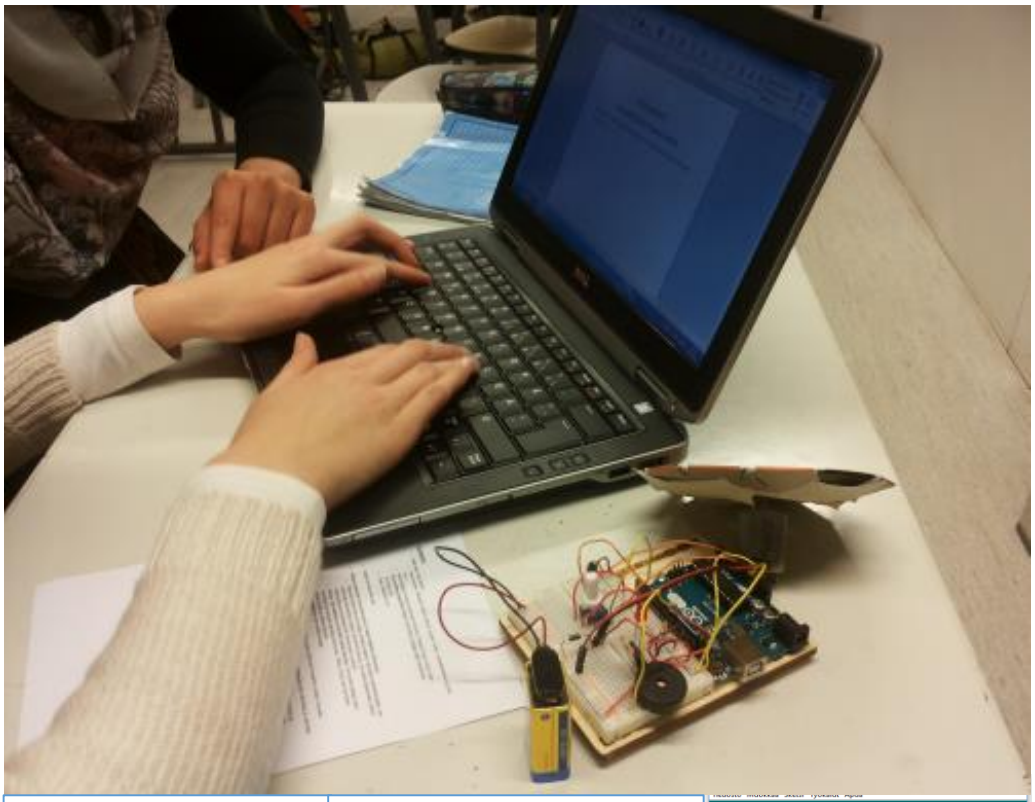
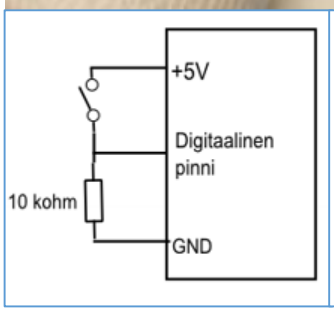
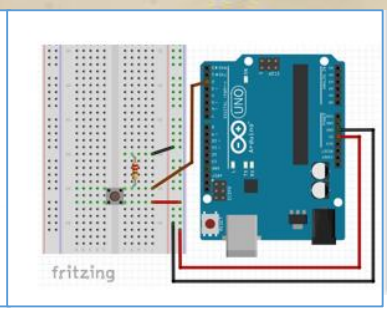
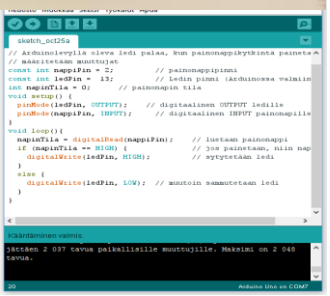
Arduino tutuksi

Kyösti Blinnikka









Arduino tutuksi

Tämän oppaan työt tehtyäsi sinulla on perustiedot ja -taidot

- tehdä ledivilkkusysteemejä Arduinolla
- kytkeä painonappeja Arduinoon
- mitata jännitettä Arduinolla (→ pariston jännitteen mittaus, lämpötilan mittaus, valaistuksen mittaus)
- tuottaa ääntä Arduinolla
- ohjata servoa
- ohjata sähkömoottoria
- säätää tehoa (PWM → ledin himmentäminen/moottorin kierrosluvun säätäminen)
- mitata etäisyyksiä Arduinoon liitetyllä ultraäänianturilla
- liittää LCD-näyttö Arduinoon

Näitä ”palikoita” voit käyttää monien laitteiden suunnitteluun ja kokeiluun. Samalla kun kokeilet erilaisia kytkentöjä ja ohjelmia, opit paljon sähköisten laitteiden toiminnasta

Materiaali soveltuu erityisesti yläkoulun tai lukion Arduino- tai ohjelmointikurssien aineistoksi. Alun tutkimustehtävät (ledin vilkuttelu, painonapin lukeminen, jännitteen mittaaminen) on syytä käydä aluksi huolella läpi. Muutoin tehtävien järjestys on vapaa.

* * * *

Sisällys

Arduino 5
Arduinon osat 5
Koekytkentälevy 6
Ohjelmointiympäristö 7
Ensimmäinen ohjelma BLINK 8
LEDin vilkuttelu, liikennevalot 10
RGB-ledi 16
Ohjelman rakenteita, FOR-silmukka 19
Painonapin lukeminen 21
Jännitteen mittaaminen – analoginen input 26
Jännitteenjakaja – potentiometrin lukeman lukeminen 30
Äänen tuottaminen 32
Lämpötilan mittaaminen 36
Valaistuksen mittaaminen 38
Servon ohjaaminen Arduinolla 40
PWM Pulse Width Modulation 42
Moottorin ohjaaminen Arduinolla 46
Ultraäänianturin liittäminen Arduinoon 49
LCD-näytön liittäminen Arduinoon 51
Oman funktion tekeminen 53
Oman laitteen tekeminen – projektityö 55
Kertausta: peruskytkennät 56
Ohjelman ohjausrakenteita 57
Toistolauseet 58
Vastusten värikoodi 59
Projektityö 61
Harjoitustyöt 62
Tutkimuksissa tarvittavat välineet ja komponentit 63

Arduino

”**Open Source Electronics Prototyping Platform**”

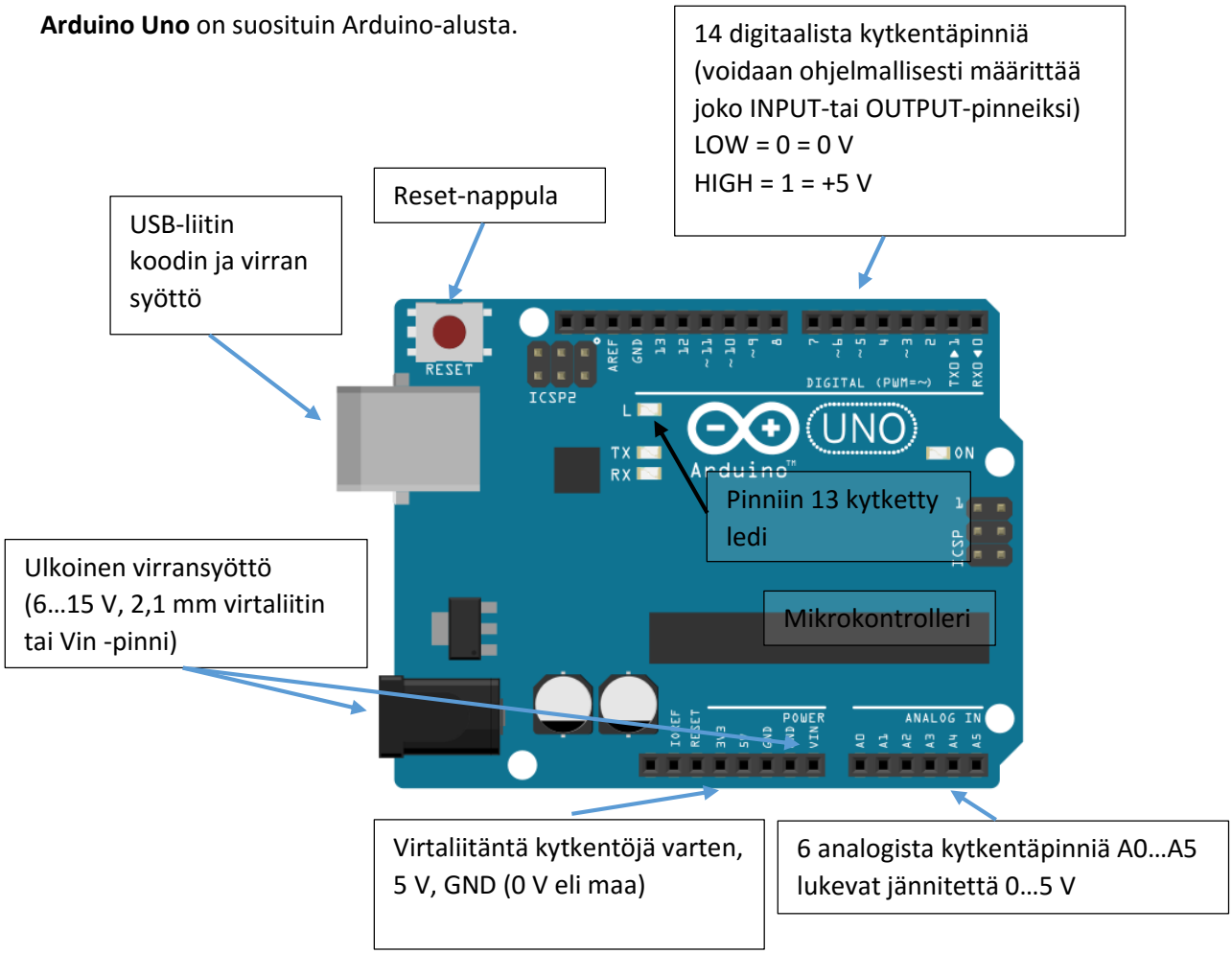
Arduino/Genuino on avoimeen laitteistoon ja ohjelmistoon perustuva mikrokontrolleri-elektroniikka-alusta ja ohjelmointiympäristö. Arduinon avulla toteutetussa laitteessa on kaksi kokonaisuutta: Arduinoon liitetty **ulkoinen kytkentä** ja Arduinoon **syötetty ohjelma**.

Arduinokortin INPUT-napoihin voi liittää erilaisia antureita, säätimiä tai kytkimiä, jotka ohjaavat OUTPUT-napoihin kytkettyjä ledejä, releitä, servoja tai moottoreita. Laitteen toiminnot määritetään laitteeseen syötetyllä ohjelmalla. Arduino-ympäristö tarjoaa helpon ja halvan keinon opetella elektroniikkaa ja ohjelmointia nykyaikaisella tavalla.

Arduino on hyvin avoin konsepti ja erilaisia *duino-tuotteita myydään nettikauppojen välityksellä huokeaan hintaan. Euroopassa ”aidot” Arduino-tuotteet myydään **Genuino**-nimen alla. Projektin kotisivu on osoitteessa www.arduino.cc.

Arduinon osat

Arduino Uno on suosituin Arduino-alusta.

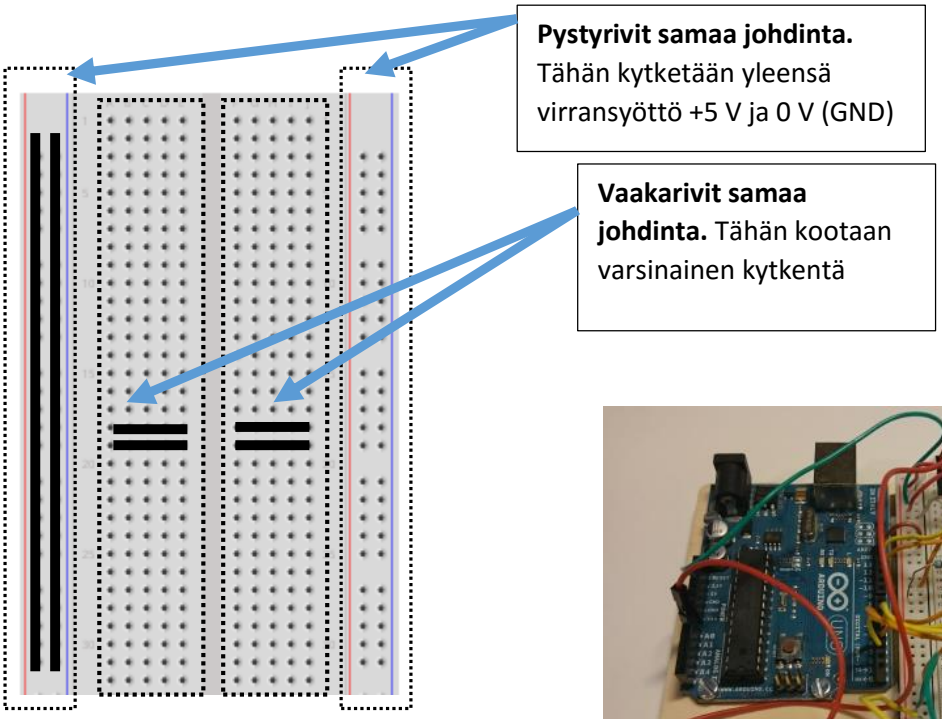


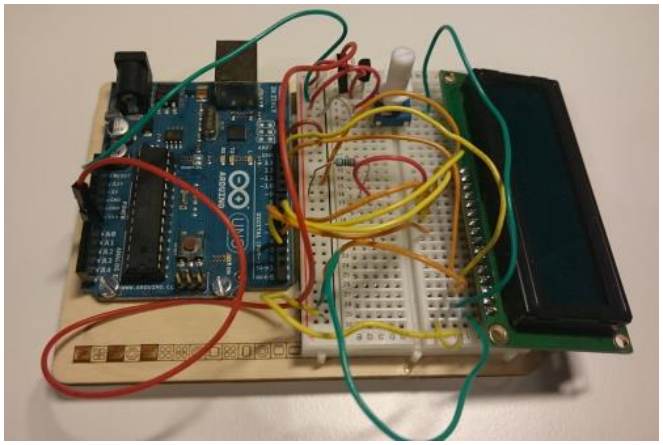
Koekytkentälevy

Kokeilukytkennät kannattaa rakentaa koekytkentälevylle. Koekytkentälevy (engl. Breadboard) on alusta, jolle voi koota kytkennän tarvitsematta juottaa komponentteja kiinni. Laitteet toteutetaan sijoittamalla komponentit koekytkentälevylle, jolloin samassa rivissä olevat komponenttien pinnit ovat yhteydessä toisiinsa. Ulkoiset liitokset Arduinoon ja muihin komponentteihin tehdään hyppylangoilla

Arduino/Genuino Starter Kit on aloittelevalle harrastajalle sopiva sarja, jossa Arduinokortin, komponenttien ja kattavan ohjekirjan lisäksi saa rakentelualustan, jossa samalle alustalle on asennettu Arduino Uno ja koekytkentälevy (kuva alla). Hieman halvemmalla alkuun pääsee hankkimalla pelkän Arduinokortin ja koekytkentälevyn. Tällöin muut komponentit täytyy kuitenkin hankkia erikseen. Liitteessä 1 on lueteltu töissä tarvittavat osat ja komponentit.

Tutkimustöiden kytkentäkuvat on piirretty toteutettavaksi Starter Kitillä.





Komponentit asetetaan koekytkentälevyn keskellä olevalle alueella, jossa on kahdessa lohkossa kytkentäpisteet.

Reunoilla kulkevat virtajohtimille tarkoitetut reikärivit on merkitty punaisella ja mustalla viivalla. Suositeltavaa on kytkeä +5 V punaiseen ja 0 V (eli maa, GND) mustaan riviin, niin välttyy virheiltä laitteiden virransyötössä.

Koekytkentälevyn reikien välit ovat standardikokoisia, joten komponentit sopivat pääsääntöisesti hyvin kytkentäalustalle.

Ohjelmointiympäristö

Lataa Arduinon ohjelmointiympäristö osoitteesta https://www.arduino.cc/en/Main/Software. Ohjelma on ilmainen ja voit valita nappulan ”Just download”. Jos haluat tukea ohjelman kehittäjiä, niin voit valita summan ja nappulan ”Contribute & Download”. Sivulta löytyy Windows-, Mac- ja Linux-versiot.

Windows-versiota asennettaessa asennusohjelma antaa mahdollisuuden asentaa automaattisesti myös USB-ajurin. Ajuri kannattaa asentaa, koska ilman sitä tietokone ei tunnista Arduinokorttia.

Ohjelman käyttö

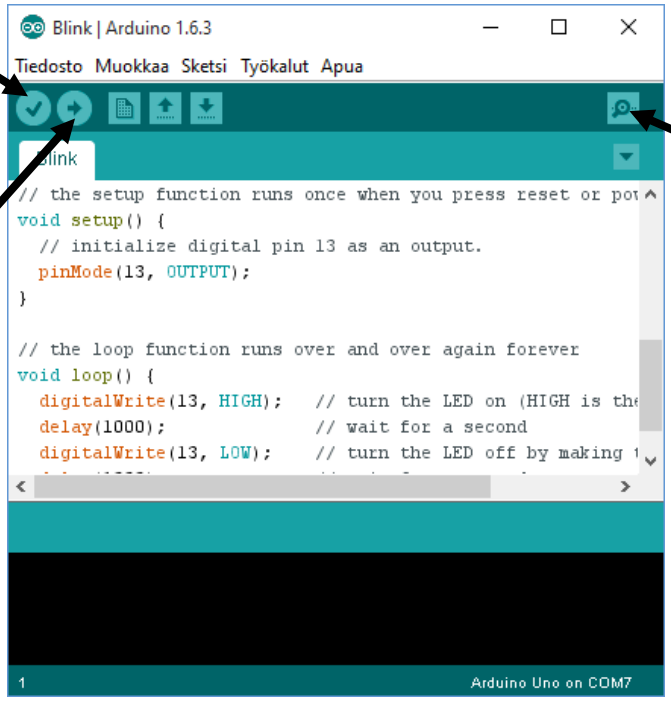
1. Avaa Arduino-ohjelma koneelle
2. Kytke Arduino-kortti USB-johdolla koneelle
3. Tarkista Tools/Työkalut-valikosta Board/Kortti (→ Arduino Uno) ja Serial Port/Portti (yleensä COM3 tai suurempi)
4. Ohjelma ”Sketch” ladataan Arduinolle vasemman yläkulman nuolella merkitystä nappulasta

Seikkaperäiset ohjeet ovat sivulla https://www.arduino.cc/en/Guide/Windows (ohjeet löytyvät myös editoriohjelman Apua-toiminnosta).

Ohjelmoinnin tärkein apusivusto on Arduinon oma opas osoitteessa https://www.arduino.cc/en/Reference/HomePage (linkki Learning → Reference).

Verify/Tarkista
Ohjelma tarkistetaan, mutta ei ladata Arduinolle

Upload/Lataa
Ohjelma ladataan Arduinolle



Avaa sarjaportin monitorointi-ikkunan

Ensimmäinen ohjelma BLINK

Ohjelma vilkuttaa kortilla jo valmiiksi olevaa digitaaliseen pinniin 13 kytkettyä lediä. Mitään kytkentöjä ei koekytkentälevylle tarvitse tehdä.

1. Avaa Arduino-ohjelma tietokoneelle. Ohjelmointiympäristö aukeaa.

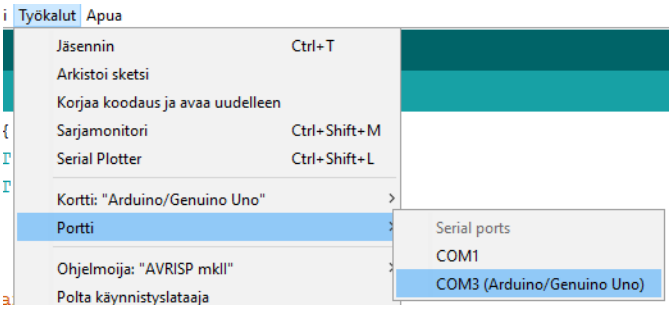
2. Avaa editoriohjelmaan tiedosto File/Tiedosto → Examples/Esimerkit → Basics → Blink. Vaihtoehtoisesti voit kirjoittaa ohjelman editori-ikkunaan (koodi on seuraavalla sivulla).

Lataa ohjelma Arduinolle

3. Kytke Arduinokortti USB-johdolla tietokoneeseen.

Tarkista, että tietokone tunnistaa Arduinon:

- Valikossa Työkalut → Kortti → Arduino/Genuino Uno,
- Työkalut → Portti → (jokin COM-portti valittuna, yleensä COM3 tai suurempi).



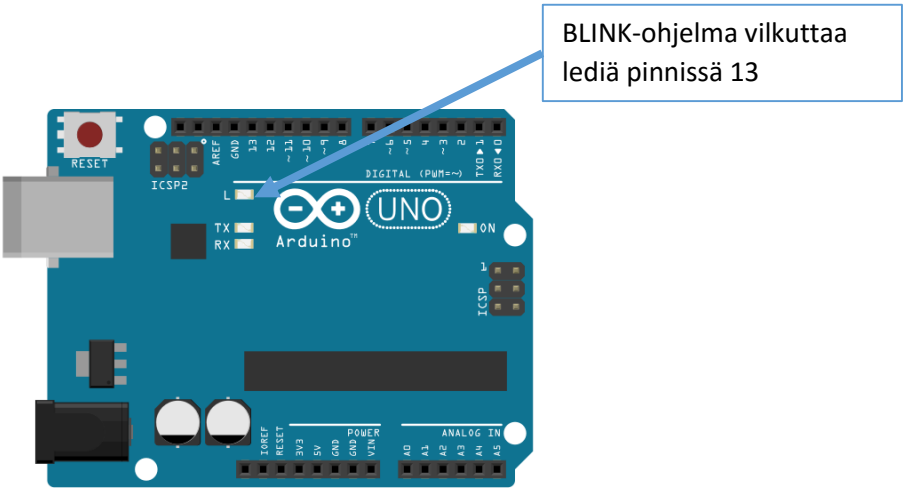
Lataa ohjelma Arduinolle työkalupalkin nuolinappulasta.

Arduino saattaa samalla kysyä talletatko tiedoston.

Tiedostoa ei ole pakkoa tallettaa, joten voit painaa nappulaa ”Peruuta”.

Ladattu ohjelma vilkuttaa pinniin 13 kytkettyä lediä.

BLINK-ohjelma on myöhemminkin käyttökelpoinen erityisesti silloin, kun täytyy nopeasti kokeilla, toimiiko yhteys Arduinolevylle oikein ja onko levy kunnossa vai viallinen.



```
/*  Blink
  Sytyttää ja sammuttaa lediä sekunnin välein
*/

// Pinniin 13 on kytketty ledi Arduinolevyllä


// SETUP ajetaan, kun ohjelma käynnistetään

void setup() {

  // asetetaan pinni 13 OUTPUT-navaksi.

  pinMode(13, OUTPUT);

}

// silmukkaa LOOP pyöritetään, kunnes levyltä katkaistaan virta:

void loop() {

  digitalWrite(13, HIGH);   // ledi päälle (HIGH → 5 voltin jännite)

  delay(1000);              // odotetaan sekunti

  digitalWrite(13, LOW);    // ledi pois päältä (LOW → 0 voltin jännite)

  delay(1000);              // odotetaan sekunti

}
```

Hyvään ohjelmointitapaan kuuluu kirjoittaa ohjelmatekstin joukkoon selventäviä kommentteja.

Ohjelman kommentit on erotettu joko /*…*/ tai // merkeillä. Kommenteissa kerrotaan ko. lauseen, käskyn tai ohjelman osan toiminta tai merkitys.

Kun ohjelmaa ladataan Arduinolle, kone ohittaa kommentit. Kommentit ovat ohjelmoijaa varten tehtyjä muistiinpanoja.

Varsinaiset ohjelman käskyt on merkitty oheiseen esimerkkiin lihavoidulla tekstillä.

Arduino-ohjelmassa on kaksi osaa: *setup*-lohko ja *loop*-lohko.

- *Setup*-lohkossa asetetaan ohjelman kertaluontoiset asetukset ja tämä ohjelman osa ajetaan yhden kerran ohjelman käynnistyessä.
- *Loop*-lohko on varsinainen ohjelmasilmukka. Silmukkaa pyöritetään niin kauan, kuin levyllä on virta päällä.

Tutki koodia ja kokeile

1. Tutki ohjelmakoodia. Yritä päätellä, mitä kukin ohjelman käsky tarkoittaa.
   Ohjelmassa oleva *delay*-käsky keskeyttää ohjelman etenemisen 1 sekunnin ajaksi (1000 ms). Ledi vilkkuu siis sekunnin tahdissa.

2. Muuta *delay*-käskyssä olevaa aikaa pidemmäksi tai lyhyemmäksi. Kuinka lyhyen välähdyksen pystyt vielä havaitsemaan?

   Kun teet ohjelmakoodiin muutoksia, paina työkalupalkissa olevaa latausnappulaa. Muutettu ohjelma siirtyy Arduinokortille. Pelkkä tekstin muuttaminen ei vielä riitä!

LEDin vilkuttelu, liikennevalot

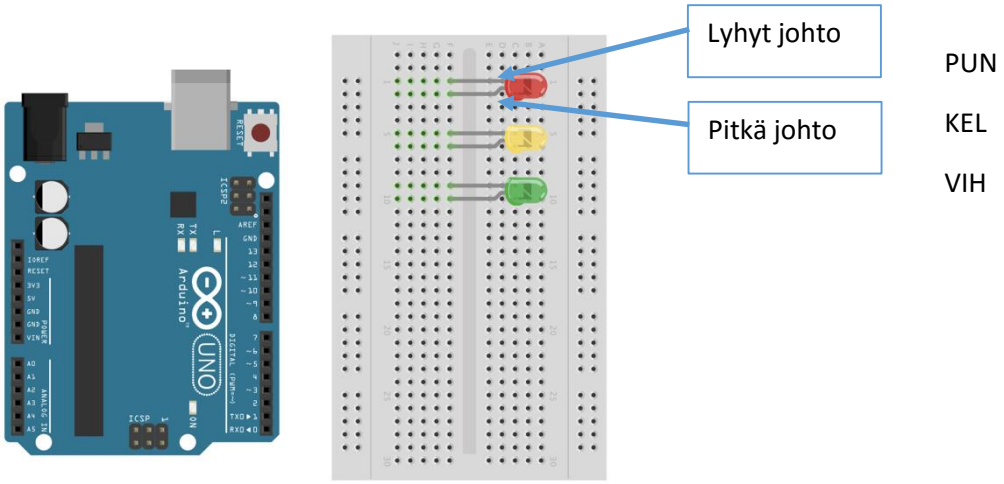
Tarvittavat välineet:

- koekytkentälevy ja Arduino/Genuino Uno
- 3 lediä (punainen, keltainen ja vihreä)
- 3 kpl 220 ohmin vastuksia (värikoodi punainen-punainen-ruskea tai punainen-punainen-musta-musta. **Huom.: vastusten värikoodimerkinnöistä on lukukappale kirjasen loppuosassa**)
- hyppylankoja johdotusta varten (4 kpl)

Kytkennän kokoaminen

1.
Aseta Arduinokortti ja koekytkentälevy kuvan mukaisesti.

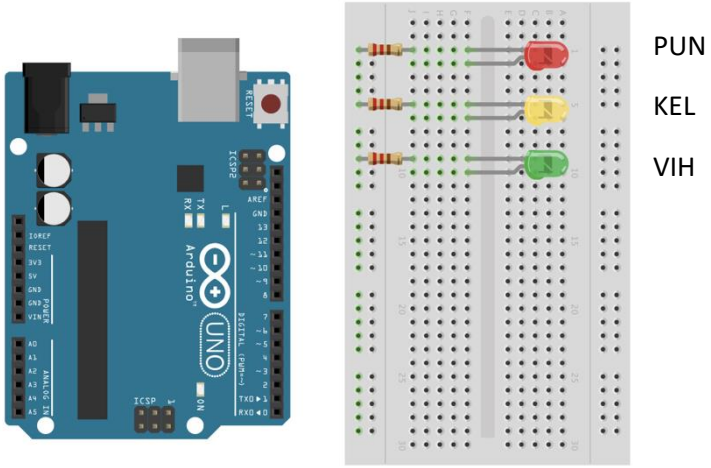
2.
Paina punainen, keltainen ja vihreä ledi koekytkentälevylle. Ledissä on kaksi johtoa. **Pidempi johto asetetaan ”alapuolelle”.**



3.
Laita kolme 220 ohmin vastusta kuvan mukaisesti koekytkentälevylle.

220 ohmin vastuksen ympärillä on värirenkaat punainen-punainen-ruskea (tai punainen-punainen-musta-musta). Vastukset voit asettaa kummin päin hyvänsä, suunnalla ei ole väliä.

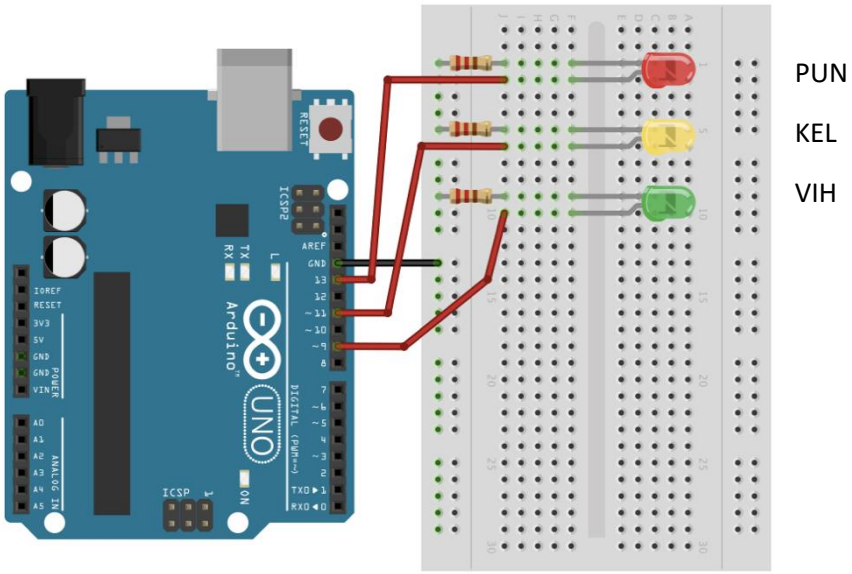
Kummankin vastuksen toinen johdin tulee asettaa aivan levyn reunalla olevaan reikäriviin (pystysuuntainen rivi). Ledin puoleinen johto tulee kiinnittää samalle riville ledin ylemmän johdon kanssa (vaakasuuntainen rivi).



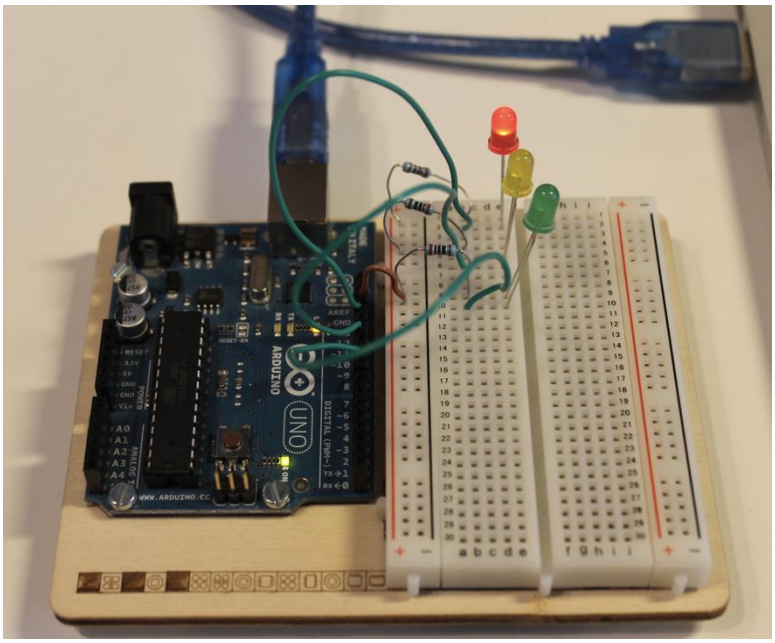
4.
Yhdistä Arduinokortti kytkentälevyyn kolmella hyppylangalla. Lankojen värillä ei ole väliä. Tärkeintä on, että itse erotat eri johtimet.
Arduinon pinnit on numeroitu piirilevyn reunaan.

Kytke hyppylanka

- Arduinon pinnistä 13 punaisen ledin alempaan kytkentäjohtoon (samalle reikäriville),
- Arduinon pinnistä 11 keltaisen ledin alempaan kytkentäjohtoon (samalle reikäriville),
- Arduinon pinnistä 9 vihreän ledin alempaan kytkentäjohtoon (samalle reikäriville),
- Arduinon pinnistä GND (Ground) koekytkentälevyn reunan reikäriviin.



Kytkentä on valmis!



Kokeillaan BLINK-ohjelmaa:

Etsi valikosta Tiedosto/File → Esimerkit/Examples → Basics → Blink. Saat valmiin koodin muokattavaksi.

```
void setup() {
   pinMode(13, OUTPUT);
}

void loop() {
   digitalWrite(13, HIGH);
   delay(1000);
   digitalWrite(13, LOW);
   delay(1000);
}
```

Ohjelman täydentäminen – kaikki ledit vilkkumaan

Lisää ohjelmaan seuraavat rivit

```
void setup() {
   pinMode(9, OUTPUT); // Asetetaan pinnit 9, 11 ja 13 OUTPUT-pinneiksi
   pinMode(11, OUTPUT);
   pinMode(13, OUTPUT);
}

void loop() {
   digitalWrite(13, HIGH); // vilkutetaan pinnin 13 lediä
   delay(1000);
   digitalWrite(13, LOW);

   digitalWrite(11, HIGH); // vilkutetaan pinnin 11 lediä
   delay(1000);
   digitalWrite(11, LOW);

   digitalWrite(9, HIGH); // vilkutetaan pinnin 9 lediä
   delay(1000);
   digitalWrite(9, LOW);
}
```

Lataa ohjelma Arduinolle. Ohjelma vilkuttaa ledejä, mutta toiminta ei ole vielä aivan samanlaista kuin oikeissa liikennevaloissa.

Lisätutkimuksia

1.
Muuta ohjelmaa niin, että saat liikennevalot toimimaan ”oikein”:

- punainen palaa pitkään,
- punaisen vaihtuessa vihreäksi punainen ja keltainen ledi palavat hetken yhtä aikaa,
- vihreä palaa pitkään,
- vihreän vaihtuessa punaiseksi keltainen valo palaa hetken aikaa.

Mieti, miten saat nämä toiminnot ohjelmaan?

2.
Tutki ohjelmakoodia ja päättele, mitä ohjelman käskyt

digitalWrite
delay

tekevät. Miten koodiin on merkitty käskyn ”kohde” ja ”vaikutus”.

Mitä merkitystä mahtaa olla ohjelmassa alussa olevilla *pinMode*-käskyllä?

Miksi arvelet ohjelmassa olevan kaksi lohkoa, *setup* ja *loop*?

3.

A. Tee ohjelma, joka vilkuttaa kaikkia ledejä yhtä aikaa samassa tahdissa. Lataa ohjelma Arduinolle ja testaa toiminta.
B. Muokkaa ohjelmaa niin, että ledit syttyvät vuorotellen ja saat ledistä toiseen liukuvan valon.
C. Tee kytkennästä ”metronomi”. Yksi ledi (väri) vilkkuu kolme kertaa (yks-kaks-kol-) ja neljännellä iskulla (-nel) kaikki ledit loistavat

Tärkeät funktiot

pinMode(*pinni, toiminta*);

Asettaa käskyssä määritetyn pinnin joko OUTPUT-pinniksi tai INPUT-pinniksi.

pinni: jokin Arduinon pinninumero
toiminta: INPUT tai OUTPUT

digitalWrite(*pinni, arvo*);

Kirjoittaa arvon HIGH tai LOW digitaaliseen pinniin

pinni: jokin Arduinon pinninumero
arvo: HIGH vastaa 5 voltin jännitettä digitaalisessa pinnissä (looginen 1), LOW vastaa nollan voltin jännitettä digitaalisessa pinnissä (looginen 0).

Parametri HIGH sytyttää ja LOW sammuttaa ledit eli asettaa pinnin jännitteeksi joko +5 V tai 0 V. Parametri tarkoittaa käskylle annettavia alkuarvoja, joiden mukaan toiminto suoritetaan.

delay(*aika*);

Keskeyttää ohjelman annetuksi ajaksi.

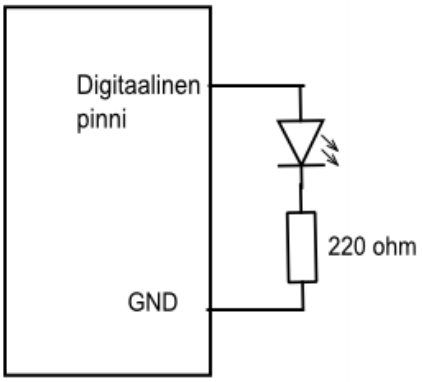
aika: aika annetaan millisekunteina (1000 *mS* = 1 *s*)

Tehtäviä

1. Miksi ohjelmakoodia on hyvä kommentoida?
2. Mitä merkitystä on ohjelman setup-lohkolla?
3. Mihin lohkoon varsinainen ohjelma sijoitetaan?
4. Millä merkillä ohjelmarivit erotetaan toisistaan, entä ohjelman lohkot?

Ledin kytkentä Arduinoon

Diodi on puolijohdekomponentti, joka päästää sähkövirtaa vain toiseen suuntaan, ns. päästösuuntaan. Kun diodi käännetään toisinpäin, estosuuntaan, sen läpi ei pääse sähkövirtaa. LED (Light Emitting Diode) on diodi, joka lähettää valoa kytkettäessä se päästösuuntaan.



Päästösuunnassa olevan diodin resistanssi on pieni, jolloin komponentin läpi saattaa kulkea pienelläkin jännitteellä suuri sähkövirta, joka voi vioittaa lediä. Ledin kanssa sarjaan tulee kytkeä sarjavastus, joka rajoittaa ledin läpi kulkevan sähkövirran noin 10…20 milliampeeriin.

Hehkuvan ledin napojen välillä vaikuttaa ns. kynnysjännite, noin 1,5 V. Arduinon digitaalisessa pinnissä on 5 voltin jännite. Vastuksen napojen välinen jännite on siis *(5 - 1,5) V = 3,5 V*. Kun jännite ja sähkövirta tiedetään, voidaan vastuksen resistanssin arvo laskea:

$R = U/I = 3{,}5\ V / 0{,}02A = 175\ \Omega$

Lähin vastuksen kaupasta saatava standardiarvo on 220 ohmia, jota voidaan käyttää kytkennöissä ledin sarjavastuksena. Vastusten arvot 220… 1000 ohmia käyvät hyvin ledin sarjavastukseksi.

Kirjasen kytkennöissä ledin sarjavastus on kytketty virtapiirissä säännöllisesti ledin ”maan” puolelle. Aivan yhtä hyvin vastuksen voi sijoittaa toiselle puolelle. Järjestyksellä ei ole sähköisen toimivuuden kannalta väliä.

RGB-ledi

Tarvittavat välineet:

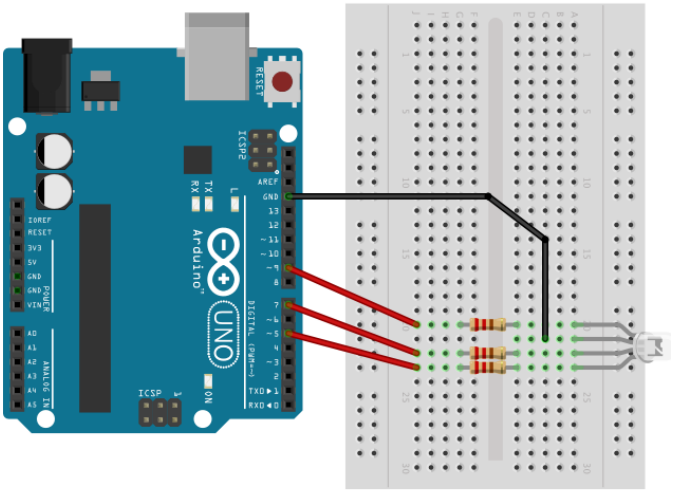
- koekytkentälevy ja Arduino/Genuino Uno
- RGB-ledi tai 3 lediä (punainen, keltainen ja vihreä)
- 3 kpl 220 ohmin vastuksia (punainen-punainen-ruskea tai punainen-punainen-musta-musta)
- hyppylankoja johdotusta varten (4 kpl)

RGB-ledi sisältää samassa kotelossa punaisen, vihreän ja sinisen ledin. Mikäli käytössä ei ole RGB-lediä voit käyttää tavallisia yksivärisiä ledejä samaan tapaan kuin liikennevalotutkimuksessa.

RGB-ledin ledeillä on yhteinen katodi. Anodiliitäntäjohtimia on kolme kullakin eri värisellä ledillä. Johdinjärjestys on kuvan mukainen, pisin johto on katodi.

Kutakin lediä voi ohjata kytkemällä Arduinon digitaalinen OUTPUT-napa 220 ohmin vastuksen kautta ledin anodille.

Kytkentä



Kytke RGB-ledi ja 220 ohmin vastukset koekytkentälevylle. Kuvaan on piirretty kullekin ledille oma hyppylanka ja vastus. Voit myös liittää vastukset suoraan Arduinon pinneille, hyppylangat ovat tällöin tarpeettomia.

Yhdistä RGB-ledin punainen ledi Arduinon pinniin 3, vihreä pinniin 5 ja sininen pinniin 7. Ledin katodi kannattaa yhdistää koekytkentälevyn reunan liitäntärivin kautta Arduinon maahan (GND).

Ohjelma

Ohjelma on samankaltainen kuin aiemmassa liikennevalo-ohjelmassa. Erona on, että lisätään ledipinnien määrittelyt globaaleina muuttujina. Tämä helpottaa koodin luettavuutta, koska pinnille annetaan toimintaa kuvaava nimi pelkän numeron sijaan.

Kokeile ledien sytyttämistä yksi ledi kerrallaan.

```
// Määritetään globaalit muuttujat, jotka näkyvä koko ohjelmalle (pinnit, joihin
// RGB-ledin eri värit on kytketty)
int ledPun = 3; // muuttujan ledPun arvoksi annetaan 3
int ledSin = 5; // muuttujan ledSin arvoksi annetaan 5
int ledVih = 7; // muuttujan ledVih arvoksi annetaan 7

void setup() {
  pinMode(ledPun, OUTPUT); // asetetaan pinnit OUTPUT-tilaan
  pinMode(ledVih, OUTPUT);
  pinMode(ledSin, OUTPUT);
}

void loop() {
  digitalWrite(ledPun, HIGH); // Tässä vilkutetaan vain yhtä lediä
  delay(1000);
  digitalWrite(ledPun, LOW);
  delay(1000);
}
```

Määritetään aluksi ledipinnit *globaaleina muuttujina.* Globaalit muuttujat sijoitetaan aivan ohjelman alkuun ennen setup-lohkoa. Globaali muuttuja näkyy ohjelman kaikille osille ja siihen voi viitata kaikkialta ohjelmassa. Tämän määrittelyn ansiosta ledipinnien muuttaminen käy helposti. Pinnien numerointia ei tarvitse muuttaa monessa kohtaa, vaan riittää, kun tekee muutoksen vain ko. globaalille muuttujalle. Lisäksi muuttujan nimi kertoo havainnollisesti muuttujan ”kohteen”. Koodia on helpompi lukea.

```
int ledPun = 3;
int ledSin = 5;
int ledVih = 7;
```

Vastaavat muutokset täytyy tehdä myös ohjelman runkoon setup-lohkoon:

```
pinMode(ledPun, OUTPUT);
pinMode(ledSin, OUTPUT);
pinMode(ledVih, OUTPUT);
```

Varsinaiseen ohjelmasilmukkaan, loop-lohkoon, sijoitetaan ohjelmakäskyt, jotka sytyttävät ledit:

```
digitalWrite(ledPun, HIGH);
digitalWrite(ledSin, HIGH);
digitalWrite(ledVih, HIGH);
```

Ledit saadaan sammutettua käskyillä

```
digitalWrite(ledPun, LOW);
digitalWrite(ledSin, LOW);
digitalWrite(ledVih, LOW);
```

Delay-lauseella saat valot loistamaan tietyn aikaa, esim.

delay(1000);

keskeyttää ohjelman yhden sekunnin ajaksi.

Muokkaa ohjelmaa
Kokeile ohjelmoida samat toiminnot kuin aiemmin.

A. Muokkaa ohjelmaa niin, että ledit syttyvät vuorotellen ja saat väristä toiseen vaihtuvan valon. Lataa ohjelma Arduinolle ja testaa toiminta.
B. Tee ohjelma, joka vilkuttaa kaikkia ledejä yhtä aikaa samassa tahdissa. Minkä väriseltä RGB-ledi näyttää? Erotatko erilliset ledit?
C. Tee kytkennästä ”metronomi”. Yksi ledi (väri) vilkkuu kolme kertaa (yks-kaks-kol-) ja neljännellä iskulla (-nel) kaikki ledit loistavat

Tehtäviä

1. Piirrä kytkennän piirikaavio. Ota mallia piirikaaviosta, jossa on yhden ledin kytkentä. RGB-ledissä on kolme erillistä lediä.
2. Mitä *pinMode*-lause tekee? Mitkä *parametrit* täytyy ko. käskylle antaa?
3. Miten saat *digitalWrite*-käskyn osoittamaan tiettyä pinniä? Mitä tarkoittaa *digitalWrite*-käskyssä LOW ja HIGH?
4. Haluat pysäyttää ohjelman toiminnan 2,7 sekunnin ajaksi. Kirjoita sopiva *delay*-lause
5. Mitä tarkoittaa *globaali muuttuja*?

Ohjelman rakenteita, FOR-silmukka

Ohjelman toistuvia toimintoja voi toteuttaa esimerkiksi FOR-silmukan avulla.

Käytä esimerkissä RGB-ledin kytkentää. Ledit on kytketty pinneihin 5, 7 ja 9.

FOR-Silmukan rakenne

```
for (int i=0; i <= 255; i++) {

// muuttuja i on silmukan laskuri. Muuttuja tulee tarvittaessa määritellä (jos sitä ei ole jo aiemmin
// määritelty). Muuttujan voi nimetä vapaasti (ei tarvitse olla i)

// Silmukkaa toteutetaan niin kauan kun laskurille asetettu ehto toteutuu.

// Silmukan kierroksen jälkeen laskurin arvoa muutetaan
// Merkintä i++ tarkoittaa sijoitusta i = i + 1. Siis laskurin arvoa kasvatetaan yhdellä

// ohjelmakäskyt aaltosulkeiden sisään

 }
```

1.

```
// FOR-silmukka

// Vilkutetaan pinniin 5 kytkettyä lediä niin, että
// ledin palamisaika kasvaa jokaisella for-silmukan kierroksella

int led = 5;  // määritetään globaali muuttuja ja annetaan niille arvot;

void setup()
{
  pinMode(led, OUTPUT); // asetetaan OUTPUT-pinnit
}

void loop()
{
  for (int i=1; i <= 10; i++){      // FOR-silmukka, muuttujan i alkuarvo on 1, kasvatetaan
    digitalWrite(led, HIGH);        // i:n arvoa yhdellä jokaisella kierroksella kunnes i saa arvon 10
    delay(200*i);                   // Vilkun pituutta kasvatetaan 0,2 s - - > 2 s
    digitalWrite(led, LOW);         // Vilkutetaan punaista lediä (vrt. BLINK-ohjelma).
    delay(500);
  }
}
```

2.

Seuraavassa esimerkissä sijoitetaan *taulukkoon* arvoja. Taulukon arvoja voidaan hakea koodiin taulukon alkioon viittaamalla.

Huomaa, että ohjelman SETUP-lohkoon voi myös kirjoittaa normaalia ohjelmakoodia (ei siis pelkästään määrittelyjä). Varsinaisessa ohjelmaluupissa on käytetty kahta sisäkkäistä FOR-silmukkaa.

```
// FOR-silmukka

int Ledit[5, 7, 9];  // määritetään taulukko Ledit, jonka alkioiksi sijoitetaan luvut 5, 7 ja 9
// Luvut kuvaavat pinnejä, joihin on kytketty ledi.
// Koodissa tiettyyn pinniin voidaan viitata kirjoittamalla esimerkiksi Ledit[3]

void setup()
{
  for (int pinni = 1; pinni <=3; pinni++) {
    pinMode(Ledit[pinni], OUTPUT); // asetetaan OUTPUT-pinnit, jotka haetaan taulukosta Ledit
  }
}

void loop()
{
  for (int laskuri=1; laskuri <= 3; laskuri++) { // sisäkkäisiä FOR-silmukoita

    for (int i=1; i <= 10; i++){                 // FOR-silmukka, muuttujan i alkuarvo on 1, kasvatetaan
      digitalWrite(Ledit[laskuri], HIGH);        // i:n arvoa yhdellä jokaisella kierroksella
      delay(500);                                                  // kunnes i saa arvon 10.
      digitalWrite(Ledit[laskuri], LOW);         // Vilkutetaan lediä (vrt. BLINK-ohjelma).
      delay(500);
    }

  }

}
```

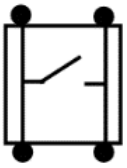
Painonapin lukeminen

Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- 2 kpl painonappikytkimiä
- 2 kpl vastuksia 220 Ω (punainen-punainen-ruskea tai punainen-punainen-musta-musta)
- 2 kpl vastuksia 10 kΩ, (ruskea-musta-oranssi tai ruskea-musta-musta-punainen)
- 2 kpl LED

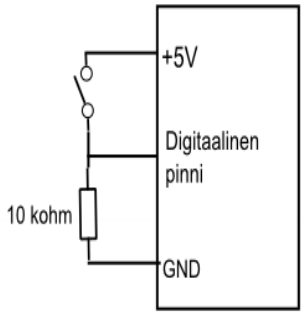
1.

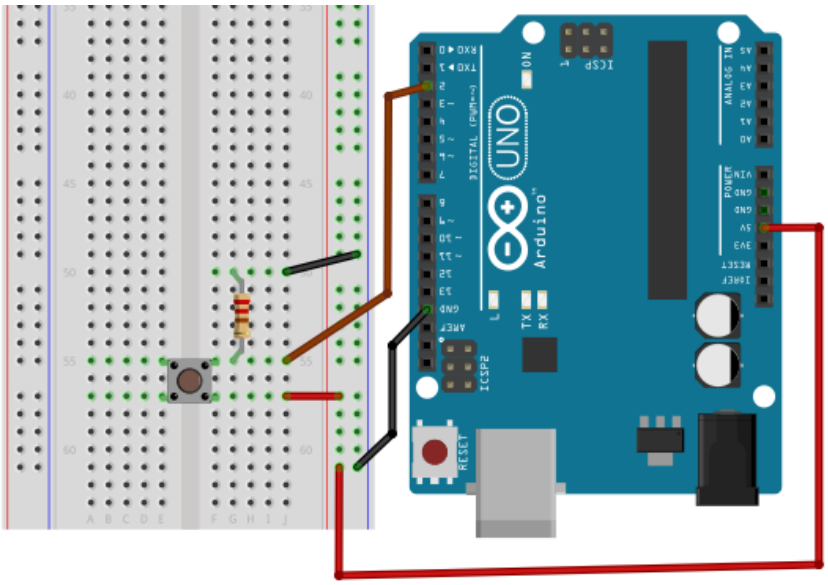
Starter Kitin painonapissa on neljä liitäntäpinniä. Kaksi pinniä on kytketty yhteen ja kytkintoiminnon kytkentä on oikeanpuoleisessa kuvassa.



Kytkimen yhteydessä käytetään 10 kΩ vastusta, joka varmistaa, että jännitetaso on varmasti 0 V tai 5 V (looginen 0 tai 1). Kun painonappia ei paineta, digitaalisen pinnin jännite on 0 volttia. Kun painonappia painetaan, digitaalisen pinnin jännite on +5 volttia.

10 kilo-ohmin vastuksen värikoodi on ruskea-musta-oranssi (tai ruskea-musta-musta-punainen).





```
// Painonapin lukeminen
// Levyllä oleva pinnin 13 ledi palaa,
// kun painonappia painetaan

// määritetään muuttujat
const int nappiPin = 2;          // painonappipinni
const int ledPin =  13;          // Ledin pinni
int napinTila = 0;               // painonapin tila

void setup() {
  pinMode(ledPin, OUTPUT);  // OUTPUT ledille
  pinMode(nappiPin, INPUT); // INPUT
                            //painonapille
}

void loop(){
  napinTila = digitalRead(nappiPin);
        // luetaan painonappi

  if (napinTila == HIGH) {
// jos painetaan, niin napinTila on HIGH:
    digitalWrite(ledPin, HIGH);  // sytytetään ledi
  }

  else {
    digitalWrite(ledPin, LOW);
// muutoin sammutetaan ledi
  }
}
```

2.
Edellisessä tutkimuksessa käytettiin Arduinolevyllä valmiiksi olevaa lediä.

A. Kokeile nyt kytkeä erillinen ledi. Lisää johonkin digitaaliseen OUTPUT pinniin ledi ja 220 ohmin vastus.

   Ledin lyhempi johdin (katodi) kytketään sarjavastuksen kautta ”maahan” ja pidempi johdin hyppylangan avulla valittuun pinniin. Kokeile toimintaa niin, että saat ledin palamaan painonapin avulla. Tee ohjelmaan tarvittavat muutokset.

B. Kokeile myös muuttaa ohjelmaa, että saat kytkennällä erilaisia valoefektejä. Esimerkiksi niin, että ledi palaa jatkuvasti, kun nappia ei paineta ja vilkkuu, kun nappia painetaan.

3.
Lisää toinen ledi ja muuta ohjelmaa niin, että ensimmäinen ledi palaa kun painonappia painetaan ja toinen ledi silloin, kun nappulaa ei paineta.

Painonapin ohjelmassa käytetään tärkeää ohjausrakennetta:

if…else

Mikäli **if**-sanan jälkeen kirjoitettu ehto on tosi, niin toiminto A toteutetaan. Muussa tapauksessa toteutetaan toiminto B.

Vertailuoperaattorina käytetään merkkiä == (yhtäsuuruus).

```
if (muuttuja == 500)
{
  // toiminto A
}

else
{
  // toiminto B
}
```

Muuttujat

Ohjelmassa käsiteltävät tiedot talletetaan muuttujiin. Muuttujat täytyy ohjelman alussa **määritellä**, jotta kone pystyy varaamaan niille muistitilaa. Määrittely kertoo, minkä tyyppistä tietoa muuttujaan talletetaan. Tärkeitä tyyppimääritteitä ovat **int** ja **float**.

int on kokonaislukumuuttuja (*int = integer = kokonaisluku*),
float on desimaalilukumuuttuja (*float = floating number = liukuluku*).

Muuttujan määrittelyn yhteydessä muuttujan voi **alustaa**, eli sille voi antaa alkuarvon.

Avainsana **const** tarkoittaa, että muuttuja on vakio eli sen arvo ei muutu ohjelmassa (*const = constant = vakio*).

Tärkeät funktiot

int digitalRead(*pinni*);

Lukee digitaalisen pinnin arvon, joko HIGH (looginen 1, +5 V) tai LOW (looginen 0, 0 V). Funktio palauttaa ohjelmassa määritettyyn muuttujaan joko arvon 1 (HIGH) tai 0 (LOW).

pinni: jokin Arduinon pinninumero

Muuttuja, vakio, funktio?

Ohjelmoinnin yhteydessä käytetään paljon matematiikan käsitteitä. Ohjelmoinnissa ja matematiikassa on tärkeä esittää asiat tarkasti ja loogisesti. Hyöty on molemminpuolinen: matematiikan tietoja voit suoraan soveltaa ohjelmoimiseen, toisaalta ohjelmoiminen elävöittää matemaattisia käsitteitä.

Ohjelmoinnissa funktio (engl. function) on tietyn toiminnon suorittamiseen tarkoitettu käsky tai käskysarja, joka usein palauttaa jonkin arvon. Esimerkiksi *digitalRead*-funktio palauttaa joko arvon HIGH (1) tai LOW (0). Funktion määrittelyssä oleva **int** kertoo, että funktio palauttaa kokonaisluvun (tässä tapauksessa 1 tai 0). Mikäli määrittelyssä funktion nimen edessä on **void**, niin funktio ei palauta mitään arvoa (esimerkiksi käsky *digitalWrite*).

Muuttujat (engl. variable) ovat koneen työmuistiin varattuja tietoalueita, joihin voi tallettaa tietoa. Muuttujaan talletettua tietoa voi muokata ja hakea ohjelman avulla. Kun muuttuja ensimmäisen kerran esitellään, koneelle täytyy kertoa muuttujaan talletettavan tiedon tyyppi. Esimerkiksi muuttujatyyppi **int** (kokonaisluku) varaa talletustilaa kaksi tavua. Yhdessä tavussa on kahdeksan bittiä (yksi bitti on ykkönen tai nolla). **Float** eli desimaaliluku vie tilaa neljä tavua.

Vakio (engl. constant) on arvoltaan muuttumaton luku. Lisäämällä avainsana **const** muuttujan määrittelyn eteen, muuttuja talletetaan niin, että sen pystyy ainoastaan lukemaan (”read-only”).

Tehtäviä

1. Mitä tarkoittavat ohjelmoinnissa termit a) funktio, b) muuttuja, c) vakio?

2. Miten voi ohjelmassa määritellä muuttujan? Miten saat muuttujan alustettua? Anna esimerkki.

3. Mitä tarkoittaa ”funktio palauttaa arvon”?

4. Millaisia arvoja funktio *digitalRead* palauttaa?

5. Etsi Arduinon omasta dokumentaatiosta tieto, millaisen lukualueen lukuja **int**-muuttujaan voi tallettaa.

6. Etsi Arduinon dokumentaatiosta, miten **if…else-**rakenteen vertailuehto voidaan muotoilla.

Jännitteen mittaaminen – analoginen input

Välineet

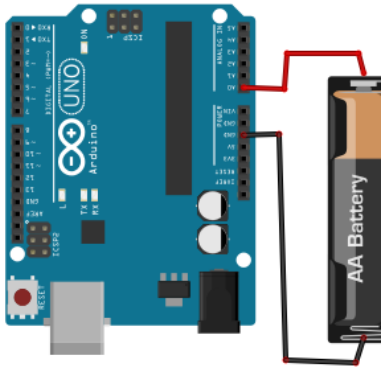
- Arduino Uno, koekytkentäalusta ja hyppylankoja
- 3 kpl 1,5 V paristoja
- Ledi, 220 Ω vastus

AD-muunnin tarkoittaa muunninta, joka muuttaa analogisen, mitä tahansa arvoja saavan jännitteen täsmälliseksi digitaaliseksi luvuksi. Arduinolevy sisältää AD-muuntimen, joka lukee Analog input-navalle kytketyn pariston tai muun komponentin jännitteen (0…5 V) ja muuttaa sen kokonaisluvuksi 0… 1023. Arduinon AD-muuntimen mittaustarkkuus on siis 5,0 V / 1024 = 4,9 mV.

5 voltin jännitteellä AD-muunnin antaa luvun 1023, 0 voltin jännite antaa luvun 0. Välissä olevat arvot voi laskea, esimerkiksi jännitettä 1,0 V vastaa kokonaisluku (1 V/5 V)*1023 = 205

1.
Kirjoita oheinen koodi ja lataa se Arduinoon.

```
// Jännitteen lukeminen analogisesta pinnistä

void setup() {
  Serial.begin(9600); // asetetaan sarjaportin nopeudeksi 9600 bittiä sekunnissa
}

void loop() {
 // luetaan analogisen pinnin A0 jännite
 int lukema = analogRead(A0);
 // tulostetaan lukema sarjaporttiin
 Serial.println(lukema);
 delay(1000);    // viive arvojen lukemiseen sekunnin välein
}
```

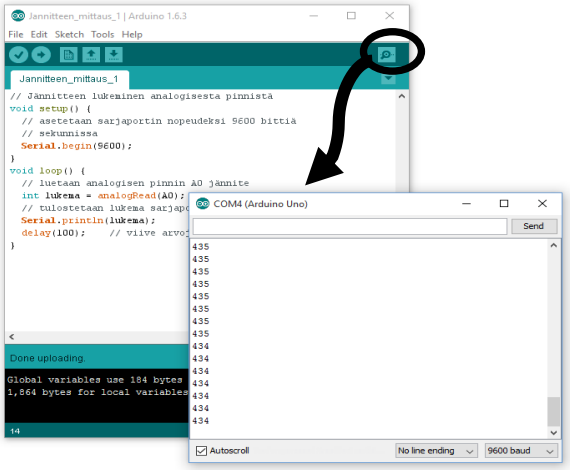


Kytke 1,5 voltin pariston plusnapa Arduinon analogiseen pinniin A0. Kytke pariston miinusnapa GND-pinniin.

Samalla tavalla voit tutkia kahta ja kolmea sarjaan kytkettyä paristoa. Paristot voit kytkeä sarjaan yksinkertaisesti niin, että asetat paristot peräkkäin. Apuna voit käyttää avonaista paristokoteloa, mikäli sellainen on käytettävissä.

Paina editoriohjelman sarjamonitorinappulasta monitorointi-ikkuna auki. Tarkista, että monitori-ikkunan oikeassa alakulmassa ilmaistu tiedonsiirtonopeus on sama kuin ohjelmaan määritelty (9600 bit/s).

Sarjamonitori-ikkunan avaaminen



Kokeile, mitä lukemia saat yhdellä paristolla (nimellisjännite 1,5 V). Kokeile myös kahdella ja kolmella sarjaan kytketyllä paristolla (nimellisjännitteet 3,0 V ja 4,5 V). **Älä kytke neljää paristoa sarjaan, sillä analoginen input on tarkoitettu alle 5 voltin jännitteille.** Kirjaa lukemat kustakin tapauksesta muistiin.

Kytke pinniin A0 kytketty hyppylanka suoraan Arduinon omiin jännitepinneihin GND, 3.3 V ja 5 V. Merkitse näistäkin mittauksista lukemat muistiin.

Laske mittausarvojen perusteella paristokytkentöjen jännitteet. Käytä kaavaa

$$jännite = \frac{lukema}{1023} * 5V$$

Kokoa mittaustulokset taulukkoon. Mittaa jännitteet myös digitaalisella yleismittarilla.

Paristo/pinni	**AD muuntimen lukema**	**Lukemasta laskettu jännite (V)**	**Yleismittarilla mitattu jännite (V)**
1 paristo			
2 paristoa			
3 paristoa			
5 V pinni			
3,3 V pinni			
GND pinni			

Vastaavatko jännitearvot oletettuja paristojen jännitteitä (nimellisjännitteet 1,5 V, 3,0 V, 4,5 V)? Entä Arduinolevyn omia jännitteitä?

Tuoreen 1,5 voltin pariston jännite voi olla luokkaa 1,6 volttia, pitkään käytetyn puolestaan reilusti alle nimellisjännitteen.

Kokeile myös, että pidät pelkkää johdinta analogisessa pinnissä. Vie kätesi johtimen lähelle ja kosketa sitä. Seuraa monitori-ikkunan lukemia. Huomaat, että analoginen input-napa on hyvin herkkä häiriöille. Lukema saattaa vaeltaa paljonkin.

2.
Muokkaa edellistä ohjelmaa, niin saat jännitemittarin. Tulokset saat suoraan voltteina.

Kirjoita oheinen koodi ja lataa se Arduinoon.

```
// Jännitemittari. Jännitteen lukeminen analogisesta pinnistä ja tulostus voltteina

void setup() {
  Serial.begin(9600); // asetetaan sarjaportin nopeudeksi 9600 bittiä sekunnissa
}

void loop() {
 // luetaan analogisen pinnin A0 jännite
 int lukema = analogRead(A0);
 // muutetaan lukema volteiksi ja vaihdetaan samalla tietotyyppiä
 float jannite = (lukema/1023.0)*5.0;
 // tulostetaan lukema sarjaporttiin
 Serial.print("Jännite volteissa: "); // tulostetaan selosteteksti
 Serial.println(jannite); // tulostetaan jännitteen arvo
 delay(1000);    // viive arvojen lukemiseen
}
```

Kokeile piirin ja ohjelman toimintaa yhdellä, kahdella tai kolmella sarjaan kytketyllä paristolla. Ovatko tulokset odotettuja?

3.
Tee paristotesteri.

Lisää koekytkentälevylle punainen ja vihreä ledi ja 220 ohmin vastukset. Kytke ledit kahteen digitaaliseen output-pinniin. Katso tarvittaessa mallia ledin vilkutustöistä. Piirrä vihkoon kytkennän piirikaavio.

Muokkaa edellisen kohdan ohjelmaa niin, että saat jännitehälyttimen: ledi syttyy kun jännite alittaa ennalta ohjelmaan asetetun arvon. Käytetyn 1,5 voltin pariston jännite voi olla alle 1,4 volttia. Tuoreen pariston jännite on 1,5-1,6 V.

Ohjelmassa voit käyttää if – else –ohjausrakennetta tyyliin

```
if (lukema < 500) {
  // ledi 1 sytytetään
  // ledi 2 sammutetaan
}

else {
  // ledi 2 sytytetään
  // ledi 1 sammutetaan
}
```

Lediä varten ohjelmaan täytyy lisätä myös *pinMode*- ja *digitalWrite*-käskyt oikeisiin kohtiin. Kokeile!

Tutki, mikä olisi sopiva arvo vertailulukemaksi. Käytä apuna sarjaportin monitoria ja edellisessä tehtävässä mitattuja jännitteen ja AD-muuntimen lukemia. Kun ohjelma on valmis, voit ”kommentoida” pois sarjaporttiin liittyvät ohjelmarivit (lisää //-merkit ohjelmarivin eteen, jolloin rivien yli hypätään ohjelmaa käännettäessä).

Voit lisätä kytkentään myös keltaisen ledin. Ohjelmoi toiminta niin, että vihreä ledi palaa, jos pariston jännite on yli 1,5 V, keltainen ledi palaa pariston jännitteellä 1,4-1,5 V ja punainen ledi palaa jännitteellä alle 1,4 volttia. Katso tässä tapauksessa ohjeistusta if-else-rakenteelle Arduinon omasta dokumentaatiosta.

Voit testata ohjelman ja piirin toimintaa säädettävällä jännitelähteellä. **Muista, että Arduino on suunniteltu maksimissaan 5 voltin jännitteelle. Älä käytä suurempaa jännitettä, Arduinolevy voi rikkoutua!**

Tärkeät funktiot

int analoglRead(*pinni*);
Lukee analogisen pinnin jännitteen (0…5 V) ja muuttaa jännitelukeman kokonaisluvuksi välillä 0…1023. Funktio palauttaa ohjelmassa määritettyyn muuttujaan ko. kokonaisluvun.
pinni: jokin Arduinon pinninumero

Serial.begin
Asettaa tiedonsiirtonopeuden, esim. **Serial.begin(9600);** asettaa nopeuden arvoon 9600 bit/s.

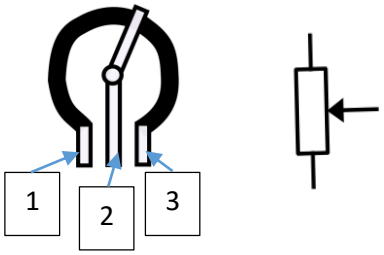
Serial.println(lukema)
Tulostaa tekstiä sarjaporttiin niin, että rivi päättyy rivinvaihtoon. Rivi on luettavissa sarjaportin monitori-ikkunasta. **Serial.println(lukema);** tulostaa muuttujan *lukema* arvon. Mikäli haluat tulostaa tekstiä, laita teksti lainausmerkkien sisään, esimerkiksi **Serial.println(”jännite on: ”);.**

Tehtäviä

1. Laadi taulukko ja merkitse siihen Arduinon AD-muuntimen jännitteitä 0 V, 1 V, 2 V, 2,5 V, 3 V, 4 V ja 5 V vastaavat kokonaisluvut.
2. Piirrä piirikaaviot eri kytkennöistä.
3. Kerro omin sanoin, miten jännitteen laskukaava saadaan.

Jännitteenjakaja – potentiometrin lukeman lukeminen

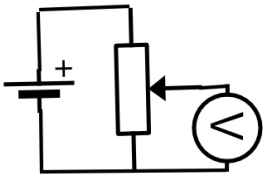
Potentiometriksi on nimetty säädettävä vastus, jossa on kolme liitäntänastaa kuvan mukaisesti. Potentiometrissä on esimerkiksi hiilestä valmistettu hevosenkengän muotoinen vastusliuska, jonka päihin on kiinnitetty liitäntänastat. Keskellä on akseliin liitetty liukukosketin, jota voidaan liikuttaa portaattomasti vastusliuskan päällä.



Potentiometrin rakenne on kuvattu myös piirrosmerkissä. Nuoli kuvaa liukukosketinta.

10 kilo-ohmin potentiometrissä on napojen 1 ja 3 välinen resistanssi 10 kΩ. Napojen 1-2 tai 2-3 välistä resistanssia voi säätää liukukoskettimella vapaasti välillä 0-10 kΩ.

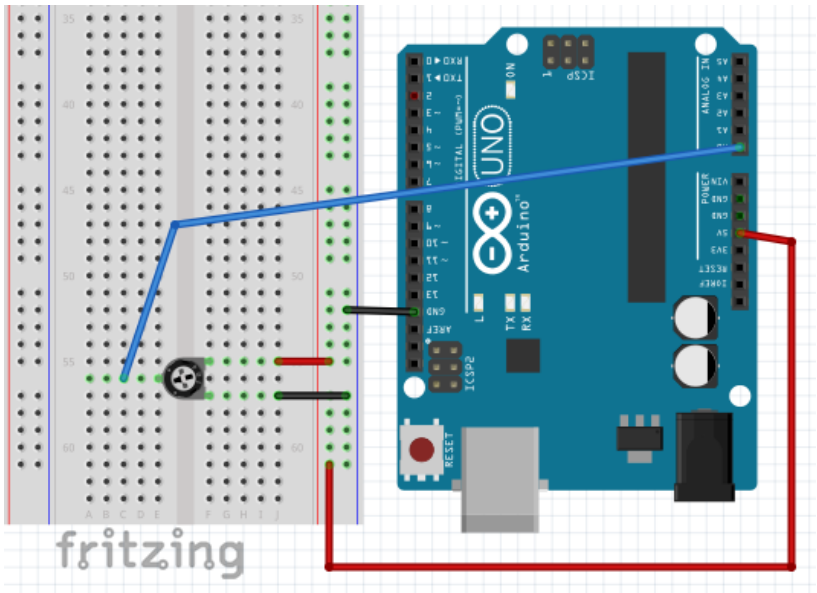
Potentiometri voidaan kytkeä **jännitteenjakajaksi**. Yleensä potentiometrin toinen äärinasta on kytketty jännitelähteen toiseen napaan (esim. Arduinon +5 V) ja toinen nasta maahan (Arduinon GND-pinni eli 0 V). Tällöin keskimmäiseen nastaan voidaan säätää mikä tahansa jännite nollan ja viiden voltin välillä.

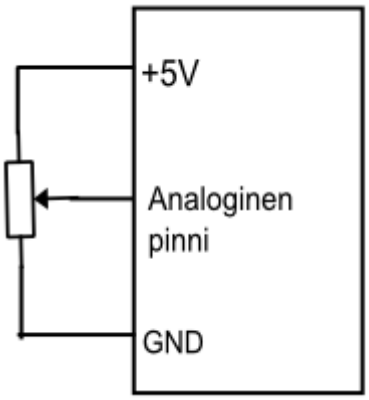


Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- 10 kΩ potentiometri
- 220 Ω vastus
- ledi

1.
Tee kytkentä.





Kokeile ohjelmana edellisen työn (pariston jännitteen mittaus) ohjelmaa. Muokkaa ohjelmaa niin, että saat luettua mittaustuloksen sekä kokonaislukuna että volteissa.

Koodi lukee potentiometrin keskimmäisen navan jännitteen ja tulostaa mittaustuloksen tietokoneelle monitori-ikkunaan

2.
Lisää kytkentälevylle vastus ja ledi. Kytke ne johonkin digitaaliseen output-pinniin. Laadi ohjelma, joka vilkuttaa lediä. Vilkkumistaajuutta voi säätää potentiometrillä.

Lisää kytkentään toinen ledi, joka vilkkuu vastakkaisessa tahdissa.

```
// ledin vilkkumistaajuuden säätö potentiometrillä

int ledi = 3;  // ledi pinniin 3

void setup() {
   pinMode(ledi, OUTPUT); // asetetaan pinni 3 OUTPUT-pinniksi
}

void loop() {
  int aika = analogRead(A0);  // luetaan analogisen pinnin A0 jännite
  digitalWrite(ledi, HIGH); // ledi syttyy
  delay(aika);    // viive, aika on potentiometriltä luettu jännite
  digitalWrite(ledi, LOW); // ledi sammuu
  delay(aika);    // viive, aika on potentiometriltä luettu jännite
}
```

Äänen tuottaminen

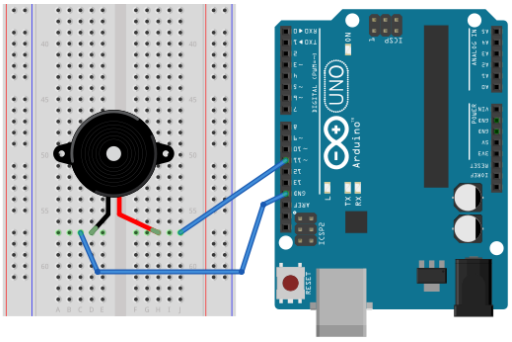
Voit tuottaa ääntä Arduinolla.

tone(pinni, taajuus, (kesto))
tuottaa äänen output-pinniin, äänen kesto ilmoitetaan millisekunneissa.

noTone(pinni)
vaimentaa äänen pinnistä

tone-käsky tuottaa kanttiaaltoa annetulla taajuudella digitaaliseen output-napaan. Kerrallaan voi antaa vain yhden äänen.

1.
Kytke pietsosähköinen kaiutin Arduinon pinneihin 11 ja GND. Kokeile koodia. Käskyä *tone* varten ei ole tarpeen asettaa pinniä OUTPUT-moodiin setup-lohkossa (eli ei tarvitse kirjoittaa *pinMode*-käskyä). Oheinen kytkentä ja ohjelma tuottaa C-duurisoinnun sävelet.



Kokeile miltä ääni kuulostaa ilman **delay**-käskyjä. Mistä tämä mahtaa johtua? Voit yksinkertaisesti ”kommentoida” delay-rivit pois lisäämällä kaksi kauttamerkkiä (//) rivin eteen. Tällöin kääntäjä hyppää rivien yli ohjelmaa ladattaessa.

```
void setup() {
}

void loop() {
  tone(11, 262, 500);  // tuotetaan 262 Hz ääni (C-sävel)
  delay(1000); // viive 1 sekunti
  tone(11, 330, 500);  // tuotetaan 330 Hz ääni (E-sävel)
  delay(1000); // viive 1 sekunti
  tone(11, 392, 500);  // tuotetaan 392 Hz ääni (G-sävel)
  delay(1000); // viive 1 sekunti
}
```

2.
”Soita” jokin yksinkertainen sävelmä Arduinolla.

Vinkki:
C-duuriasteikon sävelten taajuudet:
C 262 Hz, D 294 Hz, E 330 Hz, F 350 Hz, G 392 Hz, A 440 Hz, H 494 Hz; C 523 Hz.

(http://www2.siba.fi/akustiikka/index.php?id=18)

3.

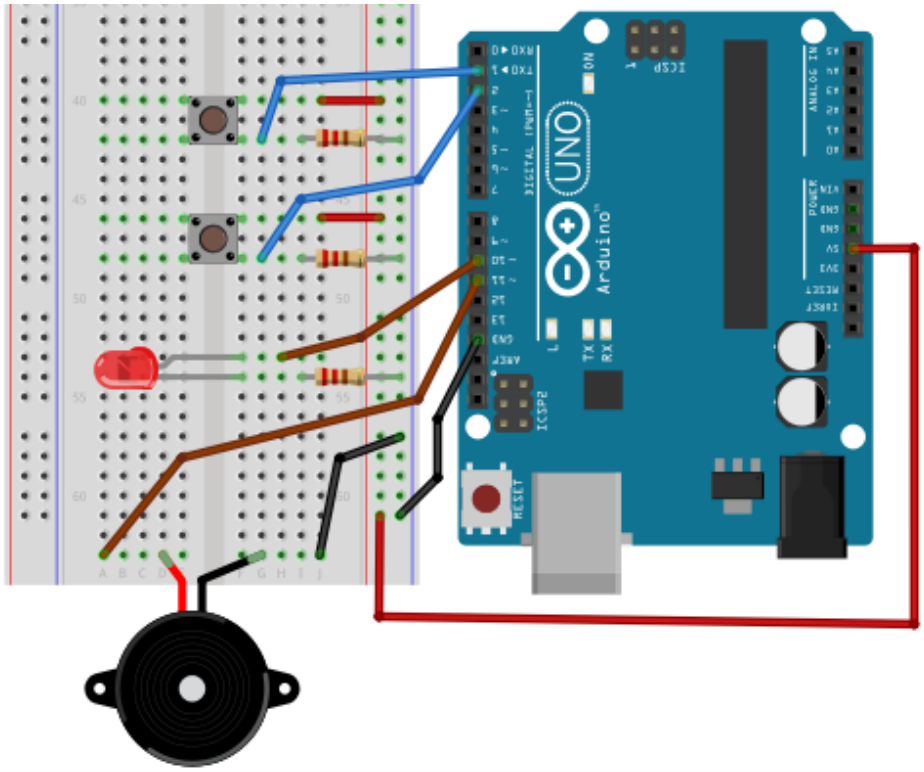
Lisää kytkentään 10 kilo-ohmin vastus ja painonappi pinniin 2 (katso esimerkkikytkentä aiemmista harjoituksista). Oheisella ohjelmalla kytkentä tuottaa äänen, kun painonappia painetaan.

```
// TONE äänen tuottaminen 2
// määritetään vakioita

const int kaiutin =  11;      // pinni, johon kaiutin kytketty
const int nappi = 2;     // pinni, johon painonappi ja vastus on kytketty
int napinTila = 0;        // muuttuja, johon napin tila luetaan (on/off)

void setup() {
  pinMode(nappi, INPUT); // asetetaan INPUT-pinni
}

void loop() {
  // luetaan napin tila:
  napinTila = digitalRead(nappi);
// mikäli nappia painetaan,
// muuttujan tila on HIGH:

  if (napinTila == HIGH) {
    // tuotetaan 300 Hz ääni
    tone(kaiutin, 300);
    }
   else
  {
    //Mikäli nappia ei paineta, on sen tila LOW
    // vaiennetaan kaiutin
    noTone(kaiutin);
  }
}
```

4.

Tee oheinen kytkentä koekytkentäalustalle. Painonappien kohdalla olevien vastusten arvot ovat 10 kΩ ja ledin kohdalla olevan 220 Ω.

Piirrä laitteen piirikaavio vihkoosi.



Tutki alla olevaa koodia. Mitä se tekee?

Tehtävä 1: Kirjoita ja syötä koodi Arduinolle.

```
// TONE äänen tuottaminen 2

// määritetään vakioita

const int nappi1 = 1;     // nappi 1
const int nappi2 = 2;     // nappi 2
const int kaiutin =  11;     // kaiutin
const int ledi =  10;   // ledi

// määritetään muuttujat kummallekin napille
// ja sijoitetaan alkuarvot (0 = LOW)

int napinTila1 = 0;        // muuttuja napin 1 tilalle
int napinTila2 = 0;        // muuttuja napin 2 tilalle


void setup() {

  // Asetetaan output-pinnit
```

```
  pinMode(kaiutin, OUTPUT);
  pinMode(ledi, OUTPUT);

  // asetetaan input-pinnit:

  pinMode(nappi1, INPUT);
  pinMode(nappi2, INPUT);
}

void loop(){
  // luetaan nappien tila:
  napinTila1 = digitalRead(nappi1);
  napinTila2 = digitalRead(nappi2);

// mikäli jompaa kumpaa nappia painetaan,
  // vastaavan muuttujan tila on HIGH:

  if (napinTila1 == HIGH) {

    // tuotetaan 300 Hz ääni ja sammutetaan ledi:

    digitalWrite(ledi, LOW);
    tone(kaiutin, 300);
  }

  else if (napinTila2 == HIGH)
  {
    // tuotetaan 450 Hz ääni ja sammutetaan ledi:

    digitalWrite(ledi, LOW);
    tone(kaiutin, 450);
  }

  else
  {
    //Mikäli nappeja ei paineta, on niiden tila LOW
    // tässä tapauksessa sytytetään ledi ja vaiennetaan
   // kaiutin

    noTone(kaiutin);
    digitalWrite(ledi, HIGH);
  }
}
```

Tehtävä: Muuta ohjelmaa niin, että syntyy erilainen ääni, kun kumpaakin nappia painetaan yhtä aikaa. Vihje: yhtäaikaisessa painamisessa lisää if-lauseen ehtoon vertailu kummankin napin arvosta.

Lämpötilan mittaaminen

TMP36 on lämpötilan mittaamiseen tarkoitettu mikropiiri, jossa on 3 liitäntänastaa. Mikropiirin uloimmat liitäntänavat liitetään Arduinon +5 V- ja GND-pinneihin (kuvassa navat 1 ja 3). Keskellä oleva output-napa (napa 2) kytketään Arduinon analogiseen input-pinniin. Mikropiirin output-navan jännite riippuu lämpötilasta lineaarisesti 10 mV/°C. Lämpötila 0°C aiheuttaa 500mV (0,5 V) jännitteen Siis 20°C aiheuttaa 700 mV (0,7 V) jännitteen. Valmistaja takaa anturin mittaustarkkuuden lämpötilavälille -40…+125 °C.

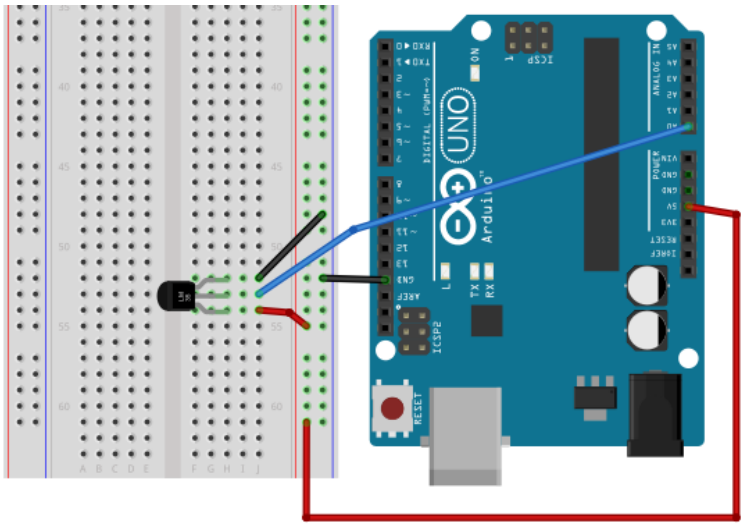
Lämpötila-anturin voi kytkeä helposti Arduinoon. Muista, että analoginen input toimii niin, että jännite 0…5 volttia muunnetaan kokonaisluvuksi 0…1023.

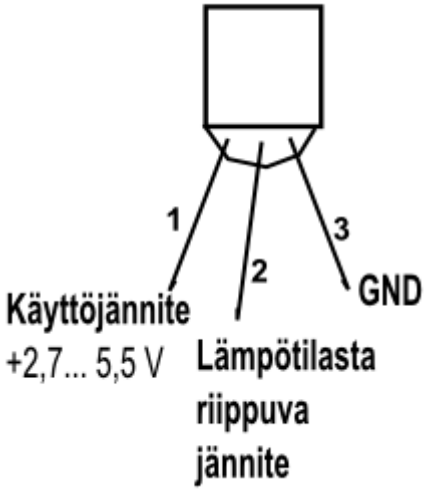
Jännitteen saat laskettua analogisen pinnin lukemasta:

$$Jännite = \frac{Lukema}{1023} * 5V$$

Jännitteestä saadaan edelleen lämpötila

$$Lämpötila = (Jännite - 0{,}5) * 100$$





Tehtävät

1. Kokoa oheinen kytkentä koekytkentälevylle**. Pidä ehdottomasti huoli, että lämpötila-anturi tulee oikeinpäin (+5 V ja GND oikein).** Muuten anturipiiri saattaa kuumentua ja rikkoutua.
2. Piirrä kytkentäkaavio vihkoosi.
3. Kokeile lämpötilan mittausta malliohjelman avulla. Lukema näkyy näytöllä Arduinon sarjamonitorilla.

```
//TMP36 lämpötila-anturin kokeilua

int anturiPinni = 0;  // Anturi on yhdistetty analogiseen pinniin 0

void setup()
{
   Serial.begin(9600);  // avataan sarjaporttiyhteys, jotta näemme lukeman näytöllä
}

void loop()
{
    int lukema = analogRead(anturiPinni);   // luetaan analogisen pinnin lukema
       float jannite = lukema * 5.0;
       jannite /= 1024.0;                          // muunnetaan lukema jännitteeksi

    Serial.print(jannite); Serial.println(" volttia");            // tulostetaan jännite

    float lampotila = (jannite - 0.5) * 100 ;  // muunnetaan jännite lämpötilaksi

   Serial.print(lampotila); Serial.println(" astetta");          //tulostetaan lämpötila

   delay(1000);                              // odotetaan sekunti
}
```

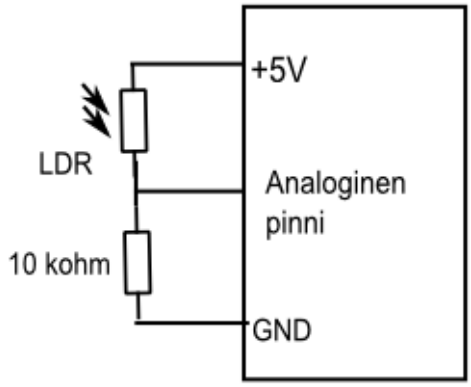
Lisätehtäviä:

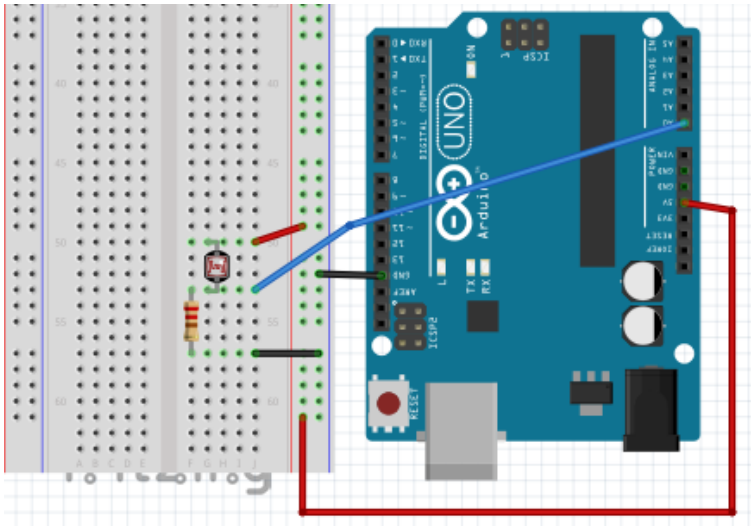
4. Lisää koekytkentälevylle pietsosähköinen kaiutin tai ledi ja 220 ohmin vastus. Muokkaa ohjelmaa niin, että saat lämpötilahälyttimen. Hälytin ilmaisee tietyn lämpötilan saavuttamisesta esimerkiksi hälytysäänellä tai vilkuttamalla lediä.
5. Muuta kytkentää ja ohjelmaa niin, että saat potentiometrillä asetettua halutun hälytyslämpötilan. Lisää potentiometri +5 V ja GND-napojen väliin, niin että luet potentiometrin liukukoskettimen jännitteen jollakin Arduinon analogisella pinnillä. Kirjoita vihkoon asiat, joita tarvitset ohjelmaa varten ja suunnittele etukäteen.
6. Etsi Internetistä TMP36 mikropiirin tiedot (TMP36 datasheet) ja tutustu mikropiirin ominaisuuksiin.

Valaistuksen mittaaminen

1.
Jännitteenjakajakytkentää voi käyttää esimerkiksi LDR-vastuksen kanssa (LDR = Light Dependent Resistor).

Mitä oheinen ohjelma tekee? Vertaa aiempaan potentiometrin lukuohjelmaan.





```
// Valaistuksen mittaaminen LDR-vastuksella.

int ledi = 13;  // Arduinolevyllä valmiina oleva ledi (pinni 13)

void setup() {
   pinMode(ledi, OUTPUT); // asetetaan pinni 13 OUTPUT-pinniksi
}

void loop() {
  int aika = analogRead(A0);  // luetaan analogisen pinnin A0 jännite
  digitalWrite(ledi, HIGH); // ledi syttyy
  delay(aika);    // viive, aika on jännitteenjakajalta luettu jännite
  digitalWrite(ledi, LOW); // ledi sammuu
  delay(aika);    // viive, aika on jännitteenjakajalta luettu jännite
}
```

2.
Kirjoita oheinen koodi ja lataa se Arduinoon. Ohjelmalla saat tulostettua sarjaportin arvon (valaistukseen verrannollisen luvun) tietokoneen ruudulle.

Mikäli käytössäsi on valaistusmittari, voit kalibroida Arduinokytkennästä oikean valaistusmittarin. Ota mallia lämpötilan mittausohjelmasta.

```
// Jännitteen lukeminen analogisesta pinnistä

void setup() {
  Serial.begin(9600); // asetetaan sarjaportin nopeudeksi 9600 bittiä sekunnissa
}

void loop() {
 // luetaan analogisen pinnin A0 jännite
 int lukema = analogRead(A0);
 // tulostetaan lukema sarjaporttiin
 Serial.println(lukema);
 delay(100);   // viive arvojen lukemiseen
}
```

3.

Lisää johonkin digitaaliseen output-pinniin ledi ja 220 ohmin vastus. Katso mallia ledin vilkutustöistä. Täydennä ohjelmaa niin, että saat tehtyä yövalaisimen. Kun valaistus pienenee riittävästi, niin ledi syttyy.

Piirrä vihkoon kytkennän piirikaavio.

Käytä ohjelman pohjana edellisessä tehtävässä annettua valaistusmittarin ohjelmaa. Muokkaa ohjelmaa niin, että saat ledin sytytettyä ja sammutettua valaistuksen mukaan. Katso tarvittaessa esimerkkiä työstä *Jännitteen mittaus* (tutkimuksen kohta 3)

Tutki, mikä olisi sopiva arvo vertailulukemaksi. Käytä apuna sarjaportin monitoria. Kun ohjelma on valmis, voit ”kommentoida” pois sarjaporttiin liittyvät ohjelmarivit (lisää //-merkit ohjelmarivin eteen, jolloin rivien yli hypätään ohjelmaa käännettäessä).

Servon ohjaaminen Arduinolla

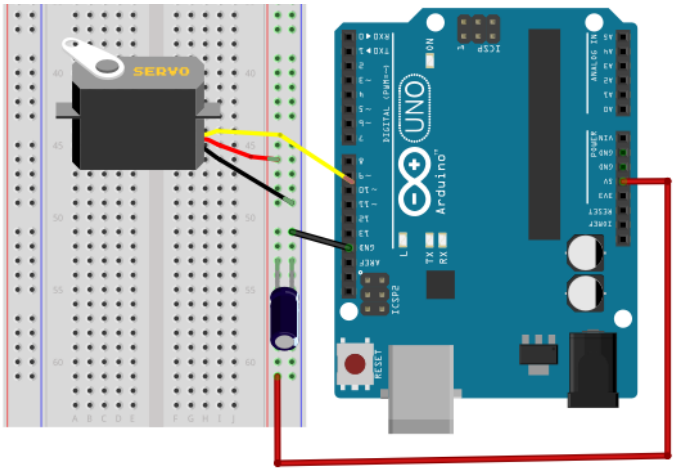
Sähköinen servo on sähkömoottorilla toteutettu ohjainlaite, jonka akselia voidaan säätää tarkasti tiettyyn kulmaan. Servon akselia voi kääntää esimerkiksi 0…180 astetta. Servoja käytetään esimerkiksi kauko-ohjattavissa laitteissa tai roboteissa liikkeiden ohjaukseen.

Servomoottorissa on tyypillisesti kolme johdinta: virta, maa ja signaali. Johtimien päissä on yleensä liitin, johon voi kokeilukytkennöissä laittaa hyppylangan pätkät Arduinoon liittämiseksi.

Tehtävät:

1.

Kytke servon virtajohdin (punainen) Arduinon +5 V-napaan, maa (musta) GND-napaan ja signaalijohto (keltainen) haluamaasi digitaaliseen output-pinniin (esimerkeissa pinni 9).



Kytke servon virtajohtojen (+5 V ja GND) väliin 100 µF elektrolyyttikondensaattori. Kondensaattori on komponentti, joka varastoi sähköä. Kun servoa käytetään, se voi välillä ottaa suuren virtapiikin Arduinon pinneistä. Tällöin pinnin jännite voi ”notkahtaa” alas ja aiheuttaa häiriötilanteen. Kondensaattoriin varastoitunut sähkö estää tämän. **Muista kytkeä elko oikeinpäin (+ ja – navat oikein!).**

Lataa oheinen ohjelma Arduinolle. Ohjelma kääntää servon akselin aluksi alkuasentoon 0 asteen kulmaan, sen jälkeen 90 ja 120 asteen kulmiin.

Ohjelmassa käytetään hyväksi valmista servokirjastoa, jossa on valmiit komennot ja funktiot servon ohjaamiseen. Kirjasto liitetään ohjelmaan alussa olevalla lauseella *#include <servo.h>*.

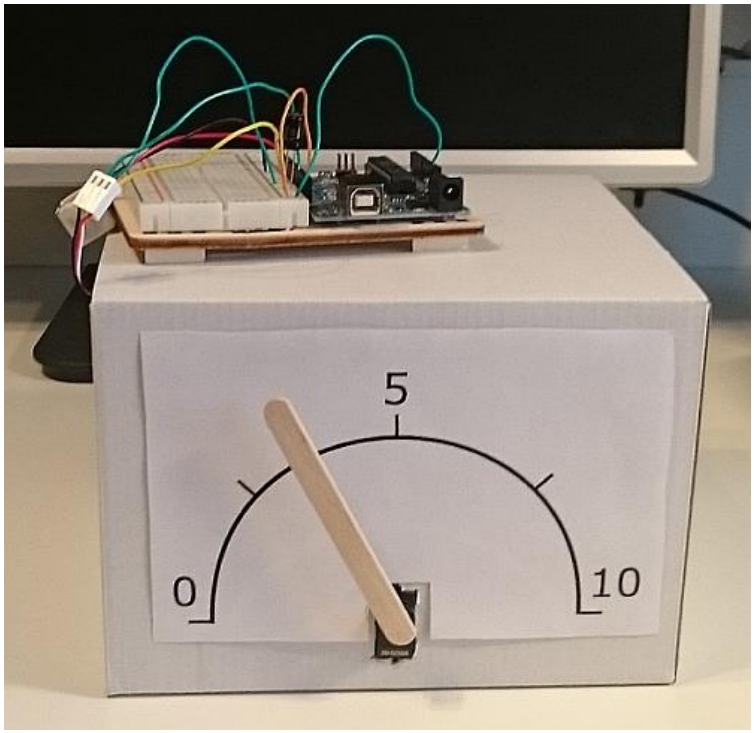
```
// Servon ohjaaminen

#include <Servo.h>          // Käytetään Servo-kirjastoa

Servo munServo;             // Määritetään objekti munServo

void setup() {
  munServo.attach(9);       // Liitetään munServo pinniin 9
}

void loop() {
  munServo.write(0);        // Ohjataan servo 0 asteen kulmaan
  delay(500);               // Odotetaan 0,5 sekuntia
  munServo.write(90);       // Ohjataan servo 90 asteen kulmaan
  delay(500);               // Odotetaan 0,5 sekuntia
  munServo.write(120);      // Ohjataan servo 120 asteen kulmaan
  delay(1000);              // Odotetaan 1 sekunti
}
```

2.
http://arduino.cc/en/Tutorial/Sweep (löytyy ohjelmaeditorin valikosta File → Examples → Servo)

Kokeile valmista Arduino-sivuston esimerkkiä *Sweep*, jossa servon akseli pyyhkäisee 180 asteen sektorin alueen edestakaisin. Esimerkissä käytetään for-silmukkaa. Tutki silmukan toimintaa Arduinon ohjesivuilta.

3.
http://arduino.cc/en/Tutorial/Knob (löytyy ohjelmaeditorin valikosta File → Examples → Servo)

Kokeile esimerkkiohjelmaa *Knob*. Tässä ohjelmassa servon asentoa säädetään potentiometrillä. Potentiometri kannattaa kiinnittää koekytkentäalustaan. Esimerkissä käytetään map-funktiota muunnokseen 0…1023 (analogRead funktion antama luku) → 0…180 (servon kääntökulma). Tutustu map-funktioon Arduinon sivuilla.



Servon avulla voi tehdä esimerkiksi analogisen näyttöpaneelin. Servosta, pahvilaatikosta ja jäätelötikusta on askarreltu viisarinäyttö. Asteikon voi tehdä suoraan paneeliin piirtämällä tai tietokoneen grafiikkaohjelmalla.

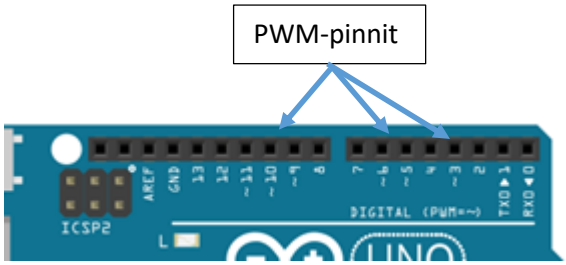
PWM Pulse Width Modulation

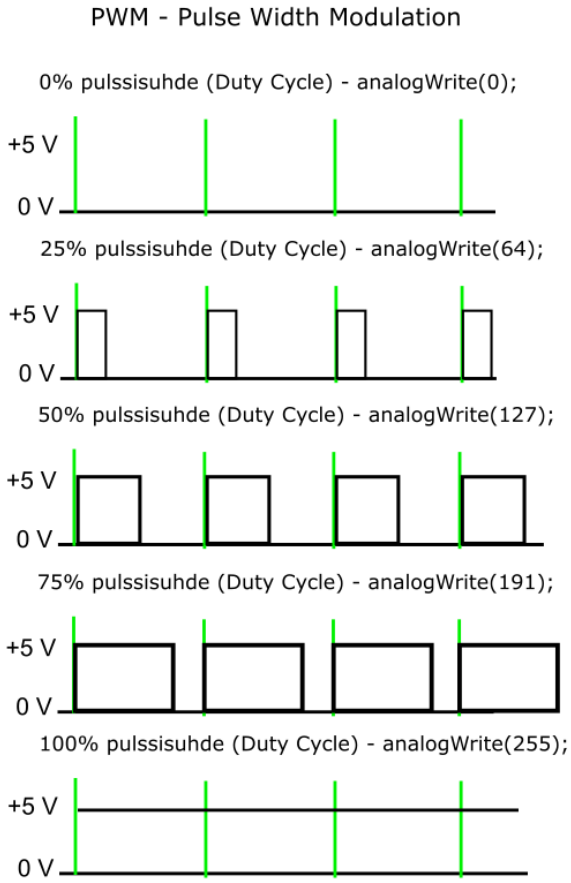
Digitaalisen output-pinnin jännite on aina 0 tai 5 volttia. Joskus olisi tarpeen säätää esimerkiksi ledin kirkkautta tai sähkömoottorin pyörimisnopeutta. Tällöin voi käyttää PWM-signaalia (PWM = Pulse Width Modulation)

Käsky *analogWrite* syöttää digitaaliselle lähtönavalle kanttiaaltoa niin, että signaalin pulssisuhdetta (Duty Cycle) voi muuttaa. Kun pulssisuhde on 50 %, niin sähköteho on vastaavasti 50 % ”heikompi” kuin jatkuva sähkövirta.

Pulssisuhde 0% vastaa 0 V jännitettä, 100% vastaa +5 V jännitettä.

PWM-pinnit on merkitty Arduinon piirilevylle aaltokuviolla ~. Arduino Unossa digitaaliset pinnit 3, 5, 6, 9, 10, 11 voivat tuottaa PWM-signaalia. PWM-signaalin taajuus on 490 Hz (joissakin pinneissä 980 Hz)





Syntaksi: ***analogWrite(pinni, arvo)***

Parametri ***arvo*** on kokonaislukumuuttuja ja sen arvo voi vaihdella välillä 0… 255. 0 vastaa pulssisuhdetta 0%, 255 pulssisuhdetta 100%.

Esim.
analogWrite(10, 64) syöttää pinniin 10 signaalia, jonka pulssisuhde on 25 %.

Tehtäviä:

Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- vastus 220 Ω, ledi (tarvittaessa useampia)
- potentiometri 10 kΩ
- vastus 10 kΩ, painonappikytkin (tarvittaessa useampia)

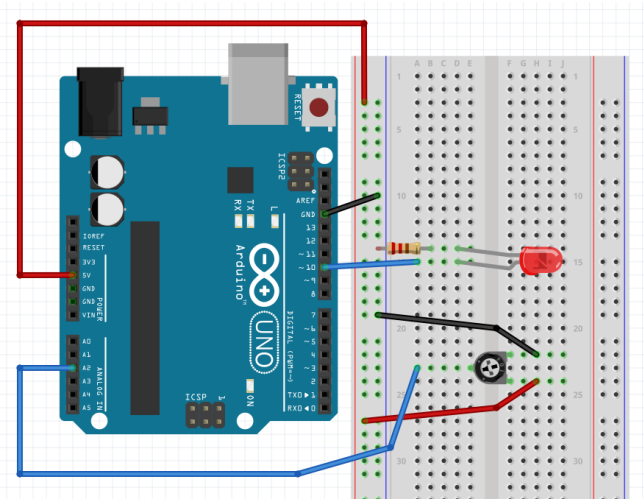
1.
Kytke ledi ja 220 ohmin vastus Arduinon pinniin 10 (tarkista, että pinnissä on PWM-merkintä ~).

Lataa Arduinolle ohjelma

```
int led = 10;

void setup() {
  // analogWrite ei vaadi pinMode asetusta
}

void loop() {

  analogWrite(led, 255);   // ledi kirkas
  delay(1000);             // odotetaan sekunti
  analogWrite(led, 130);   // ledi himmenee
  delay(1000);             // odotetaan sekunti
  analogWrite(led, 20);    // ledi himmein
  delay(1000);             // odotetaan sekunti

}
```

2.

Kytke potentiometrin äärinavat +5 V- ja GND-pinneihin. Potentiometrin liuku kytketään analogiseen pinniin A2.



Laadi ohjelma, jolla voit säätää portaattomasti ledin kirkkautta potentiometrillä säätämällä. Avaa esimerkiksi *AnalogReadSerial* – ohjelmaesimerkki ja aloita ohjelman muokkaus siitä (vaihda A0 → A2). Voit avata myös sarjaportin monitorin, jolla voit tarkistaa muuttujien arvot.

Mieti:
Potikkaa luettaessa funktio *analogRead* antaa tuloksen 0…1023.
PWM-pinniin vaaditaan kuitenkin arvo 0…255
Miten saadaan muutettua 0…1023 → 0…255?

Eräs mahdollinen ohjelmaesimerkki on seuraava:

```
int Ledi = 10;      // Ledi kytketään pinniin 10
int Potikka = 2;   // Potentiometrin jännite luetaan analogisesta pinnistä 2 (0… 5 V)
int arvo = 0;        // Muuttuja, johon potikan lukema talletetaan ( kokonaisluku 0… 1023)

void setup()
{
}

void loop()
{
  arvo = analogRead(Potikka);   // Luetaan potikan arvo
  analogWrite(Ledi, arvo/4);  // Muutetaan potikan arvo 0… 1023 analogWrite-käskylle sopivaksi 0 … 255
}
```

3.
Lisää toinen ledi ja 220 ohmin vastus toiseen PWM-pinniin (esim. pinni 11). Muuta kohdan 2 ohjelmaa niin, että ledien kirkkaus muuttuu päinvastaisesti, siis toisen loistaessa kirkkaasti, toinen ei pala. Potentiometristä säätämällä toinen himmenee ja toinen kirkastuu. Mieti: Miten saat 0…255 → 255…0?

4.
For-silmukka
Ledin himmentäminen for-silmukan avulla

Säilytä edellisen esimerkin ledikytkentä ja kokeile ohjelmaa:

```
// Ledin himmentäminen FOR-silmukan avulla
int Ledi = 10; // asetetaan muuttujan Ledi arvoksi 10

void setup()
{
}

void loop()
{
  for (int i=0; i <= 255; i++) {
    analogWrite(Ledi, i);
    delay(10);
  }
}
```

Selvitä itsellesi ohjelman toiminta. Kokeile, mitä tapahtuu, kun vaihdat käskyksi **analogWrite(Ledi, 255-i);**

5.

Lisää kytkentään painonappi ja vastus. Katso aiemmista tutkimustehtävistä mallia.

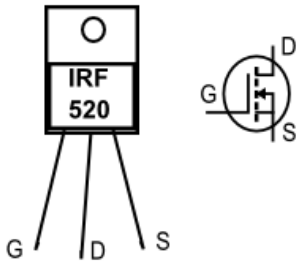
Muokkaa ohjelmaa niin, että painonappia painamalla saat ledin himmentymään/kirkastumaan.

Voit lisätä myös useampia kytkimiä ja ledejä Arduinon napoihin ja laatia niille toimintoja. Lisää kytkentään esimerkiksi kaksi painonappia, joista ensimmäistä painamalla ledi kirkastuu, toista painamalla ledi himmenee.

Moottorin ohjaaminen Arduinolla

Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- MOSFET IRF520
- potentiometri 10 kΩ
- diodi (esim. 1N4005)
- 9 V paristo ja paristoneppari (tai vastaava virtalähde)



Transistorin IRF520 johdinjärjestys

1. Tee oheinen kytkentä, kirjoita ohjelma ja kokeile toimintaa.
Moottorin rinnalle on kytketty diodi. Diodin tehtävänä on oikosulkea mahdolliset jännitepiikit, joita moottori aiheuttaa ja jotka voisivat vioittaa Arduinoa.

2. Mikäli sinulla on käytössä Arduino Starter Kit, aseta siinä oleva vanerinen CD-levyn pyörittäjä moottorin akselille. Starter Kit-sarjassa on myös värillinen pahvilevy, jota voi pyörittää moottorilla. Mikäli sinulla ei ole Starter kittiä, laita esimerkiksi maalarinteipin pätkä moottorin akselille, niin näet moottorin pyörimisen.

3. Kokeile moottorin pyörittämistä ensin pelkästään ohjelmallisesti, ilman nopeudensäätöä potentiometrillä. Kertaa sivulta 40 PWM-säädön periaate. Kirjoita koodi ja syötä se Arduinolle.

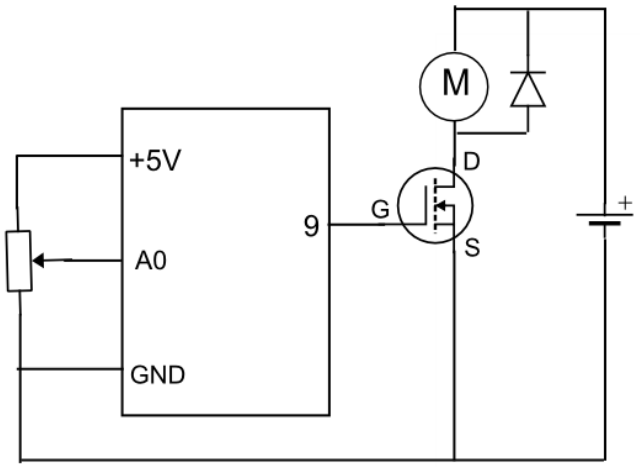
```
// Moottorin nopeuden säätö ohjelmallisesti

 const int transistoriPin = 9;  // MOSFETin ohjauspinni
 int nopeus = 64;               // säädetään nopeus

 void setup() {
  pinMode(transistoriPin, OUTPUT); // Asetetaan OUTPUT-pinni
 }

 void loop() {
  analogWrite(transistoriPin, nopeus);
   // Kirjoitetaan PWM-ohjaus transistorille
 }
```

Kokeile muuttujalle **nopeus** arvoja välillä 0…255.

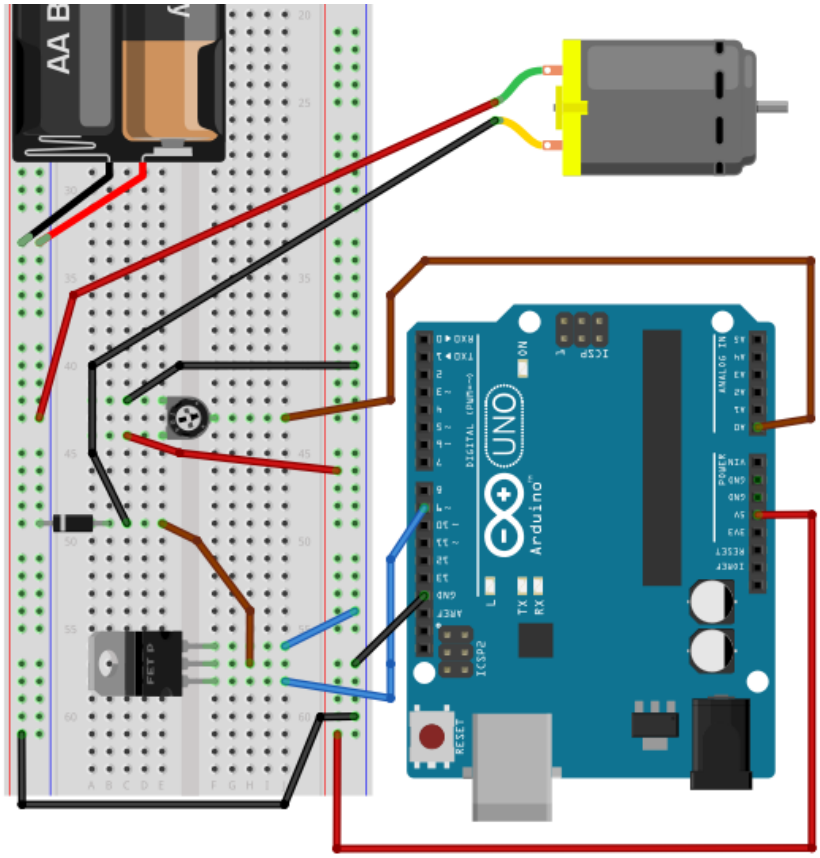
4. Tutki moottorin säätöä potentiometrillä. Muuta koodia niin, että Arduinon analoginen pinni lukee potentiometrin liukukoskettimen jännitettä. Jännitettä voi säätää 0… 5 voltin välillä..

```
// Moottorin nopeuden säätö

const int potikkaPin = 0; // Potentiometrin liuku analogiseen pinniin 0
const int transistoriPin = 9;  // MOSFETin ohjauspinni
int potikka;   // Potentiometriltä palautettu arvo

 void setup() {
  pinMode(transistoriPin, OUTPUT); // Asetetaan OUTPUT-pinni
}

void loop() {
  potikka = analogRead(potikkaPin)/4;
    // Luetaan potentiometri ja muutetaan samalla arvo sopivaksi PWM-funktiota varten
  analogWrite(transistoriPin, potikka);
    // Kirjoitetaan PWM-ohjaus transistorille
}
```

Voit kokeilla kytkentää myös lampun ohjaamiseen. Esimerkiksi valo-opin ilmiöiden tutkimiseen käytettävä 6 voltin lamppu sopii hyvin kytkentään. Lamppu kytketään moottorin tilalle. Lampun kanssa suojadiodin voi jättää pois.



Moottoriin on kytketty ulkoinen virtalähde, esimerkiksi 9 V neppariparisto. Moottori ei siis toimi ”Arduinon sähkövirralla”. Mikäli ohjaat kytkennällä valo-opin tutkimuksissa käytettyä suurivirtaista lamppu, niin käytä lampun virtalähteenä oppilasvirtalähdettä. Pidä huoli, että transistori ei tällöin kuumene liikaa.



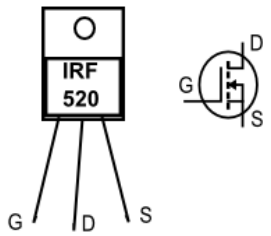
Tehtävät:

1. Miksi potentiometriltä luettu arvo jaetaan neljällä?

2. Transistoria ohjataan PWM-signaalilla. Mitä tämä tarkoittaa?

Arduinon pinnin antama maksimi sähkövirta on noin 40 mA. Tämä virta ei riitä käyttämään suuritehoisia laitteita, kuten sähkömoottoria tai suuritehoisia lamppuja.

Arduinoon voidaan kuitenkin kytkeä ulkoinen komponentti, joka puolestaan säätää moottorissa kulkevaa virtaa. Ohjauskomponenttina voidaan käyttää esimerkiksi transistoria. Esimerkkikytkennässä käytetään FET-transistoria, joka ei vaadi muita komponentteja ympärilleen.



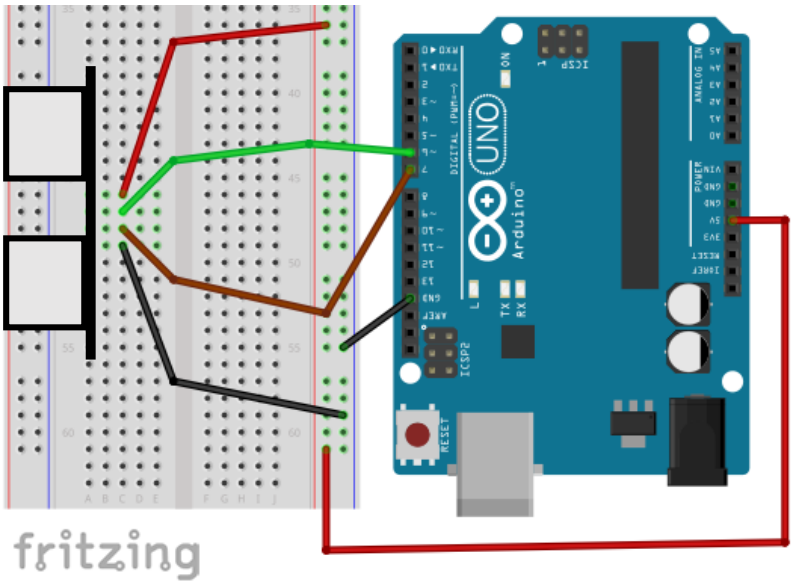
IRF520 on kytkintarkoituksiin valmistettu MOSFET, jonka johdinjärjestys on oheisessa kuvassa (G = Gate, hila, D = Drain, S = Source).

Transistoria voi verrata äärimmäisen nopeaan kytkimeen, jossa ei ole lainkaan liikkuvia osia. Hilan (G) jännitteellä voidaan vaikuttaa D- ja S-napojen välillä kulkevaan sähkövirtaan. Kun jännitettä ei ole, virtaa ei kulje, kun hilalla on jännite, niin virta kulkee.

Ultraäänianturin liittäminen Arduinoon

Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- Ultraäänianturi HC-SR04 (anturi mittaa etäisyyksiä välillä 2… 400 cm)
- Komponentteja hälytintä varten (esim. pietsosähköinen kaiutin, ledi + 220 Ω vastus tms.)



Tehtävät:

1.

Tee oheinen kytkentä koekytkentälevylle.

Anturissa on neljä kytkentäpinniä: *+5 V, GND, Trig* ja *Echo*. Pinneihin +5 V ja GND kytketään käyttöjännite. Pinniin *Trig* johdetaan kytkentäliipaisu (*Trigger = liipaisin*). Pinnistä *Echo* (*Echo = kaiku*) saadaan esineen etäisyyteen verrannollinen pulssi.

Trigger kytketään digitaaliseen pinniin 6, *Echo* pinniin 7.

2.

Ohjelmassa käytetään kirjastoa, jossa on valmiit komennot ja funktiot ultraäänianturin ohjaamiseen. Tätä kirjastoa ei ole valmiiksi asennettuna Arduinoympäristöön.

Kirjasto täytyy asentaa erikseen.

Hae kirjasto osoitteesta http://playground.arduino.cc/Code/NewPing --> Download.

Manuaalisesti kirjaston voit asentaa purkamalla kansion Arduinoympäristön Libraries-kansioon. Sammuta editoriohjelma ja käynnistä uudelleen. Ohjelma etsii uuden kirjaston ja se on Arduino-ohjelmien käytettävissä.

3.

Kirjoita editoriin oheinen ohjelma.

Kirjasto liitetään ohjelmaan alussa olevalla lauseella *#include <NewPing.h>*. Mittaustulokset luetaan tietokoneen sarjaportin monitorista.

Malliohjelmassa on määritelty vakioita #define lauseilla.

```
// Ultraäänianturi

#include <NewPing.h>          // otetaan kirjasto NewPing.h käyttöön
NewPing sonar(6, 7, 200); // Alustetaan NewPing
// pinni 6 trigger output, pinni 7 echo input, maksimietäisyys 200 cm

void setup() {
  Serial.begin(9600);                    // Avataan sarjaportti
}

void loop() {
  delay(1000);                           // Etäisyyden mittaus sekunnin välein
  int etaisyys = sonar.ping_cm();        // Mitataan etäisyys senttimetreinä
  Serial.print("Etäisyys: ");            // Tulostetaan näytölle
  Serial.print(etaisyys);
  Serial.println(" cm");
}
```

4.

Tarkista anturin mittaustarkkuus metrimitan avulla. Aseta mitta niin, että nollakohta on anturin kohdalla ja heijastukset tapahtuvat jostakin kovasta pinnasta, esimerkiksi kirjasta. Ovatko mittaustulokset oikeita?

Tutustu kirjaston dokumentaatioon osoitteessa http://playground.arduino.cc/Code/NewPing. Kirjaston komennoista ja funktioista käytetään nimitystä *metodi*.

5.

Laajenna kytkentää niin, että saat jonkin toimivan laitteen, esimerkiksi hälytyksen ledin tai summerin avulla. Voit tehdä peruutustutkan mopolle!

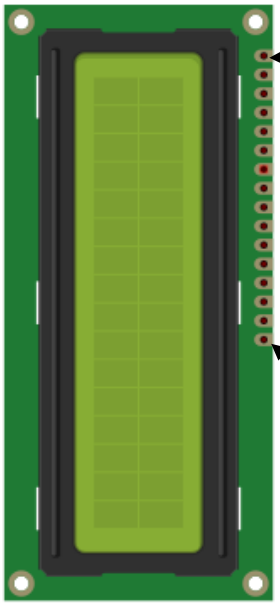
Huom.: Arduinon *tone*-käsky ei toimi NewPing kirjaston kanssa. Etsi hakupalvelun avulla kirjasto NewTone, jonka avulla saat äänimerkin toimimaan.

LCD-näytön liittäminen Arduinoon

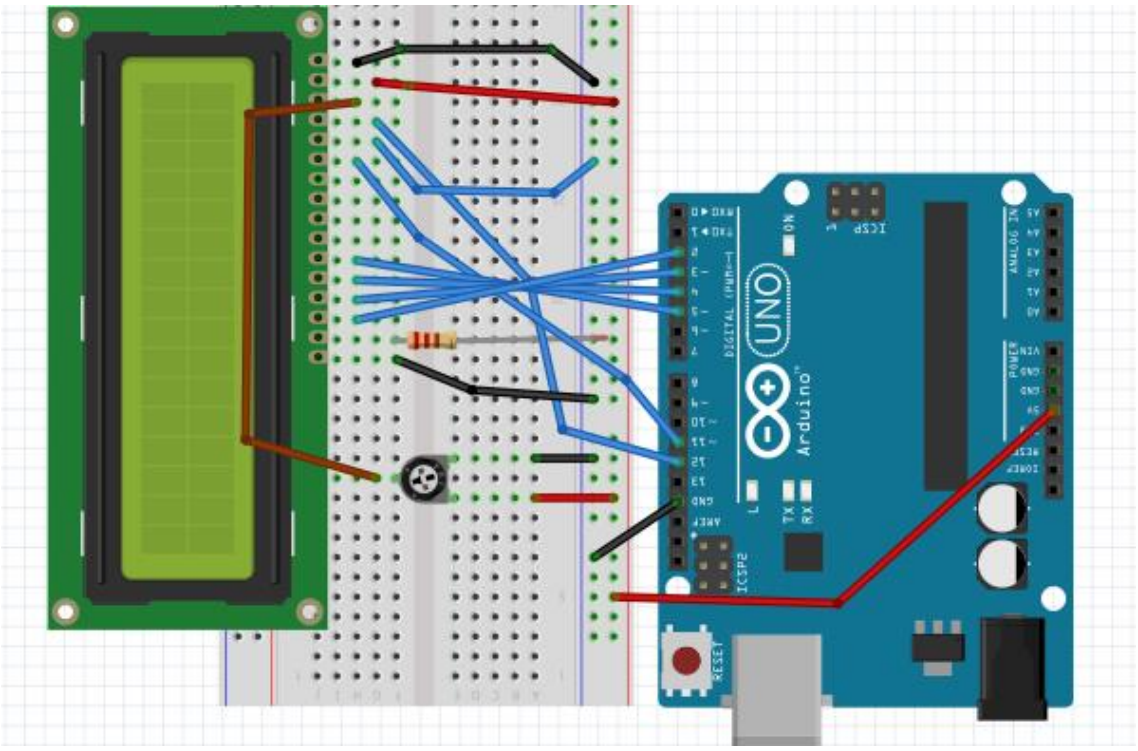
Välineet

- Arduino Uno, koekytkentäalusta ja hyppylankoja
- LCD-näyttö (LCD tulee sanoista *Liquid Crystal Display* eli nestekidenäyttö)
- potentiometri 10 k Ω, 220 Ω vastus, lämpötila-anturi TMP36

Arduinoon voi liittää LCD-näytön. Näytössä on 16 liitäntäpinniä, joista näyttöä ohjataan.



LCD pinni	kytketään
1 VSS	Arduino GND
2 VDD	Arduino +5 V
3 kontrasti	Potentiometrin liuku
4 Register Select (RS)	Arduino pinni 12
5 Read/Write (RW)	GND
6 Enable	Arduino pinni 11
7 Data 0	Ei kytketä
8 Data 1	Ei kytketä
9 Data 2	Ei kytketä
10 Data 3	Ei kytketä
11 Data 4	Arduino pinni 5
12 Data 5	Arduino pinni 4
13 Data 6	Arduino pinni 3
14 Data 7	Arduino pinni 2
15 Taustavalo +	220 ohm vastus +5 V
16 Taustavalo GND	Arduino GND



1.
Kytke LCD-näyttö huolellisesti. Aloita liitinrivin jommastakummasta päästä ja etene liitin kerrallaan. Kun USB-kaapeli kytketään Arduinoon, niin taustavalon tulisi syttyä näyttöön. Potentiometristä säädetään näytön kontrastia. Mikäli näytössä ei näy mitään, tarkista potentiometrin liitäntäpinnit.

LCD-näyttöä varten ohjelmaan ladataan kirjasto **LiquidCrystal**. Kirjasto on oletuksena jo asennettu Arduinon editoriohjelman mukana.

Kokeile ohjelmanpätkää:

```
// LCD-näytön kokeilua

// otetaan LCD-kirjasto käyttöön
#include <LiquidCrystal.h>

//määritetään ohjelmalle käytössä olevat Arduinon pinnit
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  //asetetaan LCD-näytön rivien ja sarakkeiden lukumäärä
  lcd.begin(16, 2);
}

void loop() {
  lcd.print("hello, world!"); //tulosta näytölle teksti

// tulostetaan luvut nollasta kymmeneen näytön toiselle riville
for (int i=0; i<=10;i++) {
   lcd.setCursor(0, 1); // aseta kursori kakkosrivin alkuun (numerointi alkaa nollasta)
  lcd.print(i); //tulostetaan muuttujan i arvo
  delay(500); // odotetaan 0,5 sekuntia
   }

lcd.clear(); // tyhjennetään näyttö

}
```

2.
Tutustu *LiquidCrystal*-kirjaston dokumentaatioon (*https://www.arduino.cc/en/Reference/LiquidCrystal*) ja kokeile valmiita harjoitusohjelmia.

3.
Selvitä, mitä ovat kirjastosta löytyvät *metodit.*

4.
Kokeile lisätä kytkentään jokin anturi (esimerkiksi lämpötila-anturi) ja tutki miten saat anturin lukeman LCD-näytölle.

Oman funktion tekeminen

Toistuvia toimintoja voit koota kätevästi aliohjelmiksi (funktioiksi). Toiminnot ohjelmoidaan erilleen varsinaisesta pääohjelmasta (loop-silmukka), jolloin vältytään saman ohjelmakoodin kirjoittamisesta uudelleen ja uudelleen. Kukin lohko, aliohjelma, tekee jonkin toiminnon, esimerkissä kääntää servon haluttuun asentoon, antaa halutun äänimerkin tai käy lukemassa anturin mittausarvon. Aliohjelmasta käytetään myös nimityksiä proseduuri, funktio, metodi tai rutiini. C-kielessä, johon Arduinon koodi perustuu, käytetään yleisesti nimitystä funktio.

Tutki oheisia ohjelmaa ja lataa ne Arduinolle.

1.
Servon ohjaus:
Ohjelmalla ohjataan servoa ja ohjelman toiminnot on sijoitettu aliohjelmiin. Sana *void* funktion määrittelyssä tarkoittaa, että funktio ei palauta pääohjelmalle mitään arvoa. Funktiolle syötetään muuttujiksi kokonaislukumuuttujat *kulma* ja *aika*.

Kytkentä on sama kuin aiemmassa servotutkimuksessa.

Kokeile muuttaa funktiolle *kaannaServo* annettavien muuttujien arvoja. (Huom.: kulman arvon tulee olla välillä 0…180 astetta. Aika ei voi olla kovin nopea, johtuen servon hitaudesta)

```
// Servon ohjaaminen omalla funktiolla

#include <Servo.h>          // Käytetään Servo-kirjastoa

Servo munServo;             // Määritetään objekti munServo


void setup() {
  munServo.attach(9);       // Liitetään munServo pinniin 9
}

// Määritetään funktio kaannaServo, jolle annetaan muuttujat kulma ja aika

void kaannaServo(int kulma, int aika) {
  munServo.write(kulma);    // Ohjataan servo muuttujassa kulma määritettyyn kulmaan
  delay(aika);              // Odotetaan muuttujassa aika määritetty aika
}

// Pääohjelma alkaa

void loop() {
   kaannaServo(0, 1000);    // Käännetään servo haluttuun kulmaan
   kaannaServo(45, 1000);   // ja odotetaan annettu aika
   kaannaServo(90, 2000);
   kaannaServo(135, 500);
   kaannaServo(180, 400);
}
```

1.
Äänen tuottaminen
Seuraava ohjelma tuottaa nousevan äänen Arduinon pinniin 11. Aliohjelma **nousu** on laadittu niin, että sitä kutsuttaessa annetaan äänen alku- ja lopputaajuudet eli ns. parametrit. Parametri tarkoittaa käskylle annettavia alkuarvoja, joiden mukaan toiminto suoritetaan.

Kytke pietsokaiuttimen navat Arduinon pinniin 9 ja GND-pinniin ja lataa oheinen ohjelma.

```
void setup() { // asetuksia ei vaadita
}

void nousu(int alku, int loppu)          // aliohjelma, parametreinä alku- ja lopputaajuus
{
  for (int taajuus=alku; taajuus < loppu; taajuus++) {
    tone(11, taajuus, 50);               // silmukkaa pyöritetään, kunnes saavutetaan lopputaajuus
    delay(50);
  }
}


void loop() {                            // pääohjelma
  nousu(100, 600);                       // kutsutaan funktiota nousu, 100 Hz - - > 600 Hz
  nousu(300, 800);                       // kutsutaan funktiota nousu
  nousu(500, 1000);                      // kutsutaan funktiota nousu
}
```

2.
Lisää ohjelmarunkoon funktio nimellä **lasku**, joka tuottaa laskevan äänen (vinkki for-silmukkaan: i- - tarkoittaa i = i - 1).

Kokeile **nousu-** ja **lasku**-funktioita peräkkäin niin, että kutsut niitä pääohjelmasta. Toimiiko?

Muokkaa aliohjelmia niin, että lisäät funktiolle annettavaksi parametriksi pinnin, johon kaiutin on kytketty. Kysy tarvittaessa neuvoa.

3.
Tee kytkentä ja ohjelma, jossa ”simuloit” hälytysajoneuvoa (”piipaa-auto”). Lisää kytkentään ledi ja 220 ohmin vastus (tai useampia), niin saat vilkkuvalot.

Ohjelman voit laatia niin, että

a) koko ohjelma on pääohjelmassa (loop-silmukka),
b) ohjelma on jaettu aliohjelmiin, joita kutsutaan pääohjelmasta.

Kokeile kumpaakin tapaa!

Oman laitteen tekeminen – projektityö

Voit yhdistellä tässä vihkosessa esitettyjä aihioita helposti. Suunnittelun apuna voi käyttää alla olevaa listaa, jossa on kuvattu Arduinolle ohjelmoitavia toimintoja.

- ledivilkkusysteemit
- painonappien kytkeminen
- äänen tuottaminen
- jännitteen mittaaminen (lämpötilan mittaus, valaistuksen mittaus)
- servon ohjaus
- moottorin ohjaus
- PWM - tehon säätäminen (ledin himmentäminen /moottorin kierrosluvun säätäminen)
- etäisyysmittaus ultraäänianturilla
- LCD-näytön kytkeminen

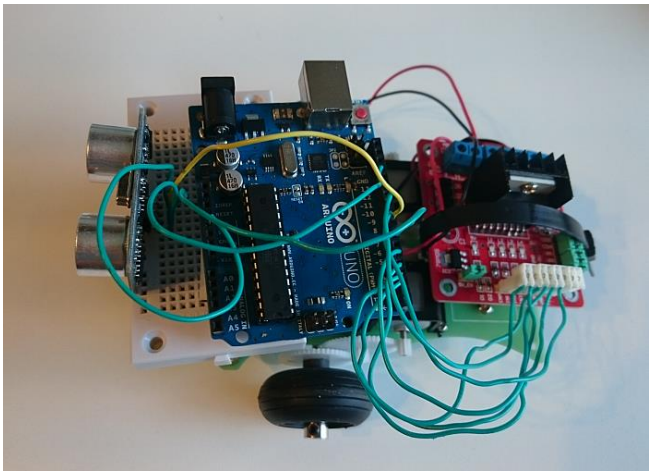
PROJEKTITYÖ

- peli?
- mittalaite?
- hälytin?
- laskuri?
- ajastin?
- soitin?
- valoefektilaite?
- joku muu?

Voit tehdä esimerkiksi lämpömittarin, jossa on LCD-näyttö. Lämpömittariin voit lisätä ledin tai summerin hälytystä varten. Lämpömittarista voit tehdä myös vaikkapa viisarinäyttöisen (servo + osoitinviisari) tai voit toteuttaa näytön ledipatsaan muotoon. Lämpömittari voi sähköttää lukeman Morse-aakkosilla. Mahdollisuuksia on paljon!

Ideoita kannattaa etsiä Internetistä. Erityisesti esim.

- Arduinon omat sivut www.arduino.cc
- sivusto www.instructables.com (sivustolla on paljon Arduinoon liittyviä laitteita, hakusana Arduino)
- youtube (hakusanana Arduino + haettava asia, esim. arduino led cube)



Laitteen mekaaninen toteutus on usein haastavin osa projektia. Internetin kauppapaikoilta löytyy esimerkiksi valmiita runkoja liikkuvaa robottiautoa varten. Valmis runko helpottaa huomattavasti laitteen mekaanista toteutusta.

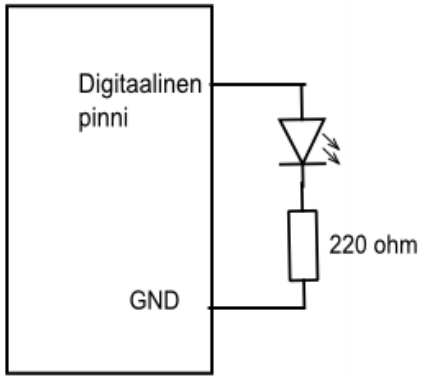
Vihkosen lopussa on ohjeita projektityön toteutukseen.

Kertausta: peruskytkennät

Ledi OUTPUT-pinniin

Esim.:

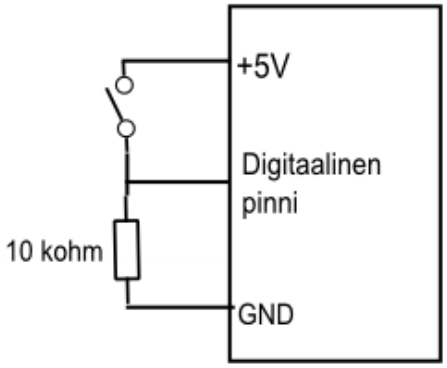
```
…
loop() {
digitalWrite(led, HIGH);
}
```



Painonappi INPUT-pinniin

Esim.:

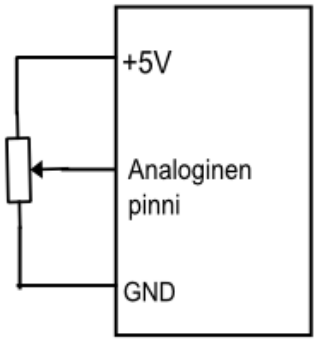
```
…
void loop(){
  nappi = digitalRead(nappiPin);
  if (nappi == HIGH) {
    digitalWrite(ledPin, HIGH);
     }
  else {
   digitalWrite(ledPin, LOW);
    }
}
```

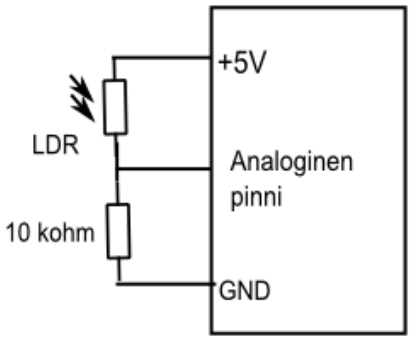


LDR tai potentiometri analogiseen input-pinniin

Esim.:

```
…
void loop()
{
    int lukema = analogRead(anturiPinni);
    Serial.println(lukema);
   delay(1000);
}
```





Ohjelman ohjausrakenteita

Ohjausrakenne	Esimerkki

if
Mikäli ehto on tosi, niin lause toteutetaan.
 	
Vertailuoperaattorit
x == y (x on yhtäsuuri kuin y)
x != y (x on erisuuri kuin y)
x < y (x pienempi kuin y)
x > y (x on suurempi kuin y)
x <= y (x on pienempi tai yhtäsuuri kuin y)
x >= y (x on suurempi tai yhtäsuuri y)
	if (muuttuja > 50)
	{
	  // tee jotakin
	}

if – else
 	
Mikäli ehto on tosi, niin toiminto A toteutetaan. Muussa tapauksessa toteutetaan toiminto B.
	if (muuttuja < 500)
	{
	  // toiminto A
	}
	else
	{
	  // toiminto B
	}

if – else if – else
 	
If-lauseeseen voidaan lisätä useampia ehtoja.
 	
Else if –lohkoja voi olla useita. Viimeisen else-lauseen voi jättää poiskin.
	if (muuttuja < 500)
	{
	  // toiminto A
	}
	else if (muuttuja >= 1000)
	{
	  // toiminto B
	}
	else
	{
	  // toiminto C
	}

switch / case
 	
Kun muuttujan arvo täsmää case-ehdon arvoon, vaihtoehto toteutetaan.
Vastaava rakenne voidaan tehdä myös if-lauseella, mutta case-rakenne on monissa tapauksissa kätevämpi.
 	
Case-toiminnosta poistutaan avainsanalla **break.**
	switch (muuttuja) {
	  case 1:
	  // toiminto A, kun muuttuja on 1
	  break;
	  case 2:
	  // toiminto B, kun muuttuja on 2
	  break;
	  default:
	  // mikäli mikään vaihtoehto ei toteudu
	  // ”default” on valinnainen (voi jättää pois)
	}

Toistolauseet

Toistolause	Esimerkki

for
 	
Toistaa toimintoa halutun määrän
 	
for (muuttujan alustaminen; ehto; muuttujan kasvattaminen) {
// toiminnot
}
 	
Huom.:
Merkintä i++ tarkoittaa i = i +1 eli muuttujan arvoon lisätään 1.
	int PWMPin = 9;
	
	for (int i=0; i <= 255; i++){
	  analogWrite(PWMpin, i);
	  // kirjoitetaan pinniin 9
	  // PWM-signaalia
	  delay(10);
	  }

while
 	
Toistaa toimintoa, kunnes annettu ehto on epätosi.
 	
while(ehto){
  // toiminnot
}
	muuttuja = 0;
	while(muuttuja < 200){
	  // toiminto
	  muuttuja++;
	}
	// silmukka lopetetaan, kun toiminto
	// on tapahtunut 200 kertaa

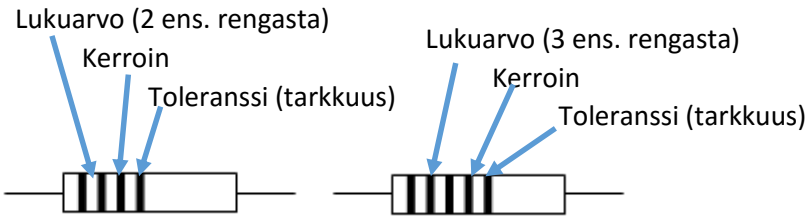
do - while
 	
Toistaa toimintoa, kunnes annettu ehto on epätosi (samaan tapaan kuin **while**). Ehdon testaus on lauseen lopussa, jolloin toiminto suoritetaan ainakin kerran.
 	
do
{
  // toiminnot
} while (ehto);
	do
	{
	  delay(50);
	  x = analogRead(0); // luetaan
	  //analoginen pinni
	  // muuttujaan x
	} while (x < 100);
	  // lopetetaan silmukka, kun luettu
	  // arvo on alle 100

Vastusten värikoodi

0 = musta
1 = ruskea
2 = punainen
3 = oranssi
4 = keltainen
5 = vihreä
6 = sininen
7 = violetti
8 = harmaa
9 = valkoinen

Toleranssi (tarkkuus):

1 % = ruskea
2 % = punainen
5 % = kulta
10 % = hopea



Vastuksen ympärillä on neljä tai viisi värirengasta.

1. ja 2. rengas (tai 1., 2. ja 3.) tarkoittavat vastuksen resistanssin lukuarvon kahta ensimmäistä numeroa. 3. rengas (tai 4.) on lukuarvon kerroin (kymmenen potenssi, nollien lukumäärä). 4. rengas (tai 5.) on vastuksen toleranssi eli tarkkuus.

Esim.

ruskea - 1	musta - 0	punainen - kulta 2	R = 1000 Ω = 1 kΩ , ± 5 %
keltainen - 4	violetti - 7	keltainen - kulta 4	R = 470 000 Ω = 470 kΩ, ± 5 %
punainen - 2	punainen - 2	musta - hopea 0	R = 22 Ω, ± 10 %

Vastuksen resistanssi

Lukeminen aloitetaan värirenkaasta, joka on lähempänä vastuksen päätä. Vastuksen resistanssi ilmoitetaan joko kolmella tai neljällä renkaalla, viimeinen rengas ilmoittaa vastuksen tarkkuuden (toleranssin). 5 % tarkkuus tarkoittaa, että 1000 Ω:n vastuksen todellinen resistanssi voi olla välillä 950 Ω – 1050 Ω

Tehtäviä

A.
Mikä on seuraavien vastusten resistanssi? Entä tarkkuus?

1. keltainen - violetti - punainen - kulta

2. ruskea - musta – sininen - hopea

3. oranssi - valkoinen - oranssi - kulta

4. punainen - sininen - keltainen - oranssi - ruskea

5. oranssi - oranssi - musta - hopea

B.
Tutki muutaman vastuksen värirenkaiden perusteella, mikä on vastusten resistanssi. Mittaa sitten resistanssin arvo digitaalisella yleismittarilla. Vertaa tuloksia, ovatko vastusten resistanssit tarkkoja (vastaako mitattu resistanssi värikoodilla ilmaistua arvoa).

Projektityö

- Mitä laite voisi tehdä?
    - Onko se peli, mittalaite, hälytin, laskuri, ajastin, soitin, valoefektilaite vai joku muu?
    - **Suunnittele paperilla** laitteen toimintaa ja **kirjaa ylös** haluamasi ominaisuudet
    - Keksi laitteelle nimi

- Listaa tarvitsemasi osat

- **Kokoa kytkentä** kytkentäalustalle ja **laadi ohjelma** laitteistolle.
    - Ohjelman laatiminen kannattaa aloittaa jostakin valmiista malliohjelmasta. Jos ohjelmalla esimerkiksi luetaan lämpötila-anturia, niin aloita lataamalla valmis malliohjelma ja muokkaat sitä haluamaasi muotoon.
    - Muista, että **kaikkea ei tarvitse tehdä kerralla**. Voit koota laitetta lohko kerrallaan ja samalla tarkistaa toiminnan. Tällöin myös opit parhaiten ohjelmointia ja elektroniikkaa!
    - Ota **valokuvia** työn edistymisestä.

- Laadi **loppuraportti** pdf-tiedostona, jonka palautat sovitulla tavalla. Raportin tulee sisältää:
    - Laitteen **toiminnan kuvaus** sanallisesti, **valokuvia** työn vaiheista ja valmiista laitteesta
    - **Kytkentäkaavio** (voi tehdä esimerkiksi jollakin grafiikkaohjelmalla tai yksinkertaisesti piirtämällä paperille ja valokuvaamalla/skannaamalla)
    - **Ohjelmalistaus**
    - **Pohdintaa** työn onnistumisesta, työssä havaituista ongelmista, kehitysehdotuksia laitteen toiminnalle

Harjoitustyöt

Kussakin tutkimustyössä on alussa perusasiat ja sen jälkeen kokonaisuuteen liittyviä syventäviä tehtäviä ja ohjelmointiharjoituksia. Voit merkitä taulukkoon tekemäsi tutkimukset. Voit myös arvioida työskentelyäsi ja kirjata muistiin oman arviosi työskentelyn onnistumisesta.

Listan lihavoidut tutkimustehtävät (ledin vilkuttelu, painonapin lukeminen, jännitteen mittaaminen) on syytä käydä huolella läpi. Näissä töissä käytetään Arduinon keskeisiä ominaisuuksia, digitaalinen tulo ja lähtö, analogisen jännitteen lukeminen. Muutoin tehtävien järjestys on vapaa.

Tutkimustyö	**Perusasiat**	**Syventävät työt**
Ensimmäinen ohjelma BLINK		
LEDin vilkuttelu, liikennevalot		
RGB-ledi		
Ohjelman rakenteita, FOR-silmukka		
Painonapin lukeminen		
Jännitteen mittaaminen – analoginen input		
Jännitteenjakaja – potentiometrin lukeman lukeminen		
Äänen tuottaminen		
Lämpötilan mittaaminen		
Valaistuksen mittaaminen		
Servon ohjaaminen Arduinolla		
PWM Pulse Width Modulation: diodin himmentäminen		
PWM: Moottorin ohjaaminen Arduinolla		
Ultraäänianturin liittäminen Arduinoon		
LCD-näytön liittäminen Arduinoon		
Oman funktion tekeminen		
Vastusten värikoodi		

Tutkimuksissa tarvittavat välineet ja komponentit

Arduino Uno
Koekytkentälevy
Hyppylankalajitelma (tai kytkentälankaa, josta voi leikata pätkiä)
Vastus 220 ohm, 0,5 W
Vastus 10 kohm, 0,5 W
Painonappi, sulkeutuva kalvokytkin
Potentiometri 10 kohm (tarvittaessa säätöakseli potikalle)
Pienoispietsokaiutin
LDR
100 uF, 16 V elko
LED pun
LED kelt
LED vihr
RGB LED yht.katodi
Lämpötila-anturi TMP36
MOSFET IRF530
Diodi 1N4005
LCD-näyttö
Pieni sähkömoottori (toimintajännite noin 9 V)
Ultraäänianturimoduli HC-SR04
Pieni servo