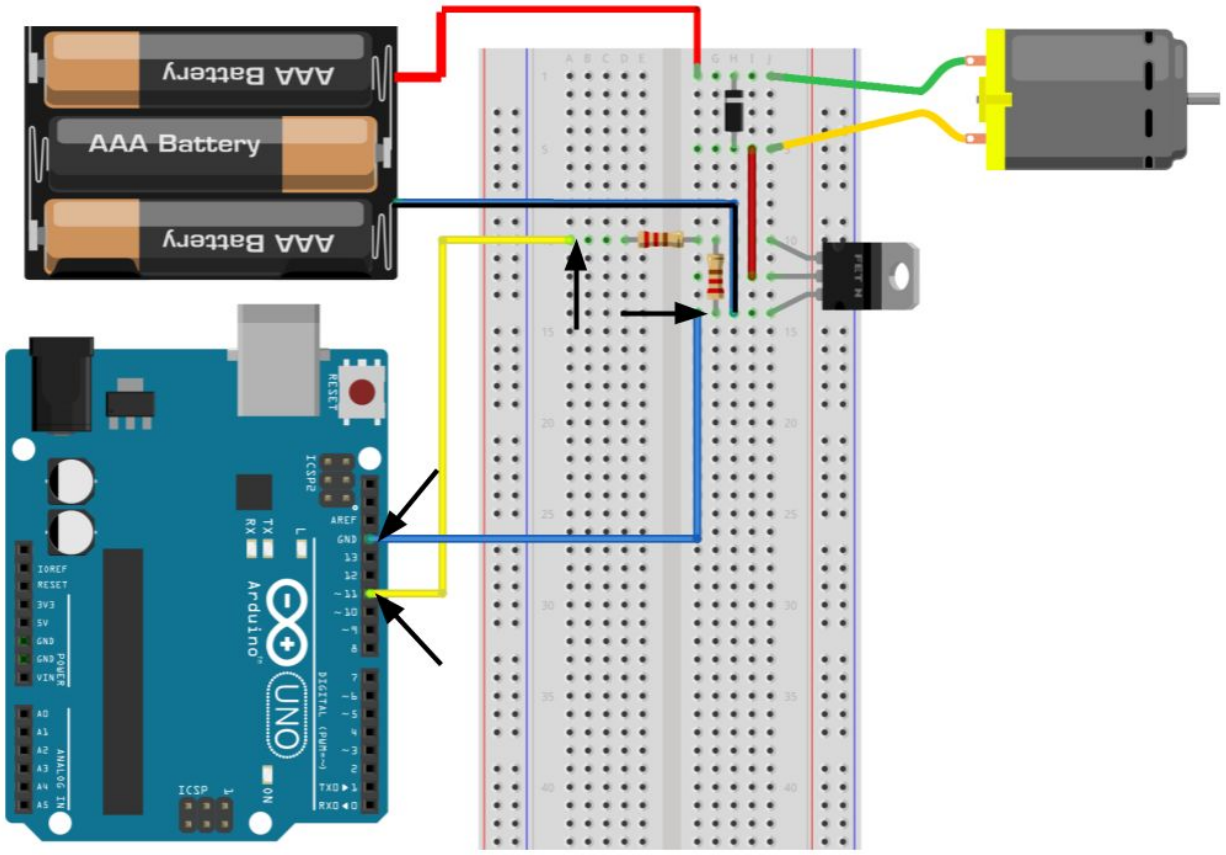
9.
Kytke **punainen hauenleukaliitin** paristomoduulin **plus-napaan** ja **musta hauenleukaliitin** paristomoduulin **miinus-napaan**.



10.

Kytke keltainen hyppyjohto Arduinon **pinnii D11** ja johdon toinen pää pisteeseen **A10**. Tätä johtoa pitkin PWVM-signaali kulkee Arduinosta transistorin **Gate**-jalkaan.

11.
Kytke sininen hyppyjohto Arduinon **pinniin GND** ja johdon toinen pää pisteeseen **F14**. Tämä johto on PWM-signaalin maajohto.



Kytkentä on nyt valmis.

Ohjelmointi

Aloitus

Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.

Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Vakio

Määritetään vakio **pwmPin**, jolla viittaamme pinniin **D11**.

```
//vakion määritys
#define pwmPin 11
```

Muuttujat

Esitellään kaksi muuttujaa. Muuttujassa pwmValue tulemme säilyttämään PWM-signaalin arvoa ja muuttujaan cInput luemme tietokoneelta tulevat merkit.

```
//muuttujien esittely ja alustus
int pwmValue = 0;
char cInput;
```

setup-funktio

Avataan sarjaliikenneyhteys nopeudella 9600 bps ja asetetaan pinni **pwmPin** toimimaan digitaalisena lähtönä.

```
void setup() {
  Serial.begin(9600);
  pinMode(pwmPin, OUTPUT);
}
```

loop-funktio

Koodataan ohjelma toimimaan näin:

- Kun tietokone lähettää tietoa, luetaan merkki muuttujaan cInput.
- Jos merkki oli plus (+), kasvatetaan PWM-signaalin pulssisuhteen ON-aikaa (moottori pyörii nopeammin).
- Jos merkki oli miinus (-), pienennetään PWM-signaalin pulssisuhteen ON-aikaa (moottori pyörii hitaammin).

```
void loop() {
  //onko mikro-ohjaimen sarjaporttiin tullut tietoa
  if (Serial.available()) {
    cInput = Serial.read(); //luetaan tieto muuttujaan cInput
    if (cInput == 43) {
      //jos merkki on +
    }
    if (cInput == 45) {
      //jos merkki on -
    }
  }
}
```

`Serial.available()` palauttaa arvo `true`, jos tietoa on luettavissa ja silloin suoritetaan ehtorakenteessa oleva koodi. Komennolla `Serial.read()` luetaan tieto muuttujaan `cInput`. Tämän jälkeen tutkitaan kahdella ehtorakenteella oliko vastaanotettu merkki plus tai oliko se miinus.

PWM-signaalin minimiarvo on 0 ja maksimiarvo 255. Koodissa tulee huolehtia siitä, että signaalin arvoksi ei anneta negatiivista lukua eikä suurempaa lukua kuin 255. Ratkaisemme tämän ehtorakenteilla.

```
    if (cInput == 43) {
      //jos merkki on +
      if (pwmValue <= 238) pwmValue += 17;
    }
    if (cInput == 45) {
      //jos merkki on -
      if (pwmValue >= 17) pwmValue -= 17;
    }
```

PWM-signaali tuotetaan funktiolla analogWrite(pinni, arvo). Funktion nimessä on vähän harhaanjohtavasti teksti **analog**, sillä PWM-signaali on kuitenkin digitaalinen signaali. Haluttaessa siitä tosin voidaan ulkopuolisella elektroniikalla tehdä analoginen signaali (RC-piiri). Tässä kytkennässä sitä käytetään kuitenkin ihan digitaalisena signaalina ohjaamaan FET-transistoria.

loop-funktion koko koodi:

```
void loop() {
  //tutkitaan, onko mikro-ohjaimen sarjaporttiin tullut tietoa
  if (Serial.available()) {

    //luetaan tieto muuttujaan cInput
    cInput = Serial.read();

    //tutkitaan, onko merkki + tai -
    if (cInput == 43) {
      //jos merkki on +
      if (pwmValue <= 238) pwmValue += 17;
    }

    if (cInput == 45) {
      //jos merkki on -
      if (pwmValue >= 17) pwmValue -= 17;
    }

    //muodostetaan PWM-signaali
    analogWrite(pwmPin, pwmValue);

    //lasketaan ja lähetetään tietokoneelle pulssisuhde
    Serial.print("PWM = ");
    Serial.print((pwmValue*100) / 255);
    Serial.println(" %");
  }
}
```

Lisäsimme koodiin lihavoitujen kommenttirivien yhteydessö olevan koodin. Koodin lopussa laskemme PWM-signaalin **ON-ajan** prosentteina suhteessa **OFF-aikaan** (pulssisuhde) ja lähetämme tiedon tietokoneelle.

Valmis ohjelma:

```
//vakion määritys
#define pwmPin 11

//muuttujien esittely ja alustus
int pwmValue = 0;
char cInput;

void setup() {
  Serial.begin(9600);
  pinMode(pwmPin, OUTPUT);
}

void loop() {
  //tutkitaan, onko mikro-ohjaimen sarjaporttiin tullut tietoa
  if (Serial.available()) {

    //luetaan tieto muuttujaan cInput
    cInput = Serial.read();

    //tutkitaan, onko merkki + tai -
    if (cInput == 43) {
      //jos merkki on +
      if (pwmValue <= 238) pwmValue += 17;
    }
    if (cInput == 45) {
      //jos merkki on -
      if (pwmValue >= 17) pwmValue -= 17;
    }

    //muodostetaan PWM-signaali
    analogWrite(pwmPin, pwmValue);

    //lasketaan ja lähetetään tietokoneelle pulssisuhde
    Serial.print("PWM = ");
    Serial.print((pwmValue*100) / 255);
    Serial.println(" %");
  }
}
```

Ohjelman kääntäminen ja lähetys Arduinoon

Tallenna ohjelma haluamallasi nimellä.
Liitä Arduino USB-kaapelilla tietokoneen USB-porttiin.

Varmista ohjelmointiympäristöstä, että valittuna on oikea Arduinon malli (**Työkalut → Kortti → Arduino / Genuino Uno**).
Varmista, että valittuna on oikea COM-portti (**Työkalut → Portti → COMx**).

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **"Lähetä"**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

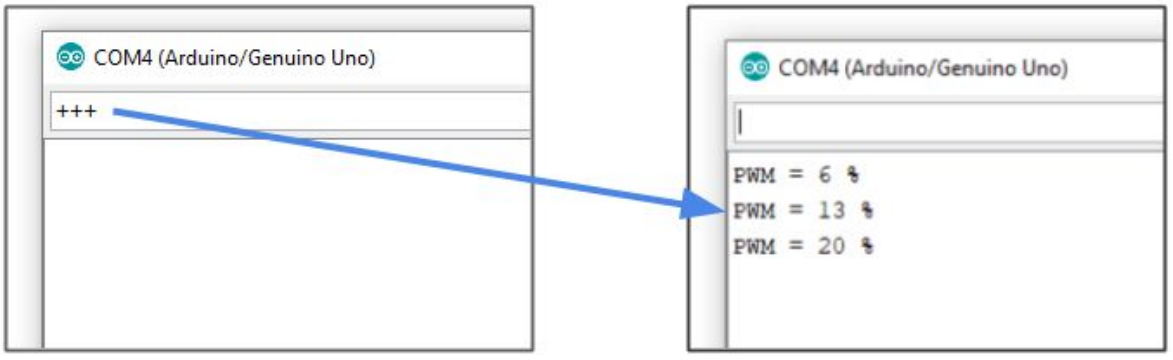
Testaus

1. Teippaa moottori pöytään tai muuhun alustaan niin, että propelli pääsee pyörimään vapaasti.

2. Kytke paristomoduulin virtakytkin **ON**-asentoon.

3. Käynnistä sarjamonitori valikosta **Työkalut / Sarjamonitori**.

4. Kirjoita lähetyskenttään kolme **plus-merkkiä** ja paina **enter**.



PWM-signaalin pulssisuhde nousee nollasta 20 prosenttiin. Plus-merkkejä lähettämällä voit kasvattaa pulssisuhdetta ja moottorin pyörimisnopeutta. Miinus-merkeillä pulssisuhde puolestaan pienenee ja moottori hidastuu.

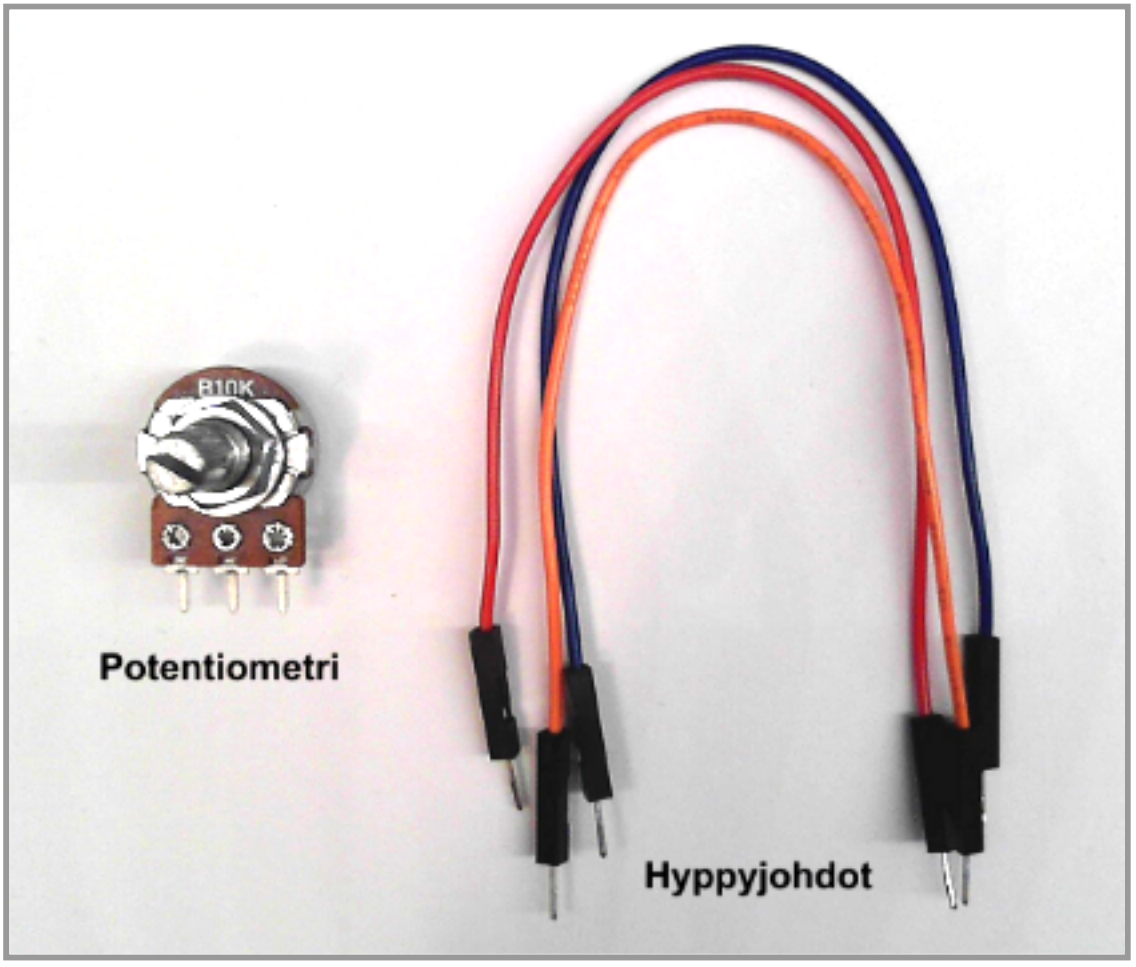
Pulssisuhteella 0% moottori sammuu kokonaan (**ON-aika** nolla) ja pulssisuhde 100% vastaa tilannetta, jossa moottori on kytketty suoraan

paristoon (**OFF-aika** nolla). Voit lähettää kerralla yhden tai useamman ohjausmerkin.

Lisätehtävä

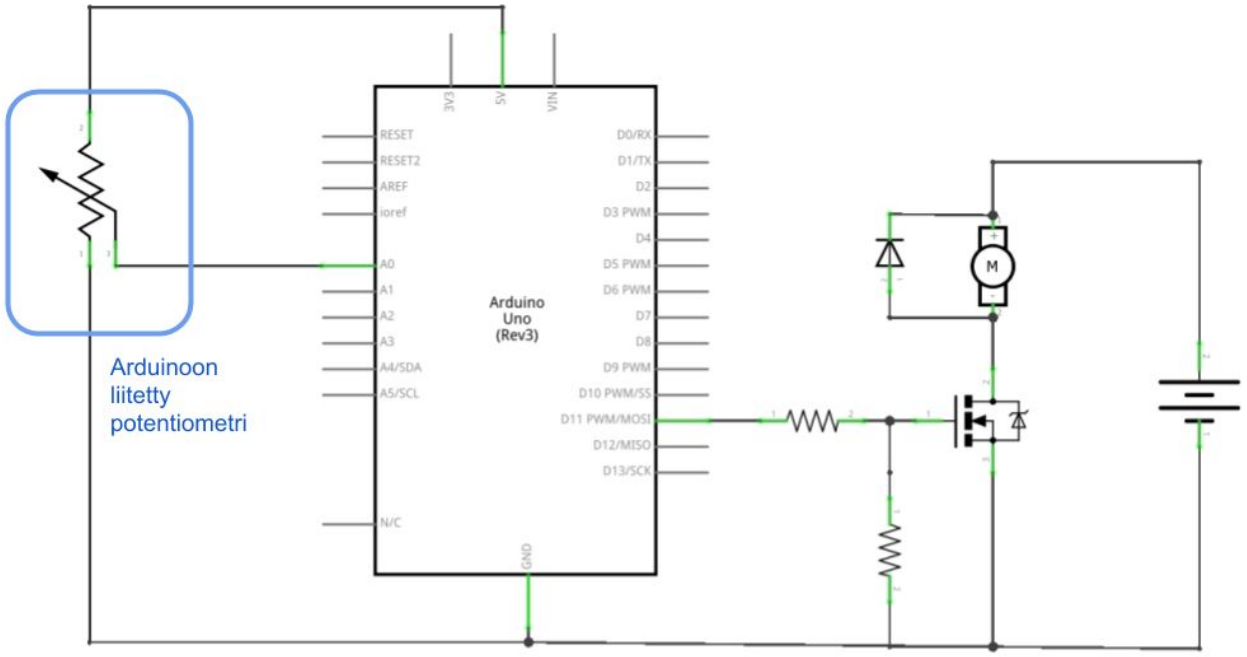
Muutetaan PWM-ohjain toimimaan ilman tietokonetta itsenäisenä laitteena. Toteutetaan tämä niin, että PWM-signaalin pulssisuhde asetetaan potentiometrilla. Tähän muutokseen tarvitaan seuraavat komponentit:

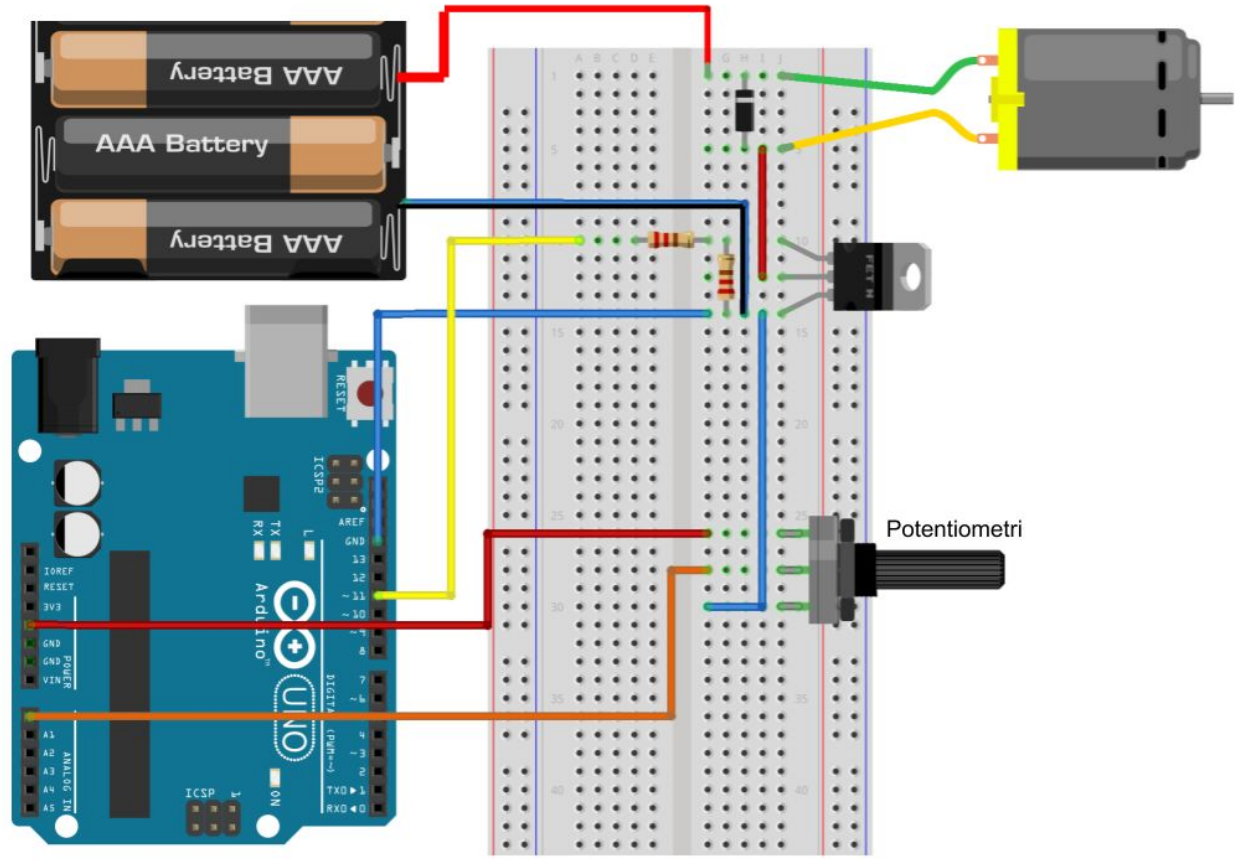
- 10 Kohmin potentiometri
- Kolme hyppyjohtoa (uros-uros): punainen, sininen ja oranssi



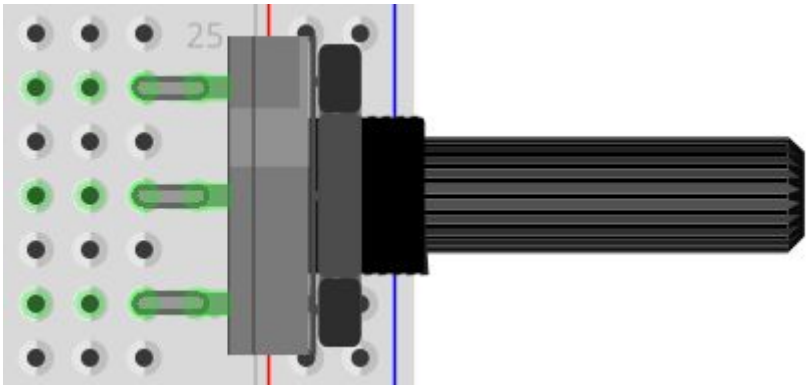
Kytkennän rakennus

Kaikki aiemmat komponentit säilyvät kytkennässä, lisäämme siihen vain potentiometrin johtoineen alla olevan kytkentäkaavion mukaisesti.





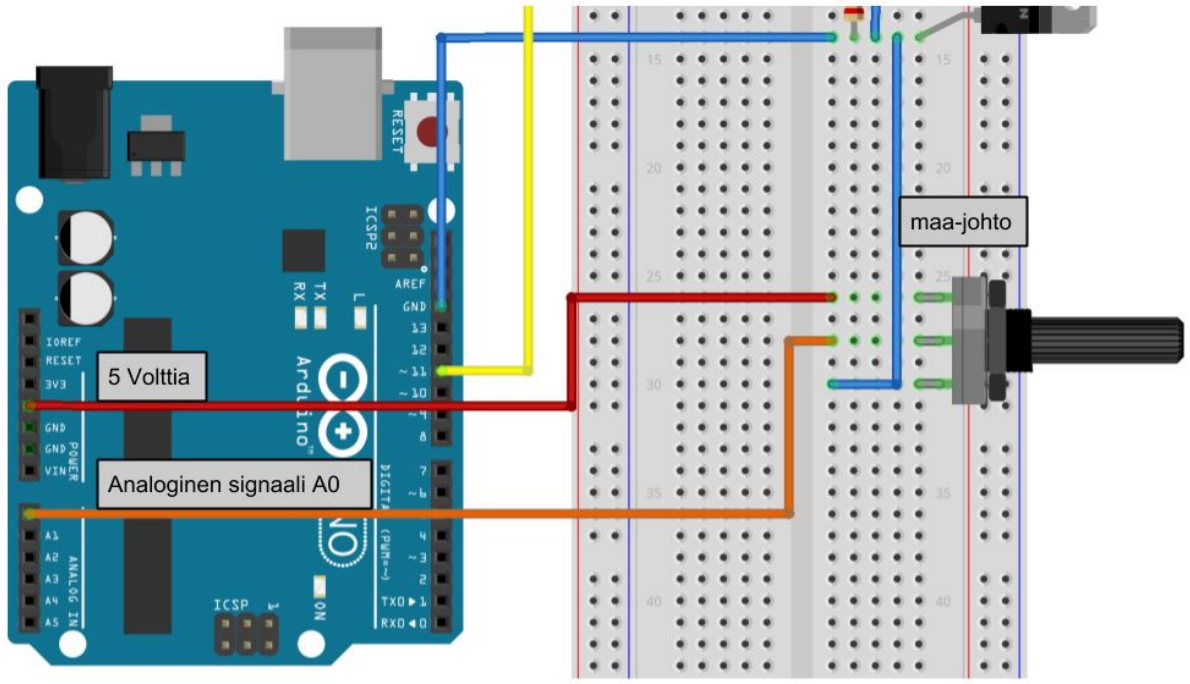
1.
Kytke potentiometri kytkentäalustalle pisteisiin **J26**, **J28** ja **J30**. Potentiometrin säätönuppi osoittaa oikealle.



2.
Liitä sininen hyppyjohto pisteeseen **i14** ja johdon toinen pää pisteeseen **F26**. Tämä on maa-johto.

3.
Liitä punainen hyppyjohto Arduinon **pinniin 5V** ja johdon toinen pää pisteeseen **F30**. Tätä johtoa pitkin potentiometriin tulee 5 Voltin jännite.

4.
Liitä oranssi hyppyjohto Arduinon **pinniin A0** ja johdon toinen pää pisteeseen **F28**. Tätä johtoa pitkin potentiometrin tuottama analoginen signaali kulkee Arduinon analogiseen tuloon A0.



Ohjelmointi

Tee ohjelmointiympäristössä uusi tyhjä ohjelmarunko ja tallenna se haluamallasi nimellä.

Teemme ohjelman, joka lukee poteniometriltä tulevan analogisen signaalin arvon ja generoi siitä arvoa vastaavan pulssisuhteen PWM-signaaliin. Tällaista laitetta on mahdollista käyttää ilman tietokonetta.

Vakiot

Potentiometrin signaali tulee analogiseen tuloon **A0**, joten tehdään vakio **potPin**, jolla viittaamme siihen. Vakio **pwmPin** on sama kuin aiemmassa ohjelmassa.

```
//vakioiden määritys
#define potPin A0
#define pwmPin 11
```

Muuttujat
Analogisen signaalin arvon tallennusta ja käsittelyä varten tehdään int-tyyppinen (kokonaisluku) muuttuja **aValue**. Muuttuja **pwmValue** on sama kuin aiemmassa ohjelmassa.

```
//muuttujien esittely
int aValue;
int pwmValue;
```

setup-funktio
Määritetään pinni **pwmPin** toimimaan digitaalisena lähtönä.

```
void setup() {
  pinMode(pwmPin, OUTPUT);
}
```

loop-funktio

Funktiossa tehdään kolme asiaa:

1. Luetaan potentiometriltä tulevan analogisen signaalin arvo ja tallennetaan se muuttujaan **aValue**.
2. Asetetaan muuttujalle **pwmValue** arvo muuttujan **aValue** perusteella.
3. Muodostetaan PWM-signaali muuttujan **pwmValue** arvon mukaisella pulssisuhteella.

1.
Analogisen signaalin lukeminen onkin sinulle jo tuttua aiemmista töistä, se suoritetaan funktiolla analogRead().

```
//luetaan analogisen signaalin arvo
aValue = analogRead(potPin);
```

2.

Arduino Unon A/D-muuntimen tarkkuus on 10 bittiä, joten sen tuottaman analogisen signaalin arvo on jokin luku väliltä 0-1023. PWM-signaalin pulssisuhde asetetaan kuitenkin luvulla väliltä 0-255. Käytetään funktiota map() muuttamaan lukualueella 0-1023 oleva luku lukualueelle 0-255.

```
//lukualueen muutos 0-1023 --> 0-255
pwmValue = map(aValue, 0, 1023, 0, 255);
```

Yllä oleva koodi muuttaa muuttujassa **aValue** olevan luvun alueelta 0-1023 luvuksi alueelta 0-255 ja sijoittaa sen muuttujan **pwmValue** arvoksi.

3.
PWM-signaali muodostetaan totutusti funtiolla analogWrite().

```
 //PWM-signaalin muodostus
 analogWrite(pwmPin, pwmValue);
```

Ohjelman valmis koodi:

```
//vakioiden määritys
#define potPin A0
#define pwmPin 11

//muuttujien esittely
int aValue;
int pwmValue;


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

void loop() {
 //luetaan analogisen signaalin arvo
 aValue = analogRead(potPin);
```

```
 //lukualueen muutos 0-1023 --> 0-255
 pwmValue = map(aValue,0,1023,0,255);

 //PWM-signaalin muodostus
 analogWrite(pwmPin, pwmValue);
}
```

Käännä ja lähetä ohjelma Arduinoon klikkaamalla kuvaketta **"Lähetä"**. Ohjelman suoritus käynnistyy Arduinossa automaattisesti heti latauksen valmistuttua.

Testaus

Potentiometrin säätönuppia pyörittämällä voit muuttaa tasavirta-moottorin pyörimisnopeutta.

Elektroninen noppa

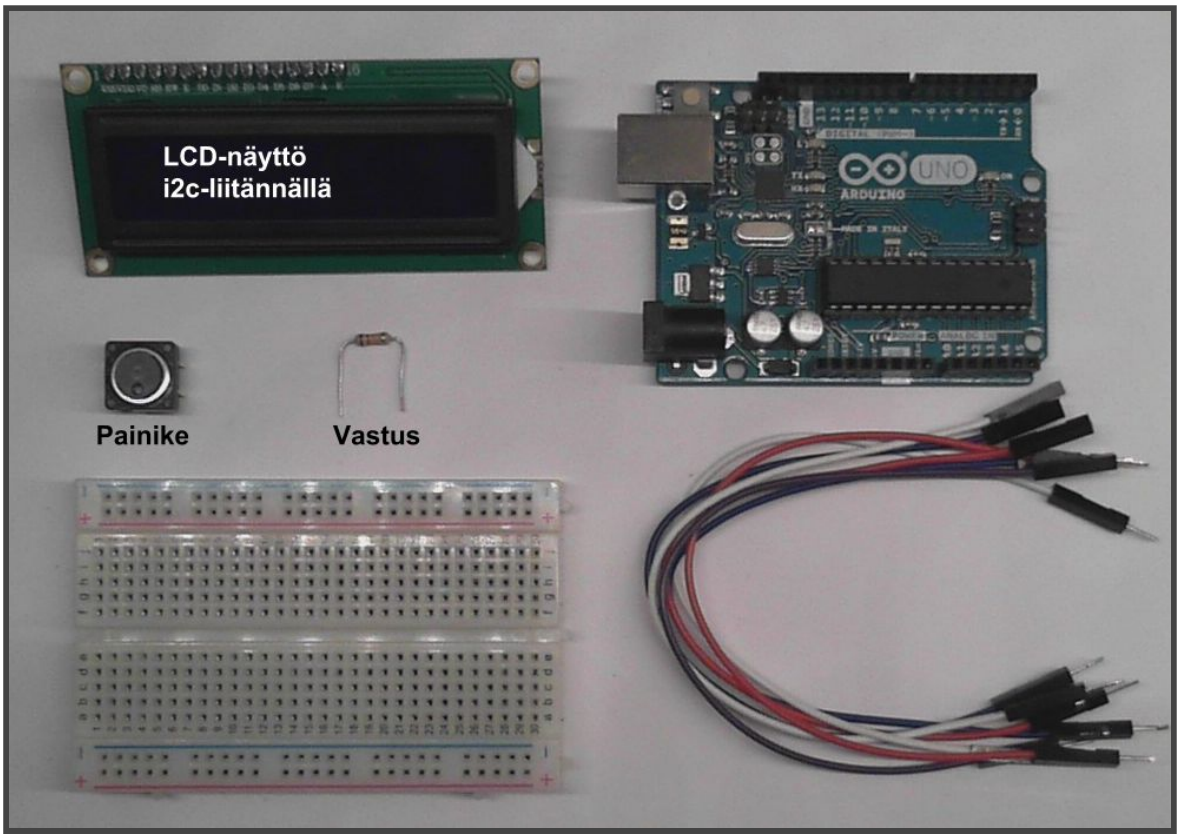
Työn kuvaus

Perinteinen heitettävä noppa on tuhansia vuosia vanha keksintö. Tässä työssä rakennetaan elektroninen versio nopasta. Uutena komponenttina perehdytään LCD-näytön käyttöön. LCD-näyttö toimii mikro-ohjaimen ulostulolaitteena, eräänlaisena monitorina. Näytöllä näytetään arvottu nopan silmäluku. Käyttämässämme näytössä on kaksi riviä ja yhdelle riville mahtuu 16 merkkiä (2x16 LCD). Näytön takana on i2c-moduuli, jonka kautta näyttö voidaan liittää Arduinoon käyttäen vain neljää johdinta.

Tarvittavat komponentit

- Arduino Uno mikro-ohjain
- Unon USB-kaapeli
- 2x16 LCD-näyttö i2c-moduulilla
- 10K ohmin vastus
- Painike

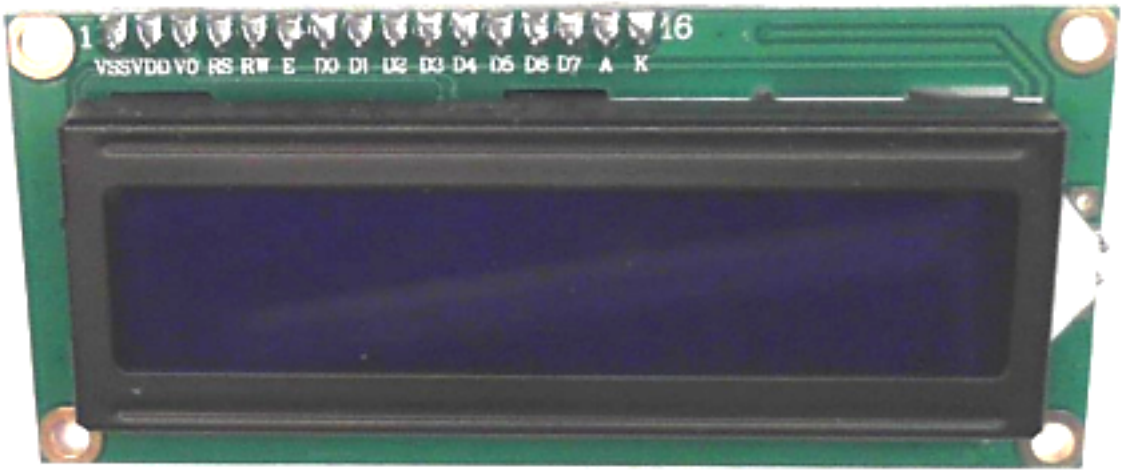
- Kytkentäalusta
- Hyppyjohtoja uros-uros ja uros-naaras



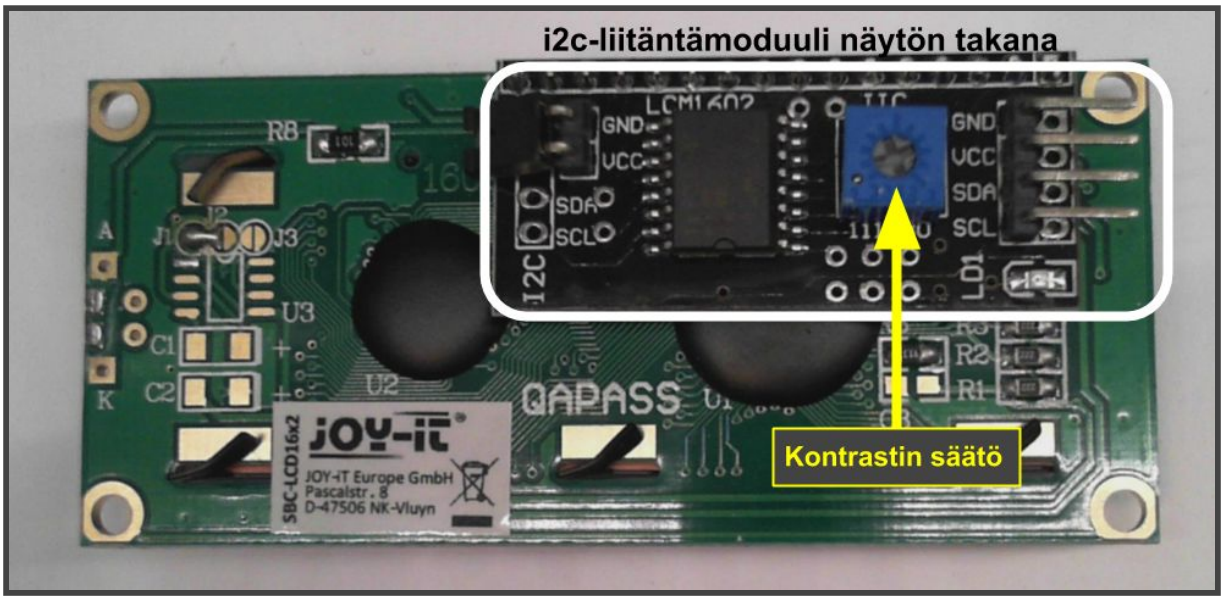
USB-kaapeli ei ole mukana kuvassa.

LCD-näyttö

Alla olevassa kuvassa on LCD-näytön etupuoli.



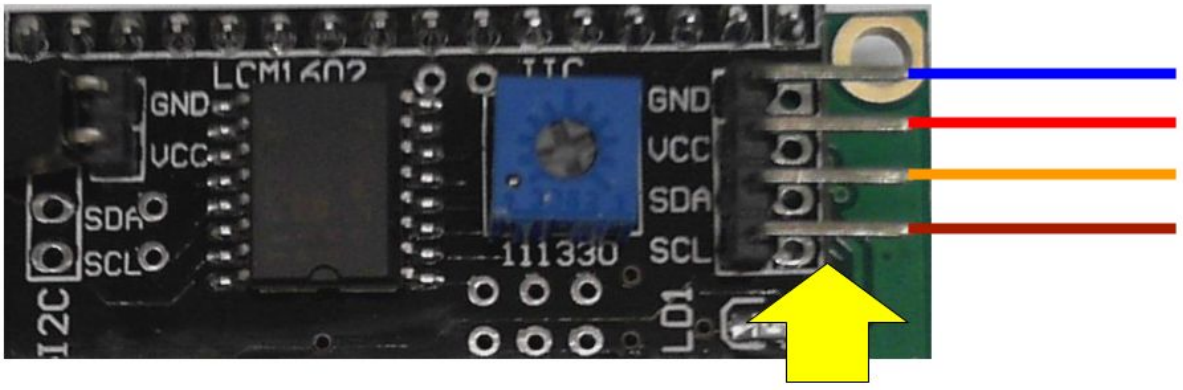
Näytön takana on i2c-liitäntämoduuli, jonka kautta näyttö voidaan liittää Arduinoon neljällä johtimella.



Jos merkit eivät näy selvästi tai ollenkaan näytössä, niin säädä näytön kontrastia takana olevasta trimmeristä. Pyöritä sitä meisselillä kohtaan, jossa merkit näkyvät selvästi.

LCD-näytön liitäntä Arduino Unoon

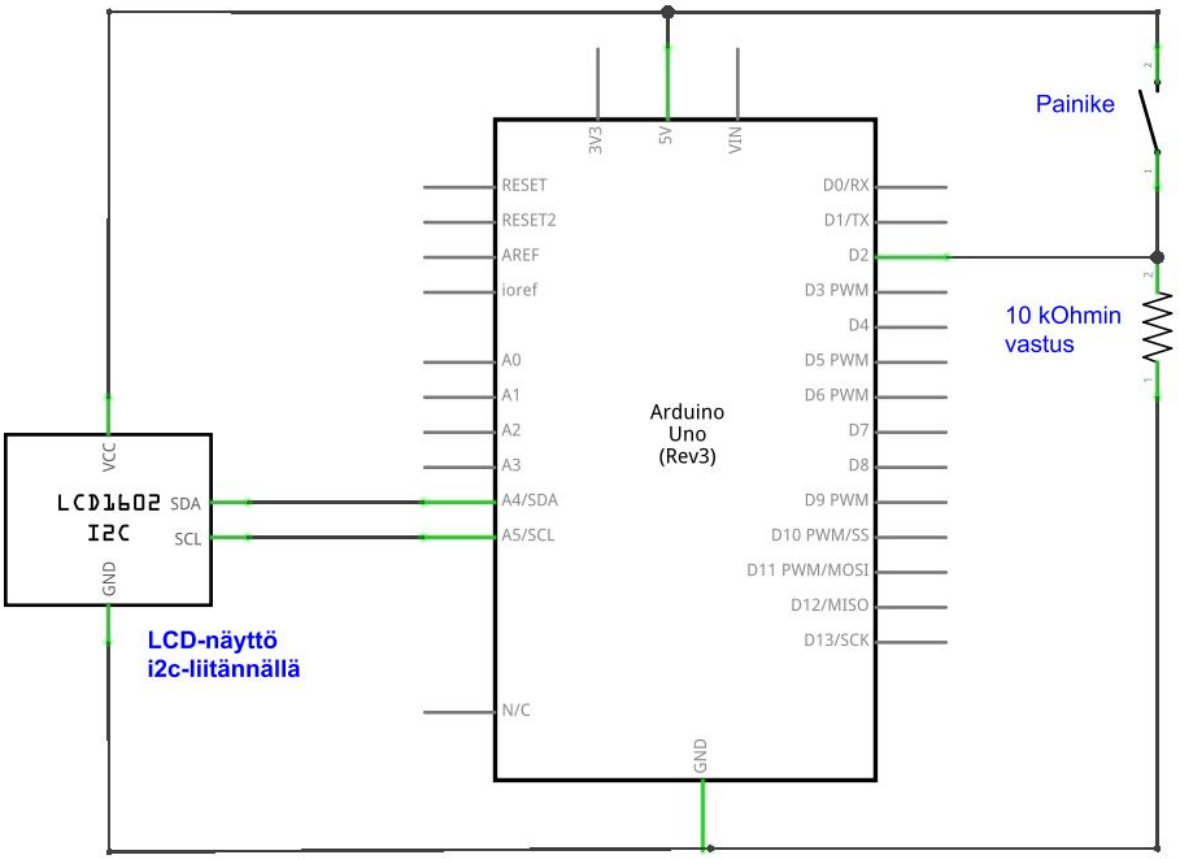
Näytön takana olevassa i2c-moduulissa on neljä pinniä. Ne liitetään Arduinoon (tarvittaessa kytkentäalustan kautta) naaras-uros -hyppyjohdoilla.



i2c-moduulin pinni	Kytkentä Arduinoon
GND	GND
VCC	+5V
SDA	A4
SCL	A5

Kytkentäkaavio

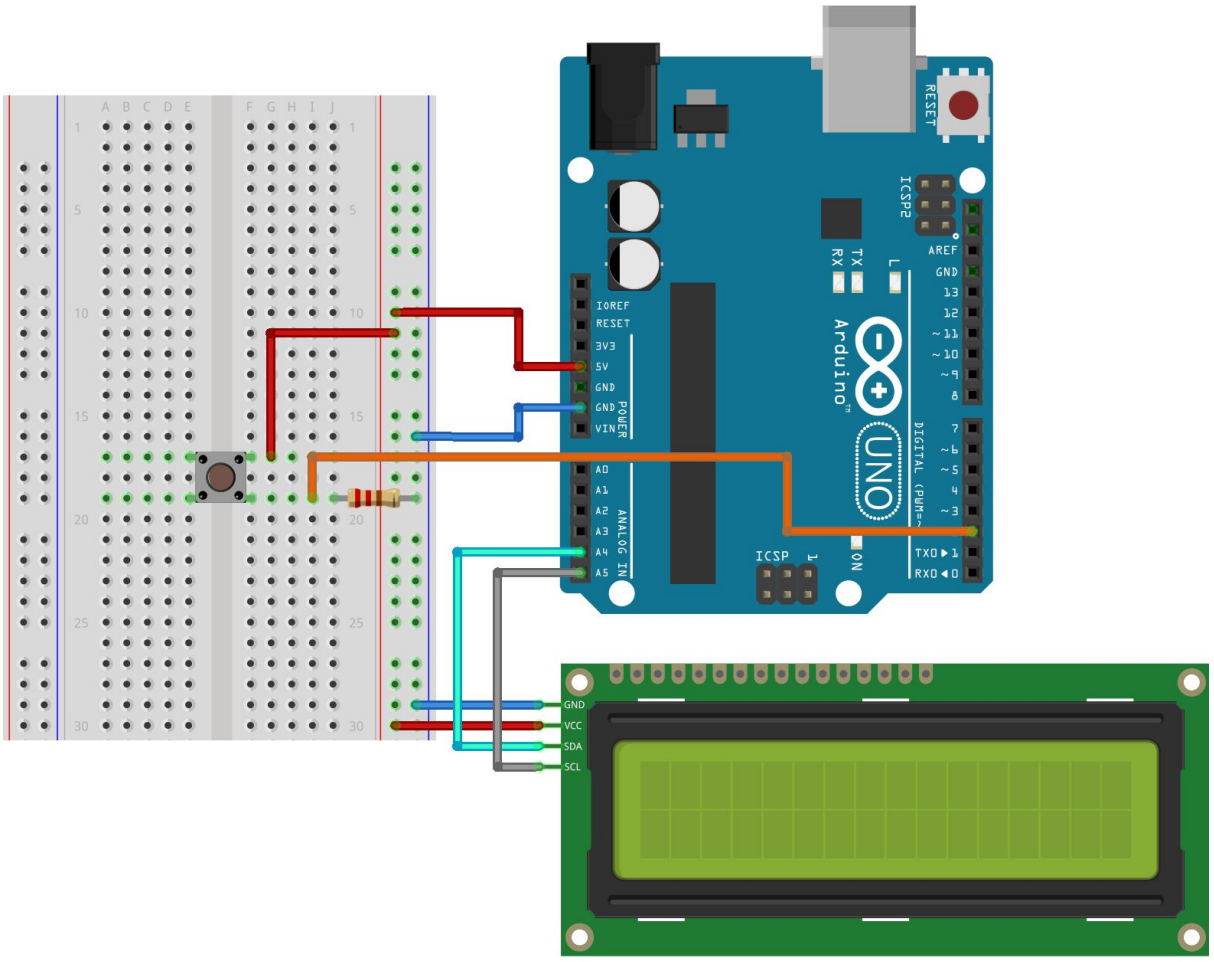
Virtapiirin rakentaminen kytkentäkaavion perusteella on sinulle jo tuttua aiemmista projekteista. Tämän vuoksi tästä työstä lähtien virtapiiristä esitetään vain kytkentäkaavio ilman yksityiskohtaisia ohjeita komponenttien liittämiseksi kytkentäalustalle.



Painike ja vastus ovat tuttuja komponentteja aiemmista projekteista. Ainoa uusi komponentti kytkennässä on LCD-näyttö. Arduino Unossa on vain yksi 5 Voltin jännitelähtö. 5 Voltin jännite tulee kuitenkin kytkeä sekä LCD-näytölle että painikkeelle. Tämä kannattaa toteuttaa näin:

- Kytke uros-uros hyppyjohto Arduinon **pinniin 5V** ja liitä johdon toinen pää kytkentäalustalle **punaisella viivalla merkittyyn pystysuoraan pisteriviin**.
- Kytke pystysuoran rivin johonkin pisteeseen hyppyjohdon urosliitin ja liitä johdon toisessa päässä oleva naarasliitin **LCD-näytön pinniin VCC**.
- Pystysuorasta rivivistä voit vetää 5 Voltin jännitteen myös painikkeelle.

Kytkentä



Ohjelmointi

Aloitus
Käynnistä Arduinon ohjelmointiympäristö työpöydällä olevasta pikakuvakkeesta.

Tee tarvittaessa uusi tyhjä projekti: **Tiedosto / Uusi**.

Ulkoisten kirjastojen tuonti
Voimme tehdä LCD-näytön ohjelmoinnista hyvin helppoa käyttämällä ulkoisen kirjaston valmiita funktioita. Ulkoiset kirjastot tuodaan käyttöön komennolla #include.

```
#include <LiquidCrystal_I2C.h>
```

Vakiot
Määritetään vakio **btnHeitto**, jolla viittaamme painiketta lukevaan pinniin **D2**.

```
#define btnHeitto 2
```

Muuttujat
Ohjelmassa käytetään kahta integer-tyyppistä muuttujaa sekä merkeistä muodostuvaa taulukkoa (Array).

```
int noppa;
int i;
//taulukon esittely ja merkkien sijoitus taulukkoon
char merkki[9] = "|/-\|/-\|*";
```

Olion muodostus
Muodostetaan luokasta **LiquidCrystal_I2C** olio **lcd**. Tämän jälkeen voimme ohjata LCD-näyttöä ko. luokan metodeilla (olio-ohjelmoinnissa funktion nimi on metodi). Attrivuuttina on i2c-moduulin osoite ja LCD-näytön koko.

```
LiquidCrystal_I2C lcd(0x27, 16, 2);
```

setup-funktio
Alustetaan LCD-näyttö metodilla `lcd.begin();` ja kytketään näytön taustavalo päälle metodilla `lcd.backlight();`.

Asetetaan pinni **btnHeitto** toimimaan digitaalisena tulona. Tämän jälkeen alustetaan satunnaislukugeneraattori analogisen tulon **A0** kohinalla (satunnainen arvo) käyttäen komentoa randomSeed().

```
void setup() {
  lcd.begin();
  lcd.backlight();
  pinMode(btnHeitto, INPUT);
  randomSeed(analogRead(0));
}
```

Oma alku()-funktio
Tehdään oma alku-niminen funktio, joka suoritetaan aina ennen nopan heittoa. Funktiossa tyhjätään LCD-näyttö ja tulostetaan näytölle aloitustekstit.

```
void alku() {
  lcd.clear();
  lcd.print(" *  N O P P A * ");
  lcd.setCursor(0,1);
  lcd.print("  Paina heitto");
}
```

loop-funktio
Aluksi loop-funktiossa suoritetaan oma funktio **alku()**. Tämän jälkeen odotetaan, että käyttäjä painaa painiketta **heitto**.

Painikkeen painalluksen jälkeen näyttöön tulee teksti **"Noppa pyörii"**. Tekstin alla esitetään merkeillä muodostettu animaatio. Animaation muodostuksessa käytetään for-toistorakennetta sekä **merkki-taulukkoa**.

Lopuksi arvotaan muuttujaan **noppa** satunnaisluku väliltä 1-6 ja esitetään luku näytöllä kahden sekunnin ajan. Tämän jälkeen loop-funktio suoritetaan uudelleen alusta alkaen.

```
void loop() {
  alku();
  while (digitalRead(btnHeitto) == LOW) {
    delay(1);
  }
  lcd.clear();
  lcd.print("Noppa pyorii");
  for (i=0; i<8; i++)
  {
    if (i > 0) {
      lcd.setCursor(i-1,1);
      lcd.print(" ");
    }
    lcd.print(merkki[i]);
    delay(200);
  }
  delay(200);
  noppa = random(1,7);
  lcd.clear();
  lcd.print("   T U L O S");
  lcd.setCursor(7,1);
  lcd.print(noppa);
  delay(2000);
}
```

Valmis koodi

```
//ulkoisen kirjaston tuonti
#include <LiquidCrystal_I2C.h>

//vakion määritys
#define btnHeitto 2

//kokonaislukumuuttujat ja merkki-taulukko
int noppa;
int i;
char merkki[9] = "|/-\|/-\|*";

//olion muodostus luokasta
LiquidCrystal_I2C lcd(0x27, 16, 2);

void setup() {
  //näytön käynnistys ja taustavalo päälle
  lcd.begin();
  lcd.backlight();
  pinMode(btnHeitto, INPUT);
  //satunnaislukugeneraattorin alustus
  randomSeed(analogRead(0));
}

void alku() {
  //näytön tyhjäys
  lcd.clear();
  //tulostus näytölle
  lcd.print(" *  N O P P A * ");
  //kursorin siirto toiselle riville
  lcd.setCursor(0,1);
  lcd.print("  Paina heitto");
}

void loop() {
  alku();
  //odotetaan että painiketta painetaan
  while (digitalRead(btnHeitto) == LOW) {
    delay(1);
```

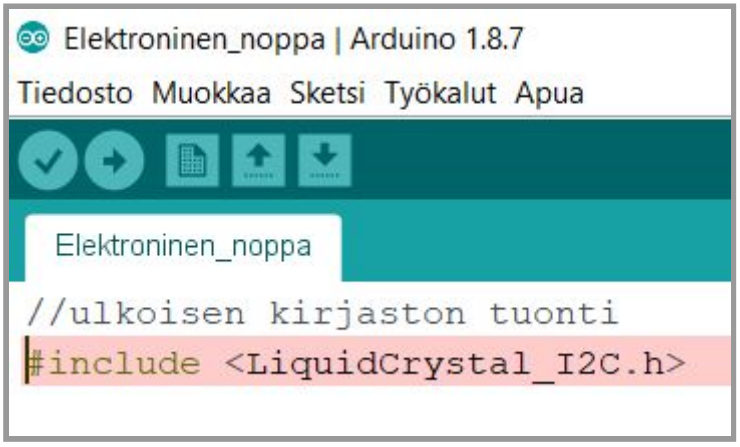
```
  }
  lcd.clear();
  lcd.print("Noppa pyorii");
  //animaatio for-toistorakenteella ja merkki-taulukolla
  for (i=0; i<8; i++)
  {
    if (i > 0) {
      lcd.setCursor(i-1,1);
      lcd.print(" ");
    }
    lcd.print(merkki[i]);
    delay(200);
  }
  delay(200);

  //satunnaisluvun arvonta väliltä 1-6
  noppa = random(1,7);

  lcd.clear();
  lcd.print("   T U L O S");
  lcd.setCursor(7,1);
  lcd.print(noppa);
  delay(2000);
}
```

Kokeile kääntää ohjelma. Jos kääntäminen ei onnistu, niin katso seuraavasta kappaleesta johtuuko virhe puuttuvasta kirjastosta.

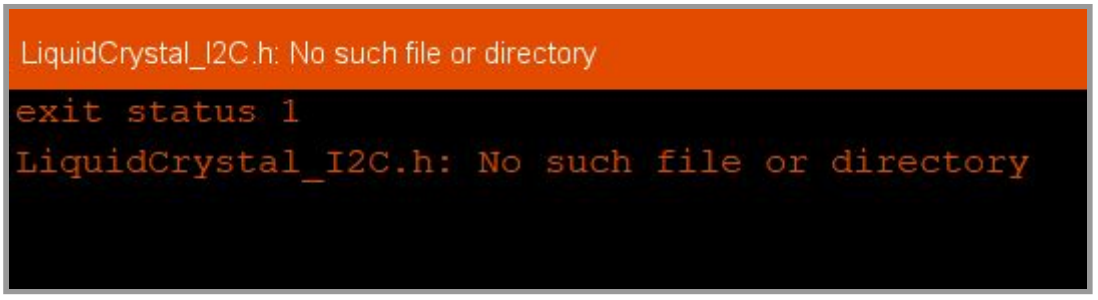
Ulkoisen kirjaston asennus

Jos käyttämääsi ohjelmointiympäristöön ei ole vielä asennettu LCD-näytön käyttöön tarvittavaa kirjastoa, tulee kääntämisen yhteydessä ao. kuvien mukainen virheilmoitus.

Virheellinen rivi:



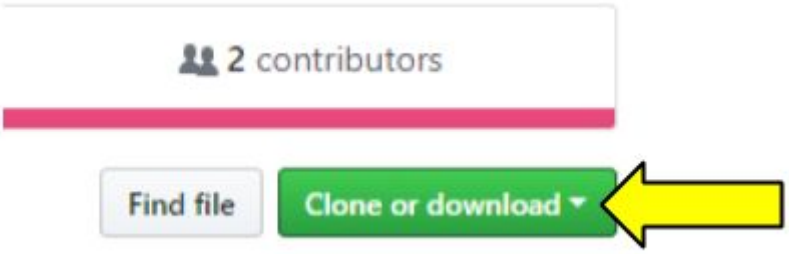
Ja virheilmoitus sivun vasemmassa alareunassa:



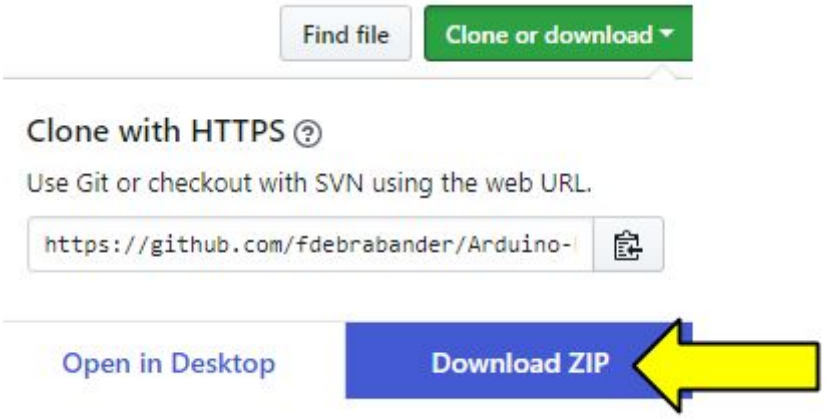
Hae ja asenna kirjasto näin:

1. Siirry linkistä osoitteeseen https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

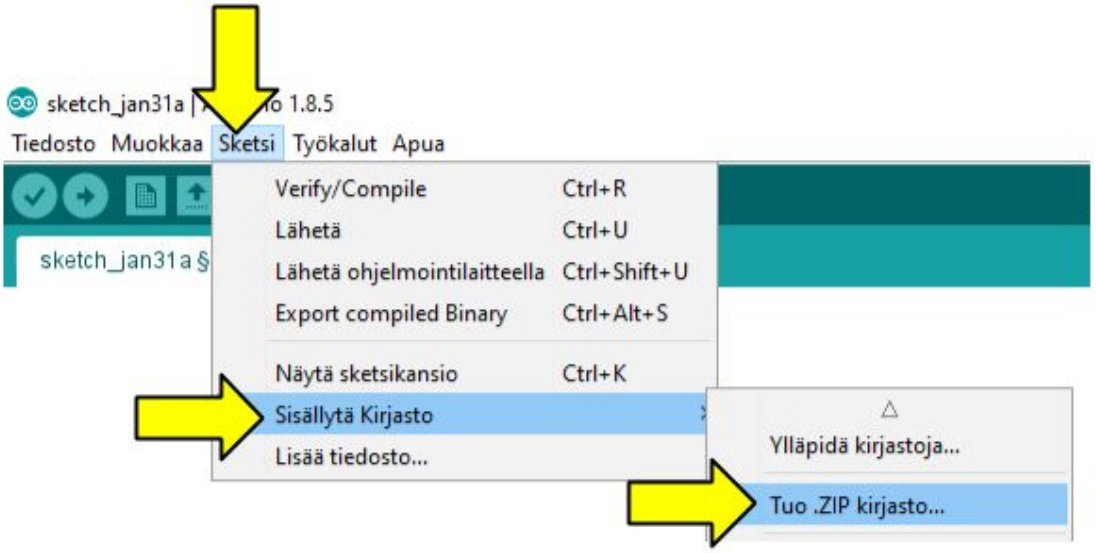
2. Klikkaa sivulta löytyvää painiketta **"Clone or download"**.



3. Klikkaa **"Download ZIP"**.

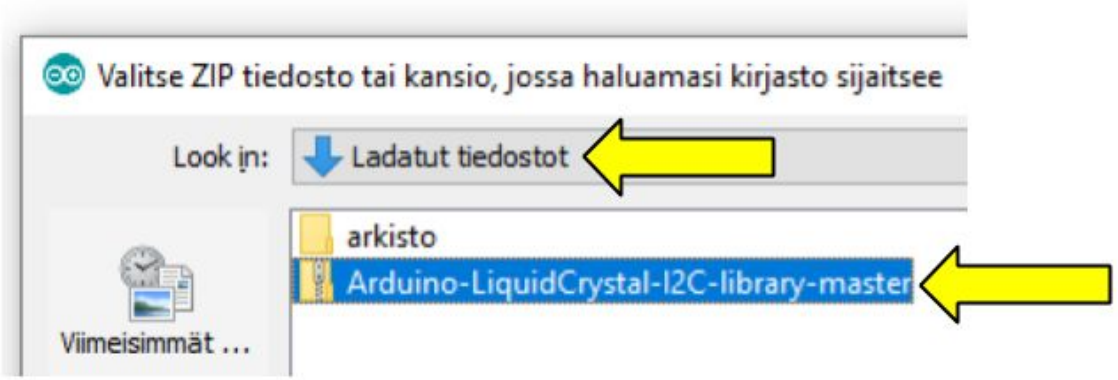


4. ZIP-tiedosto latautuu kansioon “Ladatut tiedostot”. Siirry takaisin ohjelmointiympäristöön ja klikkaa valikkoa “**Sketsi**”. Valitse ensin “**Sisällytä Kirjasto**” ja sitten valinta “**Tuo .ZIP Kirjasto...**”



5. Siirry kansioon “**Ladatut tiedostot**”, valitse tiedosto “**Arduino-LiquidCrystal-I2C-library-master**” ja klikkaa painiketta “**Open**”.

Tiedoston valinta:



Ja avaus:

Kirjasto on nyt lisätty. Samalla tavalla voit lisätä myös muita ulkoisia kirjastoja ohjelmointiympäristöön ja käyttää niiden funktioita omissa ohjelmissasi.

Tämän jälkeen puuttuva kirjasto ei aiheuta enää virheilmoituksia. Kokeile kääntää ohjelma uudelleen. Korjaa tarvittaessa virheet ja onnistuneen kääntämisen jälkeen lähetä ohjelma Arduinoon.

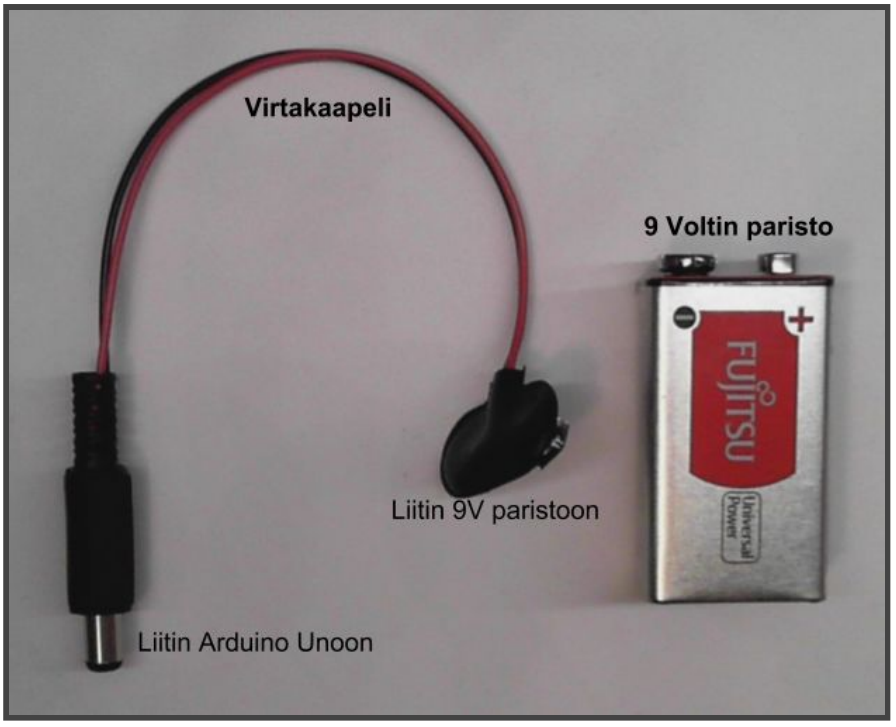
Testaus

Katso video nopan testauksesta alla olevasta linkistä.

Linkki: https://youtu.be/Xmqbe3TDCMc

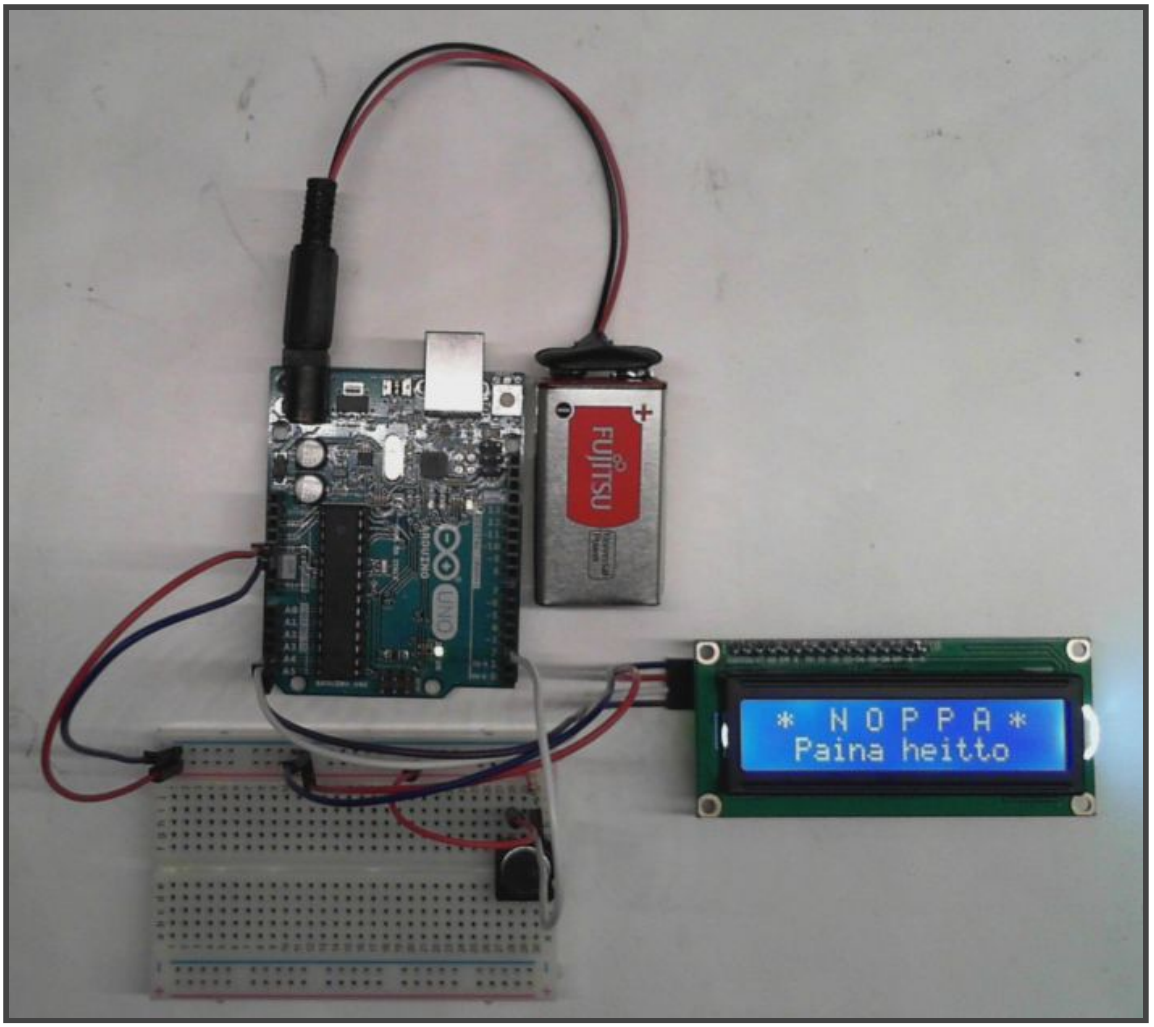
Käyttö ilman tietokonetta

Ohjelmoinnin jälkeen Arduino ei enää tarvitse tietokonetta toimiakseen, mutta virtaa se sen sijaan tarvitsee. Virransyötön voimme tehdä liittämällä 9 Voltin pariston sopivalla liitosjohdolla kiinni Arduino Unon virtaliittimeen.



Vaihtoehtoisesti 9 Voltin pariston **plus-navan** voi kytkeä myös Arduinon **pinniin VIN** ja pariston **miinus-navan pinniin GND**.

Paristolla toimiva elektroninen noppa:



Ventti-peli

Työn kuvaus

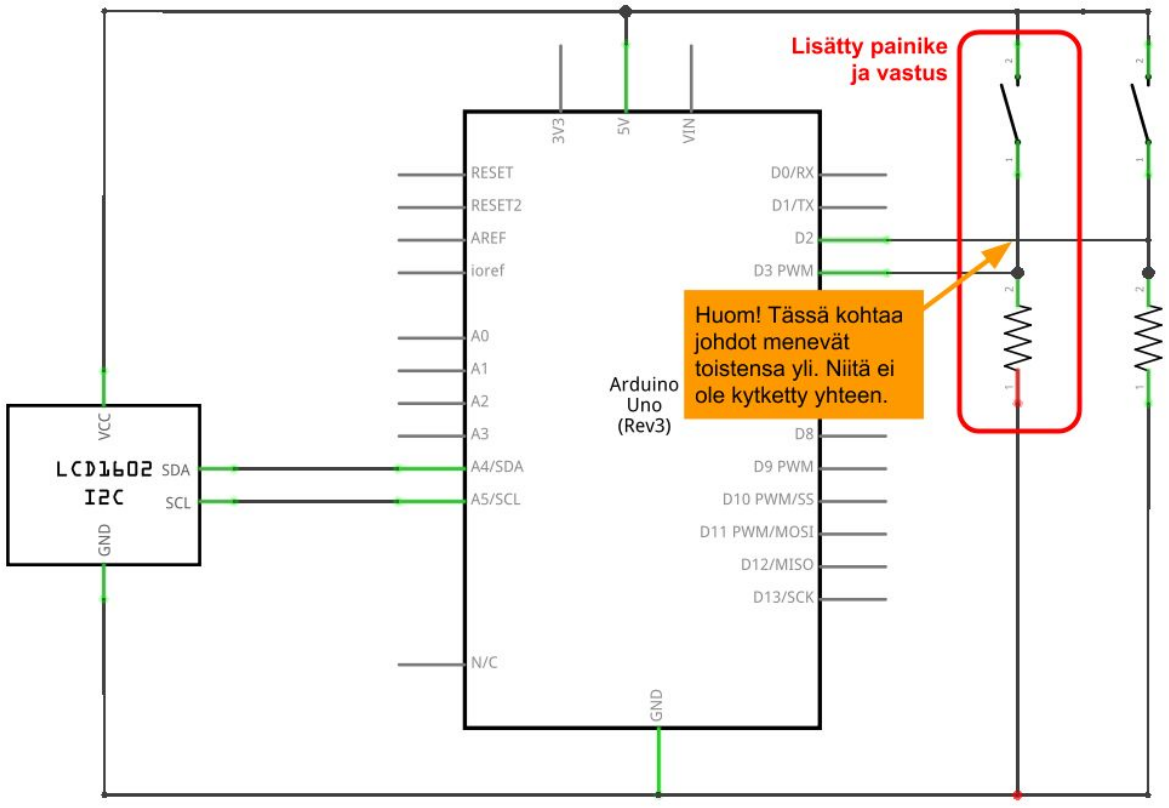
Ventti-peli on elektronisen nopan jatkotyö. Aluksi sinulla tulee olla elektronisen nopan kytkentä valmiina. Tässä ohjeessa kuvataan kytkentään tarvittavat lisäykset. Pelissä on melko paljon koodia, joten se annetaan valmiina tiedostona.

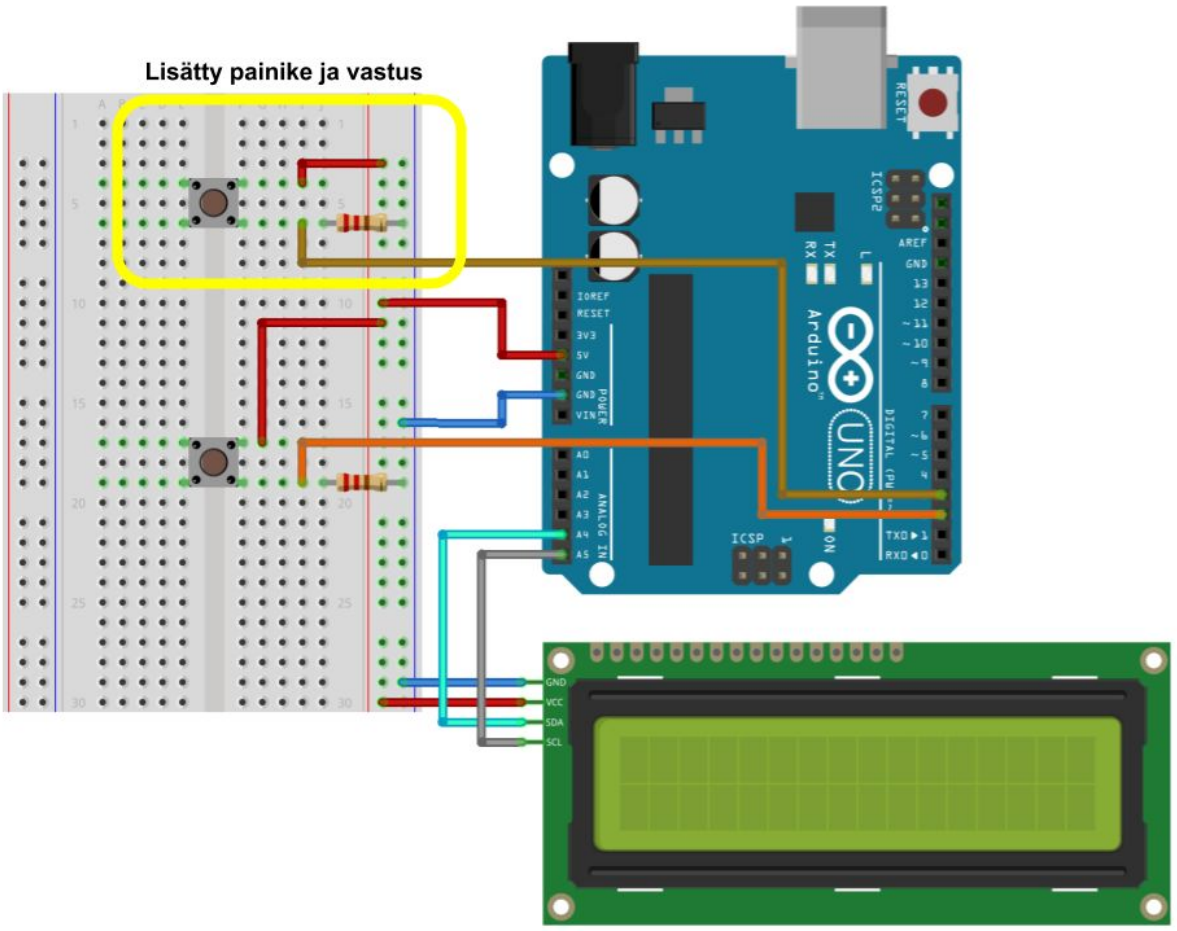
Tarvittavat lisäkomponentit

- 10K ohmin vastus
- Painike
- Hyppyjohtoja

Kytkentäkaavio

Ventin pelaamiseen tarvitaan kaksi painiketta, joten lisäämme kytkentään toisen painikkeen ja vastuksen. Kytkennässä on valmiina **pinniin D2** kytketty painike. Toinen painike lisätään **pinniin D3** oheisen kytkentäkaavion mukaisesti.





Pelin pelaaminen

Peliä pelataan parin kanssa ja tarkoituksena on päästä lukuun 21 tai mahdollisimman lähelle sitä. Jos luku menee yli 21 kyseinen pelaaja häviää pelin välittömästi. Luku 21 on ventti ja jos pelaaja saa sen, hän voittaa pelin välittömästi. Jos molemmat pelaajat pääsevät pienempään lukuun kuin 21, niin voittaja on lähimmäksi päässyt. Jos luvut ovat yhtäsuuret, niin silloin tulos on tasapeli.

Perinteistä Venttiä pelataa korteilla, joiden arvo on 1-14. Tässä pelissä ässä-kortin arvo on aina 14, joten korttien arvot ovat väliltä 2-14. **Pinniin D2** liitetty painike on nimeltään **MORE**. Sitä painamalla pelaaja ottaa uuden kortin. **Pinniin D3** liitetty painike on nimeltään **STAY**. Tällä painikkeella pelaaja päättää, ettei ota enää uutta korttia ja jää saamaansa lukuun.

Ohjelmointi ja testaus

Tällä kertaa ohjelman lähdekoodi annetaan valmiina tiedostona. Toimi näin:

1. Lataa pelin pakattu lähdekoodi omalle tietokoneellesi.
2. Pura ZIP-paketti.
3. Siirry kansioon **ventti-peli** ja avaa siellä oleva tiedosto ohjelmointiympäristöön
4. Käännä ohjelma ja lähetä se Arduinoon.

Peli on valmis pelattavaksi. Muista että **pinnissä D2 on MORE-painike** ja **pinnissä D3 painike STAY**.

Katso video pelin testauksesta. Videolla näet myös, miten 9 Voltin paristo on liitetty suoraan Arduinon **VIN (+)** ja **GND (-) pinneihin**.

Sääasema

Työn kuvaus

Rakennetaan ja ohjelmoidaan sisäkäyttöön tarkoitettu sääasema, joka mittaa ja esittää kolme säähän liittyvää suuretta:

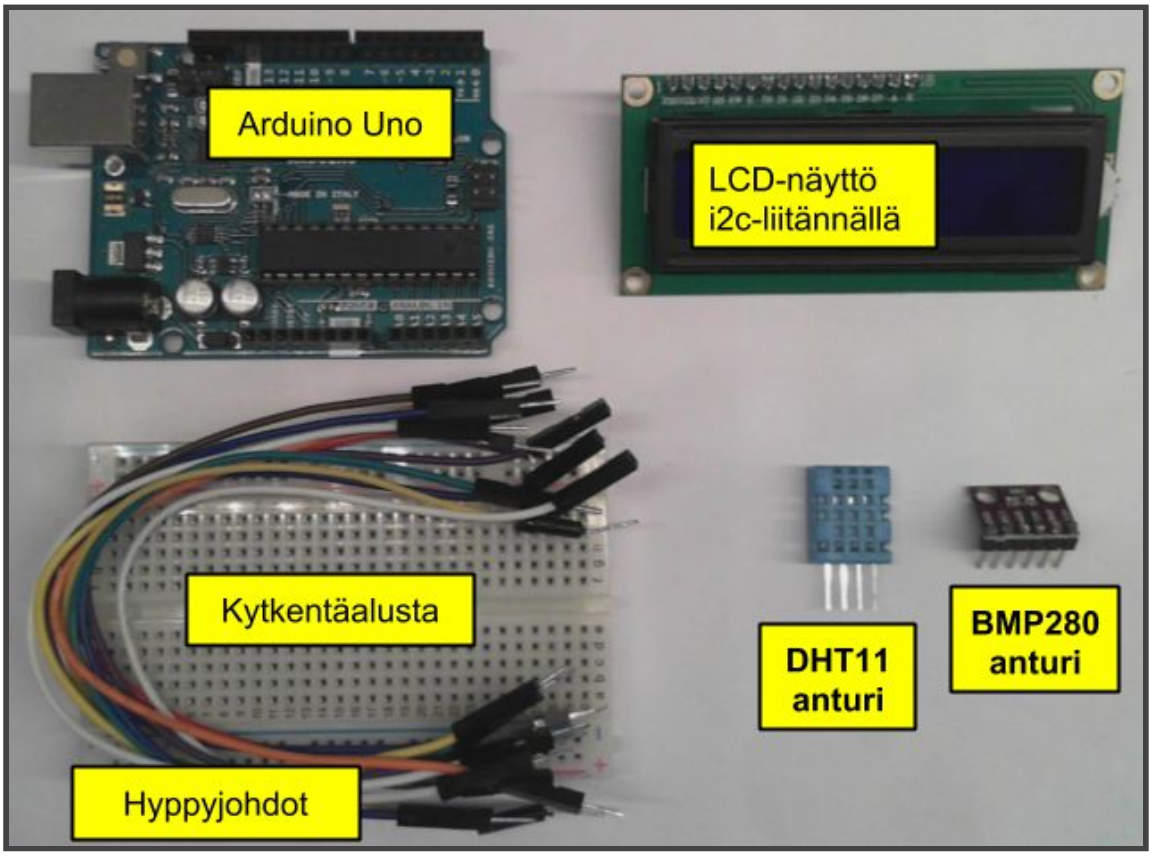
- Lämpötilaa (C)
- Ilman suhteellista kosteutta (%)
- Ilmanpainetta (hPa)

Ilmanpaineen muutoksen (nousu / lasku) perusteella voidaan tehdä myös sääennusteita. "*Karkeasti ottaen, laskeva paine kertoo matalapaineen lähestymisestä, tuulen voimistumisesta ja sään huononemisesta, kun taas paineen noustessa on odotettavissa tyynempää poutasäätä.*" (Wikipedia).

Suureiden mittaamiseen käytetään kahta anturia. DHT11-anturilla mitataan ilmankosteus prosentteina. DHT11 kykenee mittaamaan myös lämpötilaa, mutta mittaamme sen toisella tarkemmalla anturilla. Lämpötila ja ilmanpaine mitataan Boschin valmistamalla BMP280 -anturilla. Antureiden lukemiseen käytetään ulkoisia koodikirjastoja, jotka vähentävät itse kirjoitettavan koodin määrän hyvin minimaaliselle tasolle. Mittausten jälkeen tulokset näytetään LCD-näytöllä.

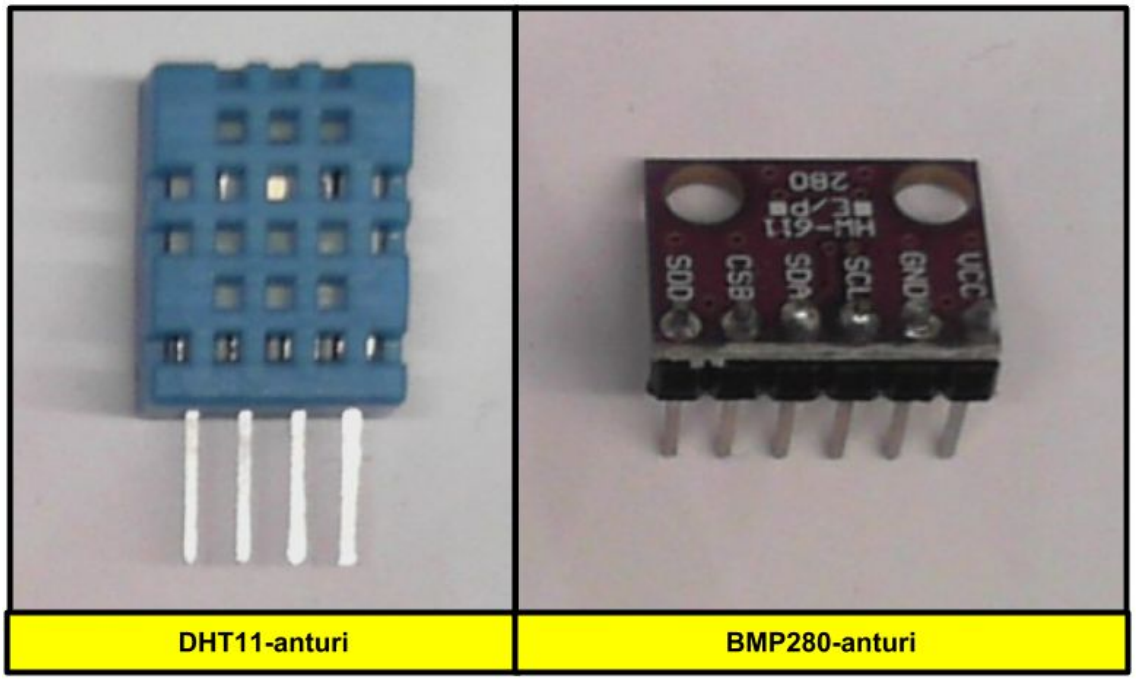
Tarvittavat komponentit

- Arduino Uno
- USB-kaapeli
- LCD-näyttö i2c liitännällä
- DHT11-anturimoduuli
- BMP280-anturimoduuli
- Kytkentäalusta
- Hyppyjohtoja (uros-uros, naaras-uros)

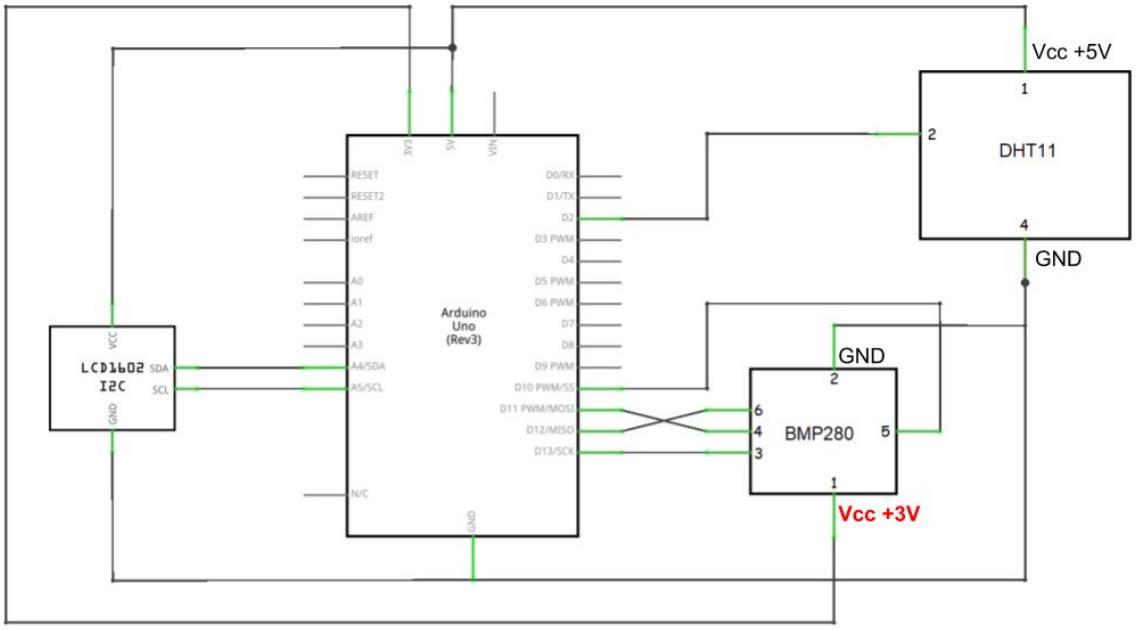


USB-kaapeli ei ole kuvassa mukana.

Anturimoduulit:

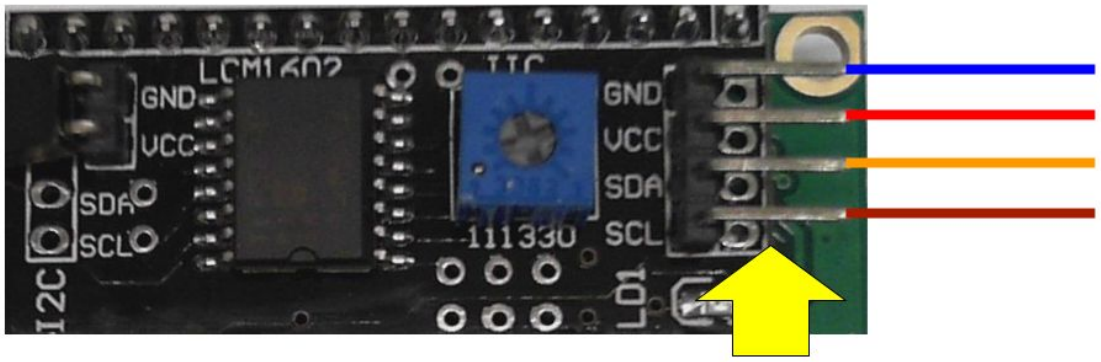


Kytkentäkaavio



LCD-näytön kytkentä Arduino Unoon

Näytön takana olevassa i2c-moduulissa on neljä pinniä. Ne liitetään Arduinoon (tarvittaessa kytkentäalustan kautta) naaras-uros -hyppyjohdoilla.

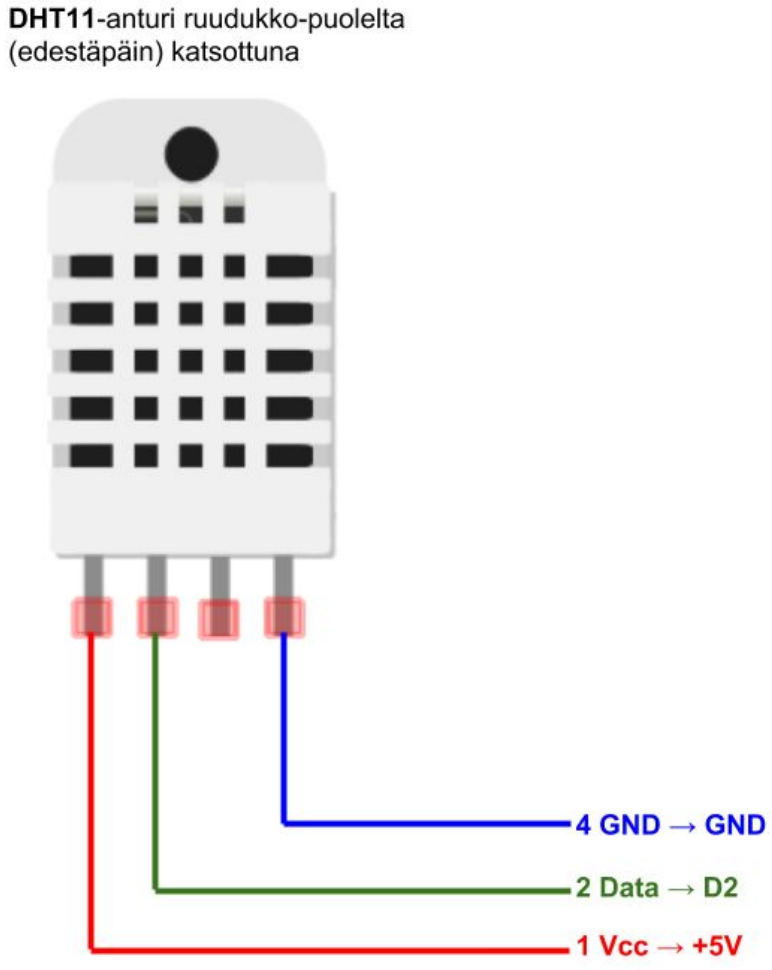


i2c-moduulin pinni	Kytkentä Arduinoon
GND	GND
VCC	+5V
SDA	A4
SCL	A5

DHT11-anturin kytkentä Arduino Unoon

Anturissa on neljä pinniä, joista kolmatta pinniä emme käytä ollenkaan. Voit kytkeä anturin kytkentäalustalle esimerkiksi sarakkeeseen A pisteisiin 1, 2, 3, ja 4. Taulukosta näet miten anturi liitetään Arduinoon.

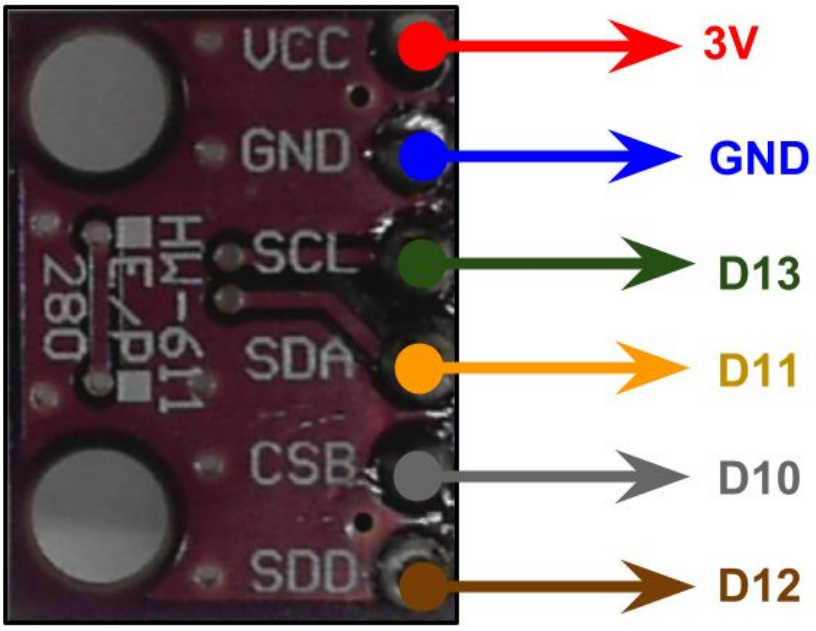
DHT11 pinni	Kytkentä Arduinoon
1 Vcc	+5V
2 Data	D2
3 ei käytössä	
4 GND	GND



BMP280-anturin kytkenä Arduino Unoon

Anturissa on kuusi pinniä ja voit liittää sen kytkentäalustalle esim. sarakkeeseen A riveille 25-30. Taulukosta näet miten anturi kytketään Arduinoon. Huomaa että anturin käyttöjännite Vcc on kolme volttia.

BMP280 pinni	Kytkentä Arduinoon
1 VCC	+3 Volttia
2 GND	GND
3 SCL	D13
4 SDA	D11
5 CSB	D10
6 SDD	D12



Ohjelmointi

Ulkoisten kirjastojen tuonti, vakioiden määritys ja olioiden muodostus

```
//ulkoisten kirjastojen tuonti
#include <Wire.h>
#include <LiquidCrystal_I2C.h>
#include <DHT.h>
#include <Adafruit_BMP280.h>

//DHT11-anturin datapinni ja tyyppi
#define DHTPIN 2
#define DHTTYPE DHT11

//BMP280-anturin liitäntäpinnit
#define BMP_SCK  (13)
#define BMP_MISO (12)
#define BMP_MOSI (11)
#define BMP_CS   (10)

//olioiden muodostus luokista
LiquidCrystal_I2C lcd(0x27, 16, 2);
DHT dht(DHTPIN, DHTTYPE);
Adafruit_BMP280 bmp(BMP_CS, BMP_MOSI, BMP_MISO,  BMP_SCK);
```

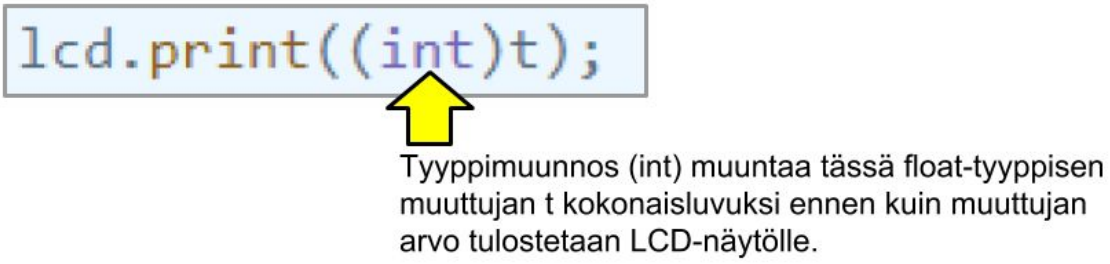
setup-funktio

```
void setup() {
  //lcd-näytön käynnistys ja taustavalo päälle
  lcd.begin();
  lcd.backlight();
  //DHT11-anturin käynnistys
  dht.begin();
  //BMP280-anturin käynnistys, virheilmoitus jos ei löydy
  if (!bmp.begin()) {
    lcd.print("BMP280 ei loydy");
    while (1);
  }
}
```

loop-funktio

```
void loop() {
  //luetaan ilmankosteus DHT11-anturilta muuttujaan h
  float h = dht.readHumidity();

  //luetaan lämpötila BMP280-anturilta muuttujaan t
  float t = bmp.readTemperature();

  //luetaan ilmanpaine hehto Pascaleina
  //BMP280-anturista muuttujaan p
  float p = bmp.readPressure() / 100;

  //LCD-näytön tyhjäys
  lcd.clear();

  //tulostetaan lämpötila LCD-näytölle
  lcd.setCursor(0,0);
  lcd.print("Lampo:");
  lcd.print((int)t);

  //tulostetaan kosteus LCD-näytölle
  lcd.print(" Kos:");
  lcd.print((int)h);
  lcd.print("%");

  //tulostetaan ilmanpaine LCD-näytölle
  lcd.setCursor(0,1);
  lcd.print("Paine:");
  lcd.print((int)p);
  lcd.print(" hPa");

  delay(1000);
}
```

Antureilta luetut suureet ovat desimaalimuodossa (float). LCD-näytölle ne tulostetaan kuitenkin kokonaislukuina. Tyyppimuunnos suoritetaan näytölle tulostuksen yhteydessä



Ulkoisten kirjastojen haku ja asennus

Antureiden lukemiseen käytetään valmiita ulkoisia kirjastoja. Jos koodin kääntäminen pysähtyy kirjaston tuonnin kohdalla (**#include <kirjastonnimi.h>**) virheilmoitukseen **"No such file or directory"**, niin ohjelmointiympäristöön ei ole vielä asennettu tarvittavia kirjastoja.

Katso työohjeesta Elektroninen noppa sivulta 11 miten ulkoiset kirjastot asennetaan. Alla olevat linkit korvaavat ohjeessa olevan kohdan 1. Lataa ja asenna alla olevista linkeistä löytyvät koodikirjastot ohjeen mukaisesti.

Koodikirjasto #1:
https://github.com/adafruit/DHT-sensor-library

Koodikirjasto #2:
https://github.com/adafruit/Adafruit_Sensor

Koodikirjasto #3:
https://github.com/adafruit/Adafruit_BMP280_Library

Jos LCD-näytön käyttämiseen tarvittavaa koodikirjasto ei ole aiemmin asennettu ohjelmointiympäristöön, niin asenna myös se ao. linkistä.

LCD-näytön koodikirjasto:
https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

Kirjastojen asennuksen jälkeen kääntäminen ei enää pysähdy puuttuvan kirjaston kohdalle.

Testaus

Kun ohjelman kääntäminen menee läpi ilman virheitä, niin lähetä ohjelma mikro-ohjaimeen. Lähetyksen jälkeen ohjelman suoritus käynnistyy ja näet LCD-näytöllä mitatun lämpötilan, kosteuden sekä ilmanpaineen.

Lämmitä BMP280-anturia laittamalla sormi sen päälle, lämpötila alkaa nousta ja kun otat sormen pois, lämpötila laskee. Hengitä DHT11-anturia kohti ja uloshengitysilman vesihöyry nostaa kosteusprosenttia.

Ilman suhteellinen kosteus on prosenttiluku, joka ilmaisee, kuinka paljon ilmassa on vesihöyryä siihen nähden, mitä kyseisessä lämpötilassa voi olla enimmillään vesihöyrynä (Ilmatieteenlaitos: ilmankosteus). Ilmanpaine on ilmakehän paine maanpintaa vastaan. Normaali ilmanpaine on 1013 hehtopascalia (ilmatieteenlaitos: ilmanpaine).