ARDUINON KÄYTTÖ SULAUTETTUJEN JÄRJESTELMIEN OPETUKSESSA

Oppimateriaalin kehittäminen

Anni-Elina Seipäjärvi

Opinnäytetyö
Kesäkuu 2015
Kone- ja tuotantotekniikka
Tuotekehitys

TAMPEREEN AMMATTIKORKEAKOULU
Tampere University of Applied Sciences

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Kone- ja tuotantotekniikka
Tuotekehitys

SEIPÄJÄRVI, ANNI-ELINA:
Arduinon käyttö sulautettujen järjestelmien opetuksessa
Oppimateriaalin kehittäminen

Opinnäytetyö 89 sivua, joista liitteitä 44 sivua
Kesäkuu 2015

Sulautetut järjestelmät korvaavat nykyään paljon sellaisia järjestelmiä, joiden virtapiirit toteutettiin aiemmin elektroninen komponentti kerrallaan. Niitä on kattavasti automaation eri sovellutuksissa ja useimmiten niiden toimintaa ohjaa mikrokontrolleri, esimerkiksi Arduino Uno (R3) -mikrokontrollerialusta. Arduino on kokonaisuudessaan avoimeen lähdekoodiin perustuva kehitysalusta, johon sisältyy sekä fyysinen piirilevy, että tietokoneella hallittava kehitysympäristö.

Opinnäytetyön tarkoituksena oli koota selkeä oppimateriaalikokonaisuus sulautettujen järjestelmien opetukseen Tampereen ammattikorkeakoulun koneautomaation opintopolun tarpeisiin. Tavoitteen mukaisesti oppimateriaali kehitettiin konetekniikan opiskelijoille, jolloin se ei sisällöltään vastaa tietotekniikan koulutusohjelman vaatimuksia. Työ rajattiin painottumaan Arduino-mikrokontrollerialustan käyttöön, joten ohjelmointikielestä esiteltiin ainoastaan ne ohjelmoinnin peruskäsitteet, jotka ovat oleellisia oppimateriaalin käytännön harjoituksille. Arduino valittiin käytettäväksi, koska se on yleinen ja monipuolinen kehitysalusta, jonka käyttö on kuitenkin suhteellisen helppoa.

Työn tuloksena muodostui kuvitetusta teoriaosuudesta, ohjelmoinnin peruskäsitteiden yhteenvedosta sekä käytännön harjoituksista koostuva materiaalikokonaisuus. Sen kautta oppija perehtyy sulautettuihin järjestelmiin ja mikro-ohjaintekniikkaan saaden käsityksen siitä, mikä Arduino on ja mitä sen avulla on mahdollista toteuttaa.

Asiasanat: arduino, mikro-ohjaintekniikka, sulautetut järjestelmät.

ABSTRACT

Tampereen ammattikorkeakoulu
Tampere University of Applied Sciences
Mechanical and Production Engineering
Product Development

SEIPÄJÄRVI, ANNI-ELINA:
Applying Arduino to Teaching of Embedded Systems
Production of Educational Material

Bachelor's thesis 89 pages, appendices 44 pages
June 2015

Embedded systems control many devices in common use today, especially in automation applications. In the early days, embedded systems were built primarily on circuit boards by adding one electronic component at a time. Modern embedded systems are much simpler to use as they are often based on microcontrollers, such as the Arduino Uno (R3) microcontroller board. Altogether, Arduino is an open-source electronics platform including both hardware and computer controlled software.

The purpose of this thesis was to research and develop teaching material about embedded systems for students specialising in automation in Tampere University of Applied Sciences. According to the objective, the material is directed for students of Mechanical and Production Engineering so it is not directly applicable in the Degree Programme of Information Technology. The focus of the thesis was delimited to the usage and application of the Arduino microcontroller board, hence the coverage of software and programming language is limited. Arduino was chosen because it is considered a diverse and yet very user-friendly microcontroller platform.

As a result, a teaching material ensemble was created, which consists of an illustrated theory part, a programming language guide and practical exercises. The material will help students familiarise themselves with embedded systems and microcontrollers, and to gain understanding of what Arduino is and how to make the most of it.

Key words: arduino, microcontrollers, embedded systems.

SISÄLLYS

ERITYISSANASTO

ADC	Analog to Digital Converter; analogia-digitaalimuunnin
CPU	Central Processing Unit; mikroprosessori eli suoritin
DAC	Digital to Analog Converter; digitaali-analogiamuunnin
FET	Field Effect Transistor; kanavatransistori
GPIO	General Purpose Input/Output; yleiskäyttöinen portti elektroniikassa
I/O - liittimet	Input/Output - liittimet
IC	Integrated Circuit; mikrokontrolleri
IDE	Integrated Development Environment; Arduinon intergroitu kehitysympäristö
PWM	Pulse Width Modulation; pulssinleveysmodulaatio
RAM	Random Access Memory; työmuisti, joka tyhjenee kun virta katkaistaan
ROM	Read Only Memory; flash-tyyppinen muisti, jossa tieto säilyy sammutettaessa
Sulautettu järjestelmä	Embedded system; laitteisto, jossa mikrotietokone on osana jotakin elektroniikkajärjestelmää

1 JOHDANTO

Tämä opinnäytetyö käsittelee sulautettujen järjestelmien oppimateriaalin kehittämistä Tampereen ammattikorkeakoulun koneautomaation opintopolun tarpeisiin. Oppimateriaalin kohderyhmä on 3. vuosiluokan koneautomaation opiskelijat, joilla on pohjatiedot käytettävistä komponenteista ja kytkentöjen suorittamisesta. Opinnäytetyön tarkoituksena on mahdollistaa opiskelijoiden perehtyminen mikro-ohjaintekniikkaan ja Arduinoon käytännön harjoitusten kautta teoriamateriaalin tukemana. Luotu materiaali on sovellettavissa myös muiden konetekniikan opintopolkujen oppiaineisiin.

Opinnäytetyön tavoitteena on luoda toimiva oppimateriaalikokonaisuus harjoituksineen käyttäen Arduino Uno (versio R3) -mikrokontrollerialustaa. Harjoituksiin voidaan soveltaa myös muita Arduino-mikrokontrollerialustoja (tai vastaavia). Arduino on valittu käytettäväksi, koska se on yleinen ja monipuolinen kehitysalusta, jonka käyttö on kuitenkin suhteellisen helppoa.

Opinnäytetyön teoriaosuudessa perehdytään lyhyesti oppimateriaalin luomiseen pedagogisen näkökulman kautta, jonka jälkeen esitellään sulautetut järjestelmät sekä mikrokontrollerin rakenne ja toimintaperiaate. Teoksen pääpaino on Arduino-kehitysalustaan perehtymisessä, mikä kattaa sekä mikrokontrollerialustaan että ohjelmointiympäristöön tutustumisen. Aihe on rajattu käsittelemään mikro-ohjaintekniikkaa eikä käytettäviä komponentteja käsitellä elektroniikan teorian kannalta.

Ohjelmointikielestä käydään läpi ainoastaan ne ohjelmoinnin peruskäsitteet, jotka ovat oleellisia oppimateriaalin käytännön harjoituksille. Mikro-ohjaintekniikan kannalta opinnäytetyössä käsitellään teoriaa nollatason FET:sta, pulssinleveysmodulaatiosta sekä AD/DA -muuntimesta. Lopuksi esitellään tiivistetysti opinnäytetyön sivuprojektina syntynyt laboratoriovälineiden tuotekehitysprosessi sekä työn pohdinta, lähteet ja liitteet. Oppimateriaaliin kuuluvat ohjelmointikielen peruskäsitteet sekä harjoitukset ovat opinnäytetyön raportin liitteinä.

2 OPPIMATERIAALIN KEHITTÄMINEN

2.1 Oppiminen prosessoimalla

Oppimateriaalia rakentaessa on pidettävä mielessä, ettei oppimisprosessi ole itsestään selvä seuraus opetusprosessista. Jokainen oppija on yksilö jonka oppimiseen vaikuttavat maailmankuva, aikaisempi tietämys aiheesta, yksilön oppimisvalmiudet sekä tiedon prosessoinnin tapa oppimistilanteessa. Perinteisesti opetus on mielletty tietoa siirtäväksi toiminnaksi. Kuitenkin tieto välittyy oppijalle vain tämän oman prosessoinnin kautta, jolloin opittu asia myös ymmärretään ja sisäistetään. Tämä tapa soveltuu sulautettujen järjestelmien oppimiseen erinomaisesti, sillä oppijalla on jo luonnostaan taipumus pohtia ratkaisuja ongelmiin harjoitusten edetessä. Prosessointiin kuuluu olennaisena osana oppijan tiedonkäsittelyprosessit, ajattelu ja ongelmanratkaisu. (Koli & Silander 2002, 10.)

Oppimateriaalilla on hyvä olla selkeä juoni joka helpottaa opittavan aiheen hahmottamista kokonaisuutena. Suomisen ja Nurmelan (2011, 18) mukaan irtonaisia tehtävänantoja on syytä välttää ja harjoitukset kannattaa sijoittaa esimerkiksi ongelmalähtöisen oppimisen, tutkivan oppimisen tai projektioppimisen kategorioihin. Tehtävävetoisuus on käypä vaihtoehto materiaalivetoisuudelle. Oppija aktivoituu pohtien ensin esitettyjä kysymyksiä joille annetaan myöhemmin vastaukset toisin kuin perinteisellä tavalla jossa kerrotaan ensin vastaukset ja kysytään niitä lopuksi. (Suominen & Nurmela 2011, 18.) Näin oppijan keskeinen rooli tekijänä korostuu.

Oppimateriaalia kehitettäessä kantavana ideana on ollut oppiminen tekemällä. Teoriaosuus on pyritty muodostamaan selkeäksi ja käytännönläheiseksi, jotta harjoitusten alkuun pääsemisen kynnys on matala. Tehtävät ovat sidoksissa toisiinsa osittain jopa kiinteästi, mutta niiden suoritusjärjestystä on mahdollista vaihdella oppimateriaalin joustavuuden takaamiseksi.

2.2 Tutkiva oppiminen

Tutkiva oppiminen on pedagoginen ajattelutapa, johon kuuluu olennaisena osana ongelmien ihmettely ja luova kokeileminen. Se perustuu ajatukseen, jonka mukaan

aikaisemmin luodun tiedon ymmärtäminen on psykologisesti samankaltainen prosessi kuin uuden tiedon luominen esimerkiksi tieteessä (Koli & Silander 2002, 24–25).

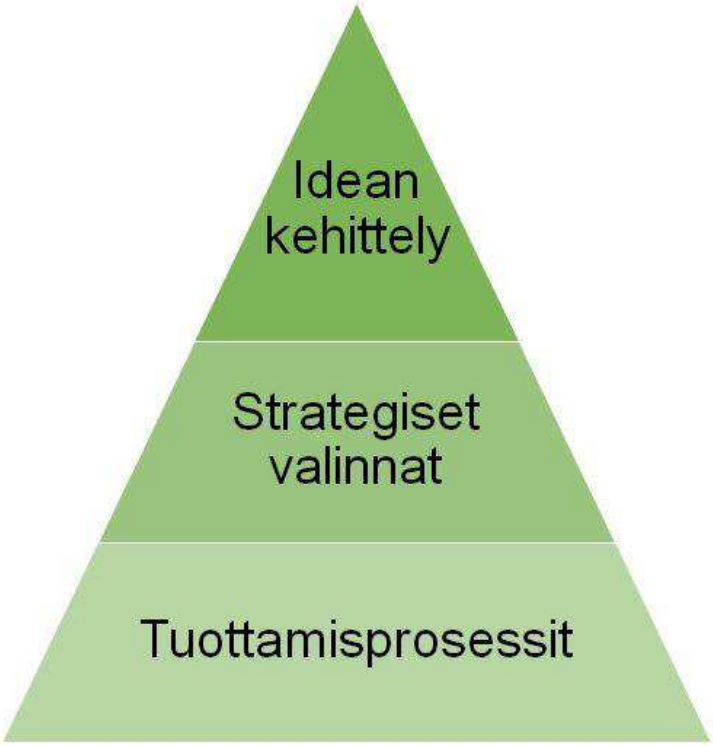
Oppijan yrittäessä ymmärtää opittavaa aihetta hän tulee samalla käyneeksi läpi samanlaisia prosesseja kuin teorian kehittäjä itse. Tätä on pyritty soveltamaan käytäntöön luomalla harjoituksia, jotka antavat opiskelijalle sopivasti haastetta olematta kuitenkaan turhauttavia. Tutkimusprosessin tuloksena oppijalle kehittyy laajentuneen ymmärryksen lisäksi aina uusia kysymyksiä, jolloin ilmiöitä pyritään selittämään yhä syvällisemmin (Koli & Silander 2002, 24–25). Uudet opitut asiat syntyvät näin ollen vanhaan tietoon perustuen itse asetettujen kysymysten, ajatusten ja teorioiden kautta. Sulautettujen järjestelmien tutkiva oppiminen on päättymätön prosessi, jonka synnyttämien uusien kysymyksien käsittelyä on mahdollista jatkaa seuraavilla opintojaksoilla tai muissa oppiaineissa. Parhaimmillaan tämä prosessi tuottaa uutta ymmärrystä ja tietoa (Koli & Silander 2002, 25).

2.3 Lukijalähtöinen kirjoitus

Yhdeksi kehitettävälle oppimateriaalille asetetuista tavoitteista nousi materiaalin käyttökelpoisuus mahdollisimman pitkään tulevaisuudessa. Opiskelumuodoista verkko-opiskelu kasvanee yhä suosituimmaksi sen joustavuuden ansiosta. Sitä vastoin käytännön harjoituksia on lähes mahdoton suorittaa etäopiskeluna jo pelkästään kytkentöjen ja tarvittavien komponenttien vuoksi. Lähiopetustuntien määrän pienentyessä ajaudutaan väistämättä tilanteeseen, jossa osa teoriaosuudesta suoritetaan etäopiskeluna hyödyntäen verkkotyöskentelyä. Näin ollen luotu oppimateriaali on pyritty muodostamaan mahdollisimman yksiselitteiseksi, jolloin sen käsittely itsenäisesti on mielekkäämpää oppijan kannalta.

Verkko-opiskelun haaste on tekstin määrä erityisesti teoriapainotteisissa dokumenteissa. Suominen ja Nurmela (2011, 78) painottavat lukijalähtöisen ja kiinnostavan tekstin merkitystä joka ottaa lukijan tarpeet huomioon. Tekstin ollessa hyödyllistä ja monipuolista näytöltä lukemisen hankaluus unohtuu ja oppija voi keskittyä tiedon prosessointiin.

Suomisen ja Nurmelan (2011) mukaan oppimateriaalin tuottamista voi kuvata oheisen diagrammin mukaisella kolmiolla (kuvio 1). Kolmion huipulla on idean kehittely, jolla haetaan soveltuvia innovatiivisia ratkaisuja. Keskellä sijaitsevat strategiset valinnat, jotka vastaavat esimerkiksi kysymyksiin millaista oppimista tavoitellaan, mikä on relevanttia tietoa ja miten oppijan tieto rakentuu. Strategisia valintoja pohdittaessa on aina huomioitava myös kohderyhmä. Alimmaisena kolmiossa on tiedon tuottamisprosessi. (Suominen & Nurmela 2011, 78.)



KUVIO 1. Oppimateriaalin tuottaminen (Suominen & Nurmela 2011, muokattu)

Tiivistetysti lukijalähtöinen materiaali täyttää vähintään seuraavat vaatimukset:

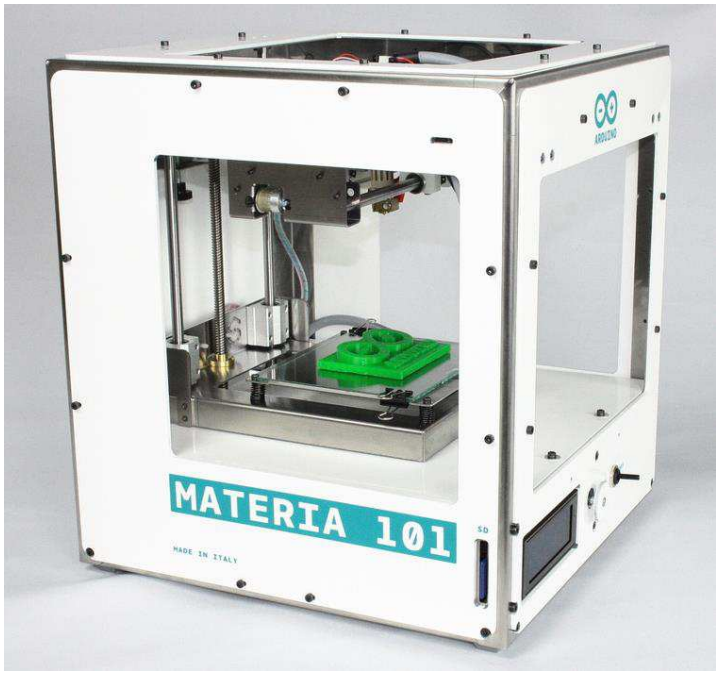
- Materiaalin sisältö huomioi kohderyhmänsä.
- Asia voidaan yhdistää käytäntöön todellisuudessa.
- Kieliasu on selkeä ja ymmärrettävä.

3 SULAUTETUT JÄRJESTELMÄT

3.1 Sulautettu järjestelmä

Sulautetuksi järjestelmäksi (engl. embedded system) kutsutaan sellaista järjestelmää, joka itsessään sisältää mikrotietokoneen ja näin ollen pystyy suorittamaan itsenäisiä päätöksiä ja toimintoja ulkoisten sensorien antamien signaalien perusteella. Nykyään sulautettuja järjestelmiä on joka puolella ympärillämme, emme vain välttämättä ajattele niiden olemassaoloa.

Arjen esimerkkejä sulautetuista järjestelmistä ovat pyykki- ja astianpesukone, jotka ajavat tietyn ohjelman tietyllä käyttäjän määrittämällä asetuksella. Käyttäjän ei tarvitse odottaa koneen vieressä ja vaihtaa toimintaa esimerkiksi huuhteluun, sen sijaan kone ymmärtää itse käynnistää huuhtelun ajastimen tai anturin tietoon perustuen. Yksinkertainen esimerkki sulautetusta järjestelmästä on mikroaaltouuni, robotti-imuri tai auton lukkiutumattomat jarrut, monimutkainen sovellus puolestaan on esimerkiksi useita prosesseja yhtäaikaisesti mahdollistava älypuhelin. Kuvassa 1 on esimerkkinä Arduino Mega 2560 -mikrokontrollerialustan avulla toteutettu 3D-tulostin, jonka voi halutessaan rakentaa itse osista.



KUVA 1. Arduino Materia 101 3D-tulostin (Arduinon kotisivut 2015)

Sulautetuissa järjestelmissä ei yleensä ole erillistä massamuistia ja on mahdollista, että järjestelmän laite on ohjelmoitavissa vain kerran. Ne havainnoivat jatkuvasti ympäristönsä tilaa ja reagoivat ympäristön määräämällä tahdilla sensorien antamien signaalien mukaisesti. Sulautetuille järjestelmille on tyypillistä käsitellä tarvittava tieto välittömästi. (Karvinen & Karvinen 2011, 7.)

Sulautettu järjestelmä on suunniteltu tiettyä toimintaa varten, eikä se ole helposti muokattavissa muuhun tarkoitukseen. Tämä erottaa sulautetun järjestelmän tietokoneesta, kuten myös tyypillisestä sulautetusta järjestelmästä puuttuva hiirestä, näytöstä ja näppäimistöstä muodostuva käyttöliittymä. Sen sijaan sulautettua järjestelmää ohjataan esimerkiksi valaistuksen määrällä tai polkimilla. (Karvinen & Karvinen 2011, 7.) Yleisesti ottaen sulautetut järjestelmät ovat usein fyysiseltä kooltaan suhteellisen pieniä ja niiden ohjaus on toteutettu mikrokontrollerin avulla.

3.2 Mikrokontrolleri

Mikrokontrolleri on olennainen osa Arduinon kehitysalustaa, sen perustajajäsenen Massimo Banzin (2011, 17) mukaan sitä voidaan pitää koko piirilevyn sydämenä. Sitä voidaan kutsua myös mikro-ohjaimeksi, mutta mikrokontrolleri on yleisempi nimitys. Siitä voidaan käyttää joissain yhteyksissä myös englanninkielistä nimeä Integrated Circuit tai sen lyhennettä IC. Mikrokontrolleri on mikropiiri, johon on integroitu mikroprosessorin lisäksi muisti- ja liityntälohkoja ja mahdollisia oheiskomponentteja. Ne ovat yleensä helposti ohjelmoitavissa ja niiden avulla laajankin ongelman voi ratkaista pala kerrallaan. (Karvinen & Karvinen 2011, 8, 17.) Kuvassa 2 on Arduino Uno:n sisältämä ATmega328-mikrokontrolleri, jonka on valmistanut Atmel Corporation.

KUVA 2. ATmega328-mikrokontrolleri (Protostack 2015)

Mikroprosessori eli suoritin (CPU) on mikrokontrollerin komentokeskus. Se on yhteydessä kaikkiin muihin mikrokontrollerin osiin ja siten kokoaa sen yhtenäiseksi systeemiksi. Mikroprosessorin pääasiallinen tehtävä on vastaanottaa tietoa, kääntää se, ja lopulta lähettää eteenpäin todetut toimenpiteet. Mikroprosessorin tehokkuudella on suuri vaikutus mikrokontrollerin toimintanopeuteen. (CircuitsToday: Basics of Microcontrollers 2011.)

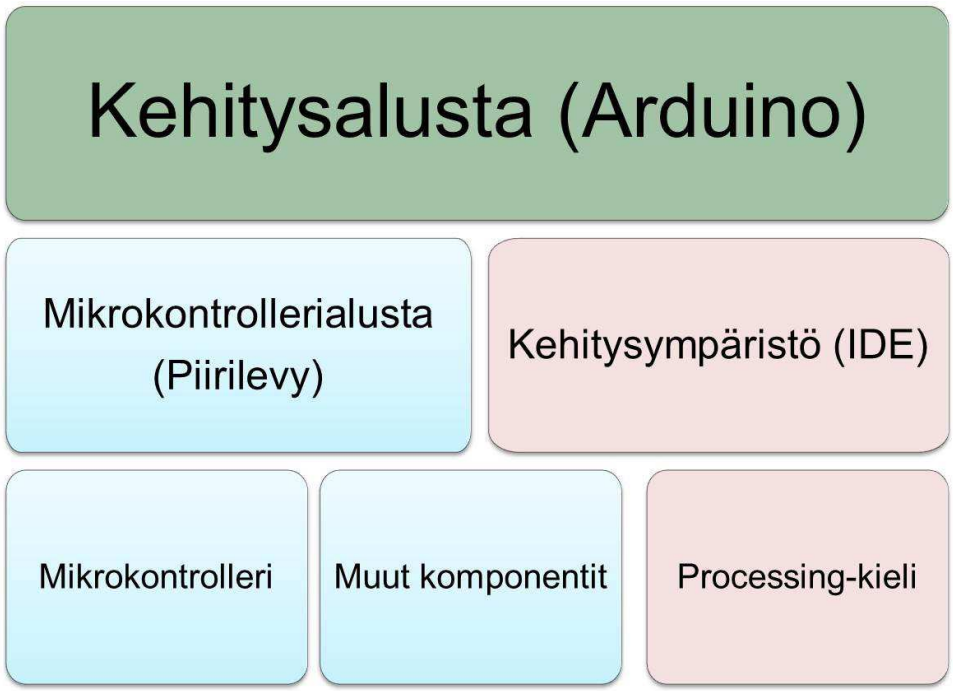
Mikrokontrolleri sisältää sekä luku- että työmuistia versiosta riippuvan määrän. Muisti on tarkoitettu ohjelman ja tiedon säilyttämiseen mikroprosessoria varten. Lukumuisti (ROM, engl. Read Only Memory) on flash-tyyppinen muisti, jossa tieto säilyy sammutettaessa. Siihen tallennetaan esimerkiksi tietokoneella luotu ohjelma, jonka mukaan mikrokontrollerialusta toimii. Sitä vastoin työmuisti (RAM, engl. Random Access Memory) tyhjenee joka kerta kun mikrokontrollerialustasta katkaistaan virta. (CircuitsToday: Basics of Microcontrollers 2011.)

Mikrokontrollerin liityntälohkoja ovat sarjaportit ja I/O - liittimet, joita käytetään tiedonsiirtoon mikrokontrollerin ulkopuolisten käyttölaitteiden (kuten LED-valo) ja oheislaitteiden (kuten tietokone) välillä. Liityntälohkot voidaan ohjelmoida suorittamaan erilaisia toimintoja. Mikrokontrolleriin mahdollisesti integroituja oheiskomponentteja ovat esimerkiksi erilaiset ajastimet ja laskurit, keskeytysten ohjaus ja AD/DA -muuntimet eli analogia-digitaali- ja digitaali-analogiamuuntimet. (CircuitsToday: Basics of Microcontrollers 2011.)

4 KEHITYSALUSTA

4.1 Kehitysalustat yleisesti

Sulautettuihin järjestelmiin tutustuminen on helpointa aloittaa valmiin kehitysalustan avulla. Kehitysalustaksi kutsutaan kokonaisuutta, johon kuuluu sekä fyysinen piirilevy, että tietokoneella hallittava kehitysympäristö eli IDE (engl. Integrated Development Environment). Kuviossa 2 on havainnollistettu kehitysalustan rakennetta jäsennellysti.



KUVIO 2. Kehitysalustan rakenne

Kehitysalustoja on runsaasti erilaisia eri käyttötarkoituksia varten. Lähes jokaisella mikrokontrollerivalmistajalla on omat kehitysalustansa, mutta ne ovat yleensä suhteellisen kalliita. Edullisia ja mielenkiintoisia vaihtoehtoja aloittelijalle Arduinon lisäksi tarjoavat muun muassa STMicroelectronics (STM32 Nucleo board), PICAXE (PICAXE-08M2), Texas Instruments (TI LaunchPad) ja lukuisat Arduino-kopiot, jotka ovat usein tunnistettavissa duino-loppuisesta nimestään (Funduino, Boarduino, Freeduino, Mintduino, Netduino, AEduino, Pinguino jne.). Kehitysalustan voi halutessaan tehdä myös itse komponenteista juottamalla. Arduino-alustojen kytkentäkaaviot ovat vapaasti ladattavissa Arduinon kotisivuilla (2015). Oma lukunsa on yhä suositumpi Rasberry Pi, joka on yhden piirilevyn tietokone ja periaatteellisesti

eroaa siten sulautetuista järjestelmistä. (Arduino kotisivut 2015; PICAXE kotisivut 2015; Rasberry Pi kotisivut 2015; STMicroelectronics kotisivut 2015; Texas Instruments kotisivut 2015.)

Kehitysalustan valintaan vaikuttavat mikrokontrollerin nopeus ja monipuolisuus, muistin määrä, suorittimen tehokkuus ja virrankulutus. Esimerkiksi useita prosesseja yhtäaikaisesti mahdollistava sovellus tarvitsee tehokkaan suorittimen. Vastaavasti jossain sovelluksessa vaatimuksena voi olla pieni virrankulutus, joka mahdollistaa laitteen käytön pitkään pelkällä paristolla. Useimmiten parhaat ratkaisut ovatkin jollain tavoin kompromisseja.

Kehitysalustoissa on eroavaisuuksia ohjelmoitavien liitinnastojen määrässä sekä siinä, millä tavoin liitinnastat ovat ohjelmoitavissa (input/output, analog/digital). Valinnassa kannattaa ottaa huomioon myös ohjelmoinnin yksinkertaisuus, käytettävä ohjelmointikieli ja kehitysalustan yleinen suosio. Arduinot ovat yleensä ohjelmoitavissa USB-kaapelilla, mutta joihinkin alustoihin tarvitaan erillinen ohjelmointilaite. Processing-kieleen perustuvaa ohjelmointia käyttävät Arduino ja Texas Instruments, C-kieleen perustuvaa STMicroelectronics ja BASIC-kieleen perustuvaa PICAXE. Rasberry Pi:tä ohjelmoidaan pääasiassa Python-ohjelmointikielellä. (Arduino kotisivut 2015; PICAXE kotisivut 2015; Rasberry Pi kotisivut 2015; STMicroelectronics kotisivut 2015; Texas Instruments kotisivut 2015.)

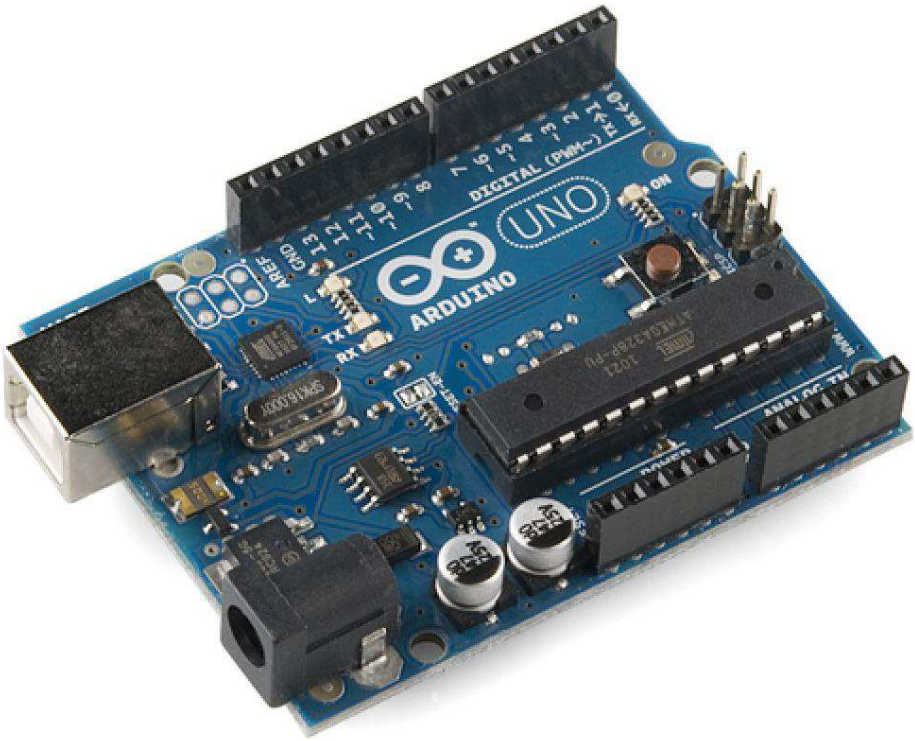
Mitä suositumpi kehitysalusta, sitä helpommin sen käyttöön löytyy ohjeita ja valmiita ohjelmia internetistä. Arduinolla on aktiivinen harrastajayhteisö, joka ylläpitää ohjeita sekä julkaisee uusia sovellutuksia Arduinon kotisivujen (2015) Playground-osiossa. Opinnäytetyön teon hetkellä verkko-osoite sinne on http://playground.arduino.cc/ .

4.2 Arduino Uno R3

Arduino Uno:a (kuva 3) pidetään usein referenssituotteena muille kehitysalustoille. Se on korvannut suuren suosion saavuttaneen Arduino Duemilanove:n ja on usein ensimmäinen hankittu mikrokontrollerialusta edullisuutensa (25 €, robomaa.fi 2015) ja yksinkertaisuutensa ansiosta. Sen liitinnastoissa eli pinneissä on reiät, joihin voi kytkeä johtoja ja hyppylankoja ilman juottamista, mikä nopeuttaa koekytkentöjen tekemistä

huomattavasti. Pitkäaikaisia kytkentöjä suunniteltaessa johdot kannattaa kiinnittää kuitenkin paremmin, sillä ne irtoavat kohtalaisen helposti.

Tässä opinnäytetyössä keskitytään Arduino Uno:n versioon R3 (Revision 3), joka perustuu Atmel Corporation:n ATmega328-mikrokontrolleriin. Arduino Uno:n käyttöjännite on 5 V ja siinä on 6 analogista input-liitinnastaa, 14 digitaalista input/output -liitinnastaa, joista kuutta voi käyttää analogisena output-liitinnastana, 16 MHz resonaattori, ICSP (engl. In-Circuit Serial Programming) -kollektori, USB-portti, reset-painike ja DC-virtaliitin. Tarkemmat tekniset tiedot Arduino Uno (R3):sta ovat listattuna taulukossa 1. (Arduino kotisivut 2015.)

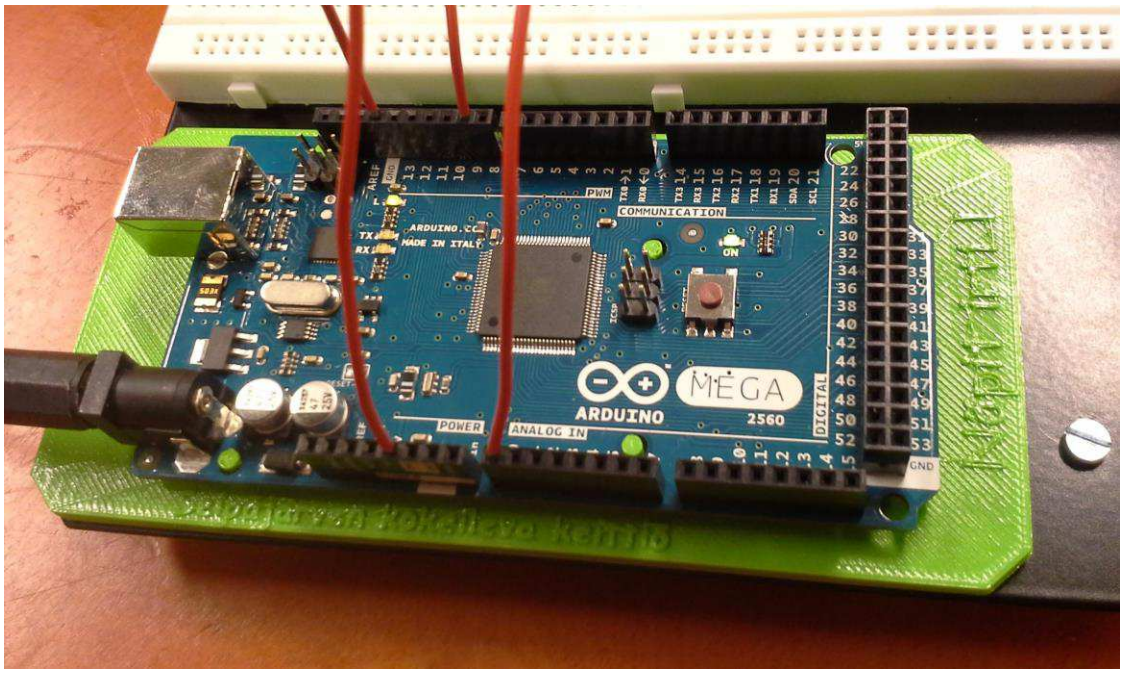


KUVA 3. Arduino Uno -mikrokontrollerialusta (Arduinon kotisivut 2015)

4.3 Arduino Mega 2560 R3

Arduino Mega 2560 -mikrokontrollerialusta (kuva 4) on korvannut aikaisemman Arduino Mega:n ja siinä on enemmän liitinnastoja ja muistia kuin Arduino Uno:ssa. Hinnaltaan se on kalliimpi kuin Uno (50 €, robomaa.fi 2015). Myös sen liitinnastoissa on reiät hyppylankojen nopeaa kiinnittämistä varten. Arduino Mega (R3) perustuu ATmega2560-mikrokontrolleriin ja siinä on Arduino Uno:sta eroten 16 analogista

input-liitinnastaa, 54 digitaalista input/output -liitinnastaa, joista 15 voi käyttää analogisena output-liitinnastana ja neljä UART (engl. Universal Asynchronous Receiver/Transmitter) -sarjaliikennepiiriä. Tarkemmat tekniset tiedot Arduino Mega 2560 (R3):sta ovat listattuna taulukossa 1. (Arduino kotisivut 2015.)



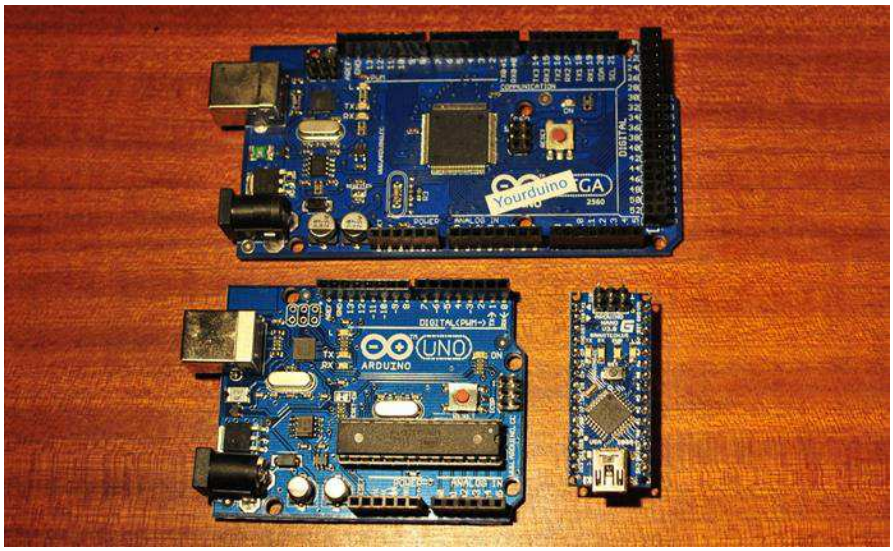
KUVA 4. Arduino Mega 2560 -mikrokontrollerialusta

4.4 Arduino Nano 3.0

Arduino Nano on fyysisiltä mitoiltaan huomattavasti pienempi ja hinnaltaan kalliimpi (45 €, robomaa.fi 2015) kuin edellä esitelty Uno. Kuvassa 5 on esitetty eri mikrokontrollerialustoiden keskinäiset mittasuhteet. Nano on herkempi rikkoutumaan kuin Uno, esimerkiksi pienemmät oikosulkuvirrat tuhoavat sen prosessorin pysyvästi. Nanoa käytetään sovelluksissa, jotka edellyttävät pientä kokoa. Toisaalta pieni koko luo omat ongelmansa, muun muassa pinnien merkintöjä on vaikeampi lukea. Nanon pinneissä onkin reikien sijaan piikit, joten sen voi painaa kiinni kytkentäalustaan mikä helpottaa kytkentöjen suorittamista.

Arduino Nano 3.0 perustuu ATmega328-mikrokontrolleriin, samoin kuin Arduino Uno (R3). Nano 3.0:ssa on Uno:sta eroten 8 analogista input-liitinnastaa, 14 digitaalista input/output -liitinnastaa, joista kuutta voi käyttää analogisena output-liitinnastana ja siitä puuttuu DC-virtaliitin kokonaan. Virta voidaan tuoda Nanolle mini-B USB -

liittimellä tai syöttämällä se tiettyihin pinneihin. Tarkemmat tekniset tiedot Arduino Nano 3.0:sta ovat listattuna taulukossa 1. (Arduino kotisivut 2015.)



KUVA 5. Arduino Mega 2560 (ylhäällä), Uno ja Nano (bajdi.com 2015)

4.5 Arduino-mikrokontrollerialustojen vertailu

Taulukossa 1 on havainnollistettu kolmen yleisen mikrokontrollerialustan välisiä teknisiä eroavaisuuksia.

TAULUKKO 1. Vertailu erilaisten Arduino-mikrokontrollerialustojen teknisistä tiedoista

Arduino	Uno (R3)	Mega 2560 (R3)	Nano 3.0
Prosessori	ATmega328	ATmega2560	ATmega328
Kellotaajuus	16 MHz	16 MHz	16 MHz
Analogiset sisääntulot	6	16	8
Digitaaliset IO / PMW	14 / 6	54 / 15	14 / 6
EEPROM (kb)	1	4	1
SRAM (kb)	2	8	2
Flash (kb)	32	256	32
USB-liitäntä	Tavallinen B	Tavallinen B	mini-B
UART (kpl)	1	4	1

5 I/O – LIITÄNNÄT

5.1 Arduino Uno – piirilevy

Arduinon piirilevy on hieman pankkikorttia suurempi mikrokontrollerikortti, jota kokonaisuudessaan voidaan kutsua mikrokontrollerialustaksi. Arduino Uno:n käyttämä ATmega328-mikrokontrolleri on kortin alaosassa sijaitseva musta mikropiiri (kuva 7). Mikrokontrolleri tarvitsee toimiakseen muita komponentteja, jotka on sijoitettu kortille kuvan 7 osoittaman esimerkin tapaan. Mikrokontrollerikortista on olemassa runsaasti erilaisia versioita joissa on eri määrä komponentteja käyttötarkoituksen mukaan. (Banzi 2011, 18.) Kuvassa 8 on esitetty vertailun vuoksi tehokkaampi ja enemmän liitäntöjä sisältävä Arduino Mega 2560 - mikrokontrollerialusta.

Piirilevyyn voidaan ohjata virta tietokoneesta USB-liittimen (type B) kautta tai kytkeä ulkoinen tasavirtamuuntaja alustan 2,1 mm:n virtaliittimeen (Banzi 2011, 18). Useampi virtalähde voi olla kytkettynä Arduinoon yhtä aikaa. Piiri valitsee automaattisesti saatavilla olevista lähteistä korkeimman jännitteen ja ohjaa sen jänniteregulaattorille. (Wheat 2011, 6.) Akku kytketään Arduino Uno:n Vin- ja GND-liittimiin. Jännitteen syöttäminen suoraan 5 V- tai 3,3 V -lähtöpinneihin ohittaa mikrokontrollerialustan jänniteregulaattorin ja voi vaurioittaa piirilevyä. (Arduinon kotisivut 2015.) Arduino Uno:n suositeltava syöttöjännite on 9 V, mutta sen rajat kannattaa tarkistaa teknisistä tiedoista (engl. datasheet), kuten myös muidenkin mikrokontrollerialustaversioiden.

5.2 Lähdön puskurointi

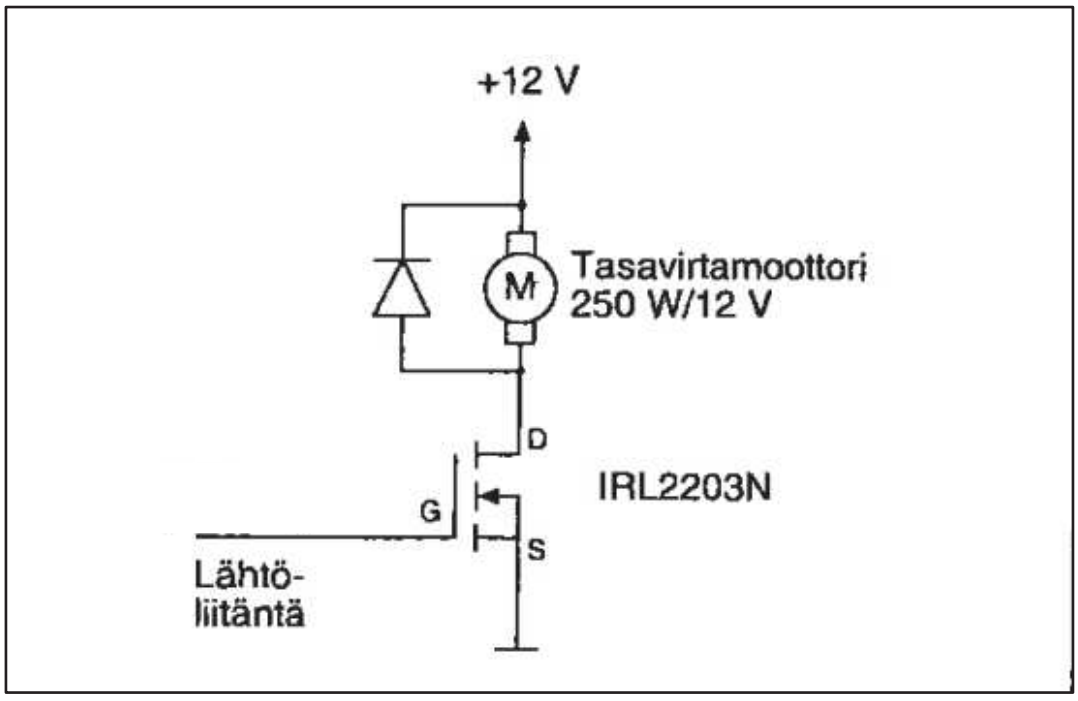
Arduino Uno:n liitinnastojen virranrajoitus on 40 mA, mikä riittää LED:n kaltaisille kohteille. Suuremmille kuormille kuten releille, tuulettimille, moottoreille tai solenoideille tarvitaan ulkoinen ohjain, joka kytkee raskaammat laitteet päälle ja pois. (Wheat 2011, 5.) Komponentiksi käy esimerkiksi yleisesti käytetty MOSFET-kanavatransistori, joka on elektroninen puolijohde ja kytkin, tai vaihtoehtoisesti tavallinen transistori. Virranrajoituksen maksimiarvo on suositeltavaa tarkistaa käytettävälle mikrokontrollerialustalle ennen kytkentöjen suunnittelua, sillä se on

tapauskohtainen. Opinnäytetyön teon hetkellä verkko-osoite Arduino Uno:n tietoihin on http://arduino.cc/en/main/arduinoBoardUno .

Arduinon liitinnastojen virranrajoituksen vuoksi liitäntöjä ei aina voi kytkeä suoraan oheislaitteisiin ilman sopivaa liitäntäelektroniikkaa. Lähdön puskurointi takaa, että lähdöistä saadaan tai ne pystyvät ottamaan vastaan tarpeeksi virtaa. Se myös tasoittaa lähdön ja kuorman jännitetasojen suuruuseroa sekä suojaa laitteen muuta herkempää elektroniikkaa.

5.3 Nollatason FET

Yksittäinen liitäntä voidaan puskuroida transistorin lisäksi erillisellä eristehila FET:llä eli kanavatransistorilla kuvan 6 mukaisesti. Kuvassa 6 käytetään N-kanavaista eristehila FET:ä IRL2203N lähdön, esimerkiksi PWM-nastan (pulssinleveysmodulaatio), puskurointiin ja kuorma on syöttöjännitteen puolella. Kanavatransistorin kuormana on 250 W, 12 V tasajännitemoottori. Moottori käyttää toimiessaan yli 20 A virran ja aiheuttaa käynnistettäessä paljon sitä suuremman virtapiikin, joita liitäntä ei ilman FET:ä kestäisi. (Koskinen 2004, 207.)



KUVA 6. Lähtö puskuroituna eristehila FET:llä (Koskinen 2004)

Eristehila FET aiheuttaa oleellisesti pienemmän tehohäviön kuin transistorilla toteutettu puskurointi sen pienen sisäisen resistanssin $R_{DS(on)}$ ansiosta. Resistanssi R vaikuttaa lähteen (engl. source, S) ja nielun (drain, D) välillä kuvan 6 mukaisesti.

Mitä pienempi sisäinen resistanssi on, sitä vähemmän kanavatransistorissa syntyy tehohäviötä. Tehohäviö aiheuttaa lämpenemistä, mikä korreloi tarvittavan jäähdytyslevyn kokoon. Toisaalta, FET:t ovat myös suhteellisen nopeita komponentteja kytkimenä, joten ne aiheuttavat suuremman häiriösäteilyn ympäristöönsä kuin hitaammat transistorilla toteutetut kytkennät. (Koskinen 2004, 207.) Kanavatransistorien nopeutta hyödynnetään esimerkiksi airsoft-aseissa, joissa tavallinen transistori ei reagoisi tarpeeksi nopeasti liipaisimen painallukseen.

Arduino-mikrokontrollerialustan lähtöliitäntään liitetyn kanavatransistorin on oltava nollatason tyyppiä (engl. Logic-Level FET). Se johtaa täysin jo 3 V ohjausjännitteellä, kun taas tavallinen eristehila FET tarvitsee johtaakseen täysin 6-7 V jännitteen. (Koskinen 2004, 207.) Arduinon mikrokontrollerialustojen käyttöjännite on yleensä 3,3 V tai 5 V.

5.4 Liittimet

Arduino-piirilevyt sisältävät versiosta riippuen vaihtelevan määrän laajasti ohjelmoitavissa olevia GPIO (engl. General Purpose I/O) -pinnejä eli liitinnastoja. Yleisesti ottaen pinnit voidaan ryhmitellä kolmeen eri kategoriaan. Arduino Uno (R3) sisältää

- 14 digitaalista liitinnastaa (pinnit 0-13)

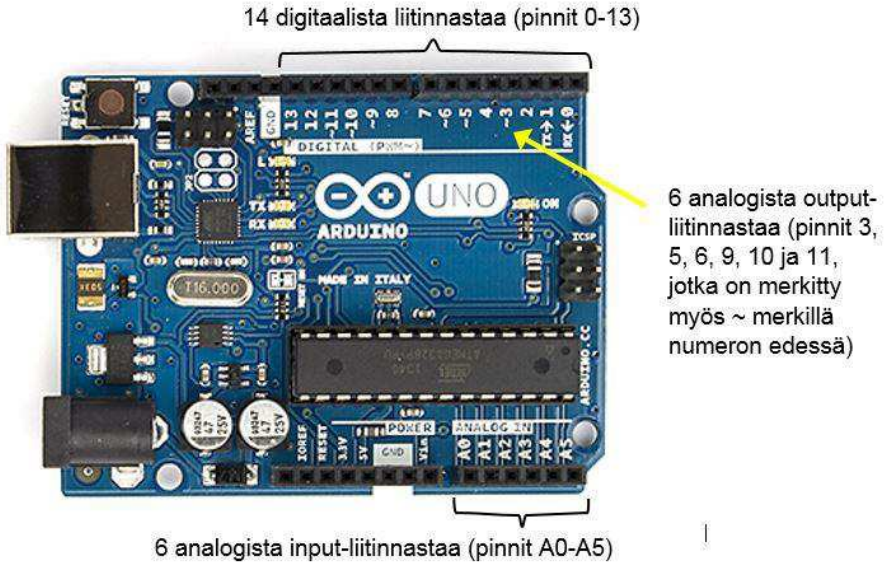
Nämä liitinnastat määritellään IDE:n luonnoksessa toimimaan joko input- tai output-liittiminä luodun ohjelman tarpeen mukaan.

- 6 analogista input-liitinnastaa (pinnit A0-A5)

Nämä pelkästään analogiset input-nastat vastaanottavat analogisia signaaleja (esimerkiksi jännitelukemia sensorista) ja kääntävät ne lukuarvoiksi välillä 0-1023.

- 6 analogista output-liitinnastaa (pinnit 3, 5, 6, 9, 10 ja 11, jotka on merkitty myös ~ merkillä numeron edessä)

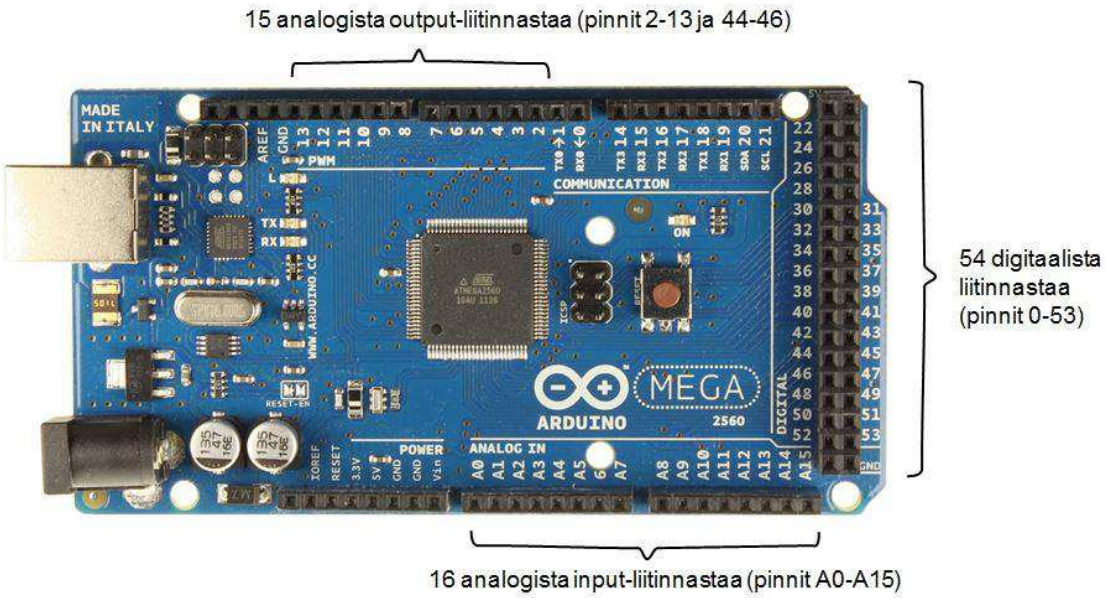
Näitä liitinnastoja voidaan kutsua myös nimellä PWM (pulssinleveysmodulaatio) -nastat. Ne voidaan ohjelmoida IDE:n luonnoksessa myös analogista ulostuloa varten. Ulostulo voi olla lukuarvo väliltä 0-255, mikä vastaa 0-5 V:n output-jännitettä.



KUVA 7. Arduino Uno R3 - piirilevy liitinnastoineen (Arduinon kotisivut 2015, muokattu)

Arduino Mega 2560 (R3) sisältää

- 54 digitaalista liitinnastaa (pinnit 0-53)
- 16 analogista input-liitinnastaa (pinnit A0-A15)
- 15 analogista output-liitinnastaa (pinnit 2-13 ja 44-46)



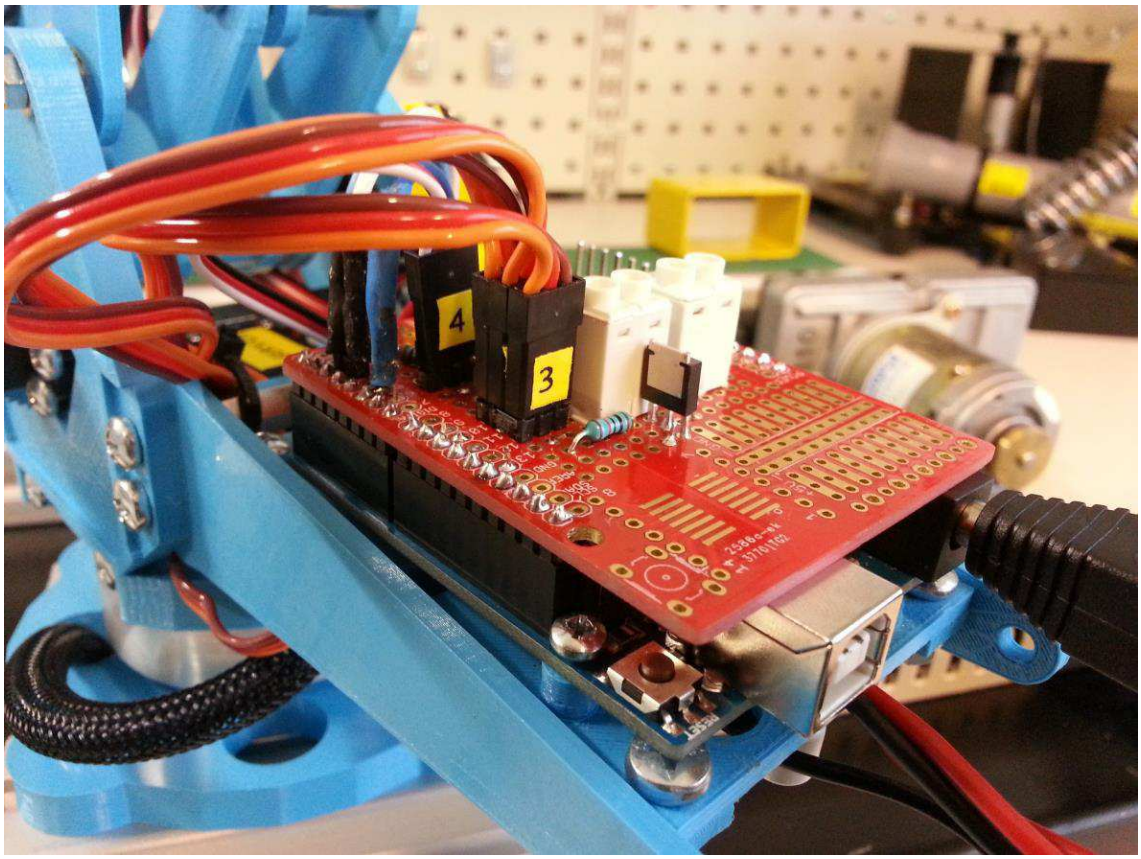
KUVA 8. Arduino Mega 2560 R3 - piirilevy liitinnastoineen (Arduinon kotisivut 2015, muokattu)

5.5 Shield-lisäkortti

Arduinon mikrokontrollerialustan päälle on mahdollista kiinnittää erillinen Shield-lisäkortti laajentamaan alkuperäisen piirilevyn käyttömahdollisuuksia. Shield on erityiseen tarkoitukseen valmistettu piirikortti, joka kiinnitetään mikrokontrollerialustan liitinnastoihin. Halutessaan lisäkortin voi rakentaa myös itse Arduinon kotisivuilta saatavien tietojen avulla. (Arduinon kotisivut 2015.)

Shield varaa käyttöönsä tapauksesta riippuen tietyt liitinnastat toimiakseen, mikä kannattaa ottaa huomioon töitä suunniteltaessa. Shield:n kiinnitetään tarvittavat komponentit kuvan 9 mukaisesti. Shield-lisäkortteja on saatavilla monipuolisesti eri

tarkoituksiin, esimerkiksi langattoman tiedonsiirron (GSM, Ethernet, WiFi, RFID, GPS), näyttöjen (TFT, LCD) tai moottorin ohjauksen mahdollistamiseksi.



KUVA 9. Arduinoon yhteensopiva Shield-lisäkortti kiinnitettynä Arduino Uno:n (Lehtonen, Lätti, Similä & Tammisto 2014)

6 ARDUINO IDE

6.1 Arduino IDE (kehitysympäristö)

Puhuttaessa Arduinosta tarkoitetaan usein työskentelyyn käytettävää fyysistä mikrokontrollerialustaa. Arduinon kehitysalusta kuitenkin muodostuu kahdesta yhtä tärkeästä osasta, joista ensimmäinen on fyysinen piirilevy ja toinen tietokoneella hallittava Arduino IDE eli integroitu kehitysympäristö (Banzi 2011, 17). IDE:llä luodaan ohjelma eli luonnos, jossa määritellään mitä mikrokontrollerialustan tulee tehdä. Tämä luonnos ladataan alustan mikrokontrolleriin USB-liitännän kautta.

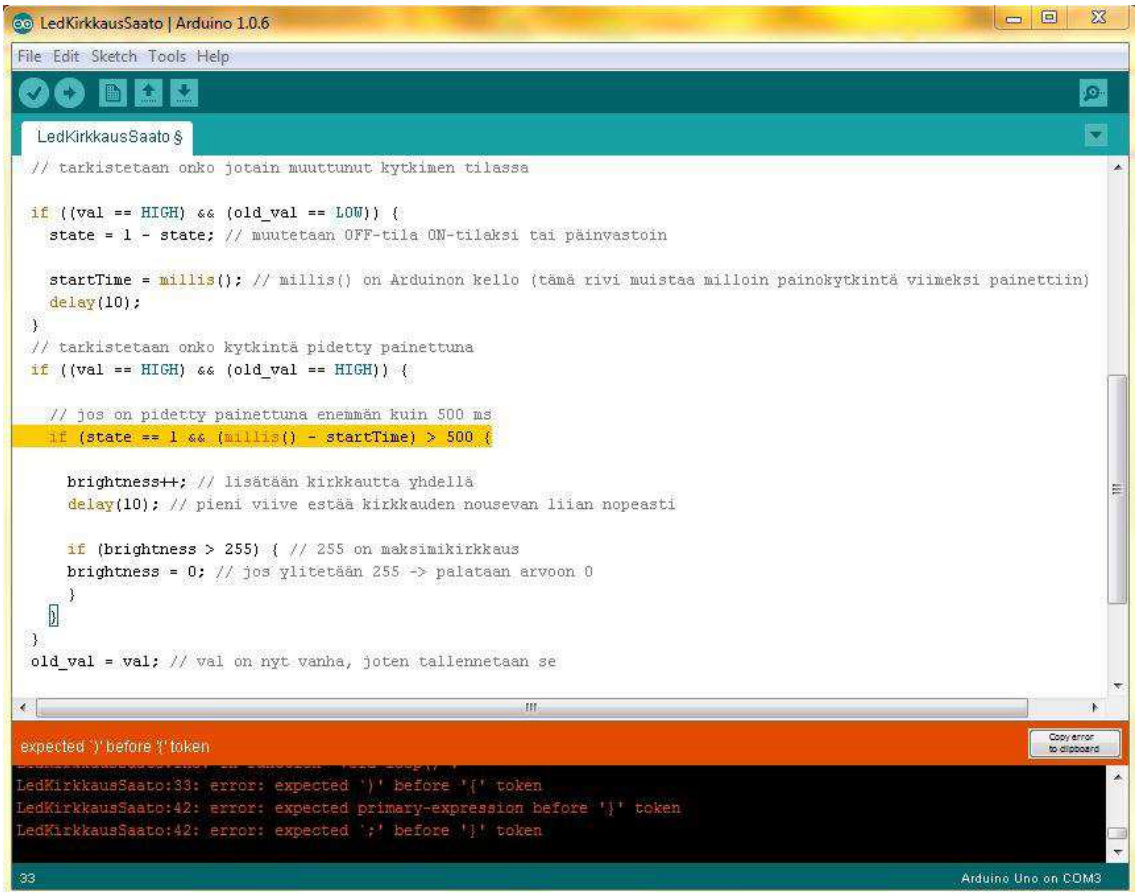
Ohjelma eli luonnos tallentuu mikrokontrollerin lukumuistiin (ROM), missä tieto säilyy virran sammuessa. Mikrokontrollerialustan ei siis tarvitse olla jatkuvasti yhdistettynä tietokoneeseen. Ohjelma voidaan esimerkiksi ladata Arduinon piirilevylle ennakkoon, jonka jälkeen piirilevy asennetaan liikkuvaan robottiin ja virransyöttö järjestetään paristoa käyttäen. Arduinon mikrokontrolleriin on mahdollista tallentaa vain yksi ohjelma kerrallaan, joten uusi luonnos korvaa vanhan luonnoksen ladattaessa.

Arduino IDE perustuu avoimeen lähdekoodiin, näin ollen se on vapaasti internetistä ladattavissa oleva ohjelma. Se on yhteensopiva Windows-, Mac- ja Linux-ympäristöjen kanssa. Kehitysympäristön ohjelmointikieli perustuu Processing-kieleen, joka latauksen alkaessa tulkitaan C-kielelle ja muunnetaan avr-gcc-kääntäjän kautta mikrokontrollerin ymmärtämälle kielelle. (Banzi 2011, 20.) Arduinon käyttämä ohjelmointikieli muistuttaa huomattavasti yksinkertaistettua C-kieltä.

6.2 Processing-kieli

Arduinon kantava ajatus on helpottaa tutustumista sulautettuihin järjestelmiin ja tuoda avoin elektroniikan kehitysalusta mahdollisimman monen käyttäjän ulottuville. Tästä syystä Arduino IDE:n käyttämä ohjelmointikieli vaikuttaa melko yksinkertaiselta perinteisiin ohjelmointikieliin verrattuna. Ohjelmointitaidosta on toki hyötyä kehitysympäristöä opeteltaessa, mutta se ei ole edellytys harjoitusten onnistumiselle. Ohjelma tarkastaa luodun kieliasun oikeellisuuden automaattisesti ennen luonnoksen

latautumista alustan mikrokontrolleriin. Mikäli luonnos on puutteellinen, ohjelma osoittaa epäselvän kohdan kuvan 10 mukaisesti korostaen. Oppimateriaalia kehitettäessä on keskitytty tärkeimpien ohjelmoinnin peruskäsitteiden, kuten funktioiden, ehtolauseiden, silmukoiden ja vertailujen hallintaan (Banzi 2011, 95–109).



KUVA 10. Arduino IDE osoittaa tarkastuksessa ilmenneen epäselvän kohdan luonnoksessa

6.3 Ohjelmoinnin peruskäsitteet

Processing-kielen käyttämät vakiokomennot voidaan jakaa kolmeen kategoriaan, rakenne (engl. structure), arvot (values) ja funktiot (functions). Rakenne esittelee luonnoksen jäsentelyyn tarkoitetut toiminnot, jotka ovat välttämättömiä ohjelman toiminnan kannalta. Arvot käsittää luonnoksessa esiintyvät vakiot ja muuttujat, joita määritellään ja hallitaan halutun toiminnon suorittamiseksi. Funktiot ohjaavat määriteltyjä arvoja yhdessä rakenteen komentojen kanssa.

Liitteessä 1 on esitelty tärkeimpiä ohjelmoinnin peruskäsitteitä, jotka ovat oleellisia oppimateriaalin käytännön harjoituksille. Kattava ja päivitetty kuvaus jokaisesta komennosta esimerkkeineen on luettavissa Arduinon kotisivuilla Learning-välilehdessä Reference-otsikon alla. Opinnäytetyön teon hetkellä verkko-osoite sinne on http://arduino.cc/en/Reference/HomePage . (Arduinon kotisivut 2015.)

6.4 Kirjastot

Arduinon kotisivuilla on saatavissa myös kasvava määrä kirjastoja (engl. libraries) yleisimpiin käyttölaitteisiin. Kirjasto on eräänlainen ohjelman lisäosa, joka laajentaa luonnoksen käyttömahdollisuuksia esimerkiksi servomoottorin ohjaukseen. (Arduinon kotisivut 2015.) Aloitettaessa ohjelmointia luonnoksen alkuun sisällytetään harjoituksessa tarvittavien laitteiden kirjastot.

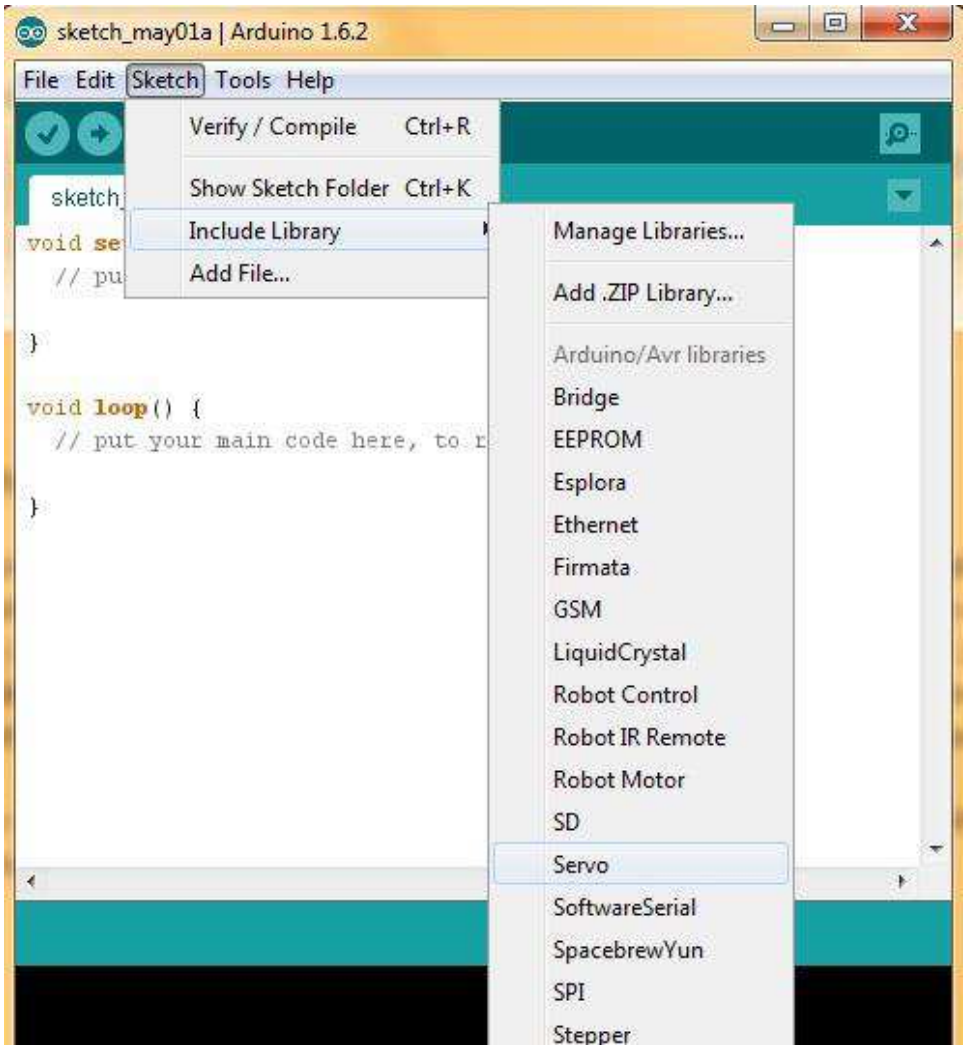
Jokaiselle käyttölaitteelle on oma kirjastonsa, eli SD-kortin kirjastolla ei voida ohjata TFT-näyttöä tai askelmoottoria. Myös laitteen valmistajalla, tyypillä ja mallilla on merkitystä, esimerkiksi saman valmistajan kaksi ulkoisesti samannäköistä LCD-näyttöä eivät välttämättä ohjaudu oikein samalla kirjastolla. Käyttölaitteita valitessa huomio kannattaakin kiinnittää ensimmäisenä toimivan kirjaston saatavuuteen kyseiselle laitteelle. Lisäksi laitevalmistaja on usein jollain tavalla ilmoittanut laitteen soveltuvuuden Arduinon kanssa käytettäväksi. Lisätiedoissa voi esimerkiksi lukea Arduino-yhteensopiva tai Arduinolle tehty (engl. Arduino-compatible, for Arduino).

Arduino IDE:ssä on mukana joitakin kirjastoja valmiina ja niitä voi ladata lisää internetistä tai luoda itse (Arduinon kotisivut 2015). Harjoitusta suunniteltaessa kannattaa tarkistaa onko internetissä saatavilla valmiita kirjastoja kyseiselle laitteelle tai selkeät ohjeet kirjaston luontiin. Kirjastoja on ladattavissa Arduinon kotisivujen lisäksi harrastajien ylläpitämillä sivustoilla ja keskustelupalstoilla. Luonnollisesti internetistä ladattuihin tiedostoihin kannattaa suhtautua harkiten ja tietyllä varauksella.

6.5 Kirjaston sisällyttäminen luonnokseen

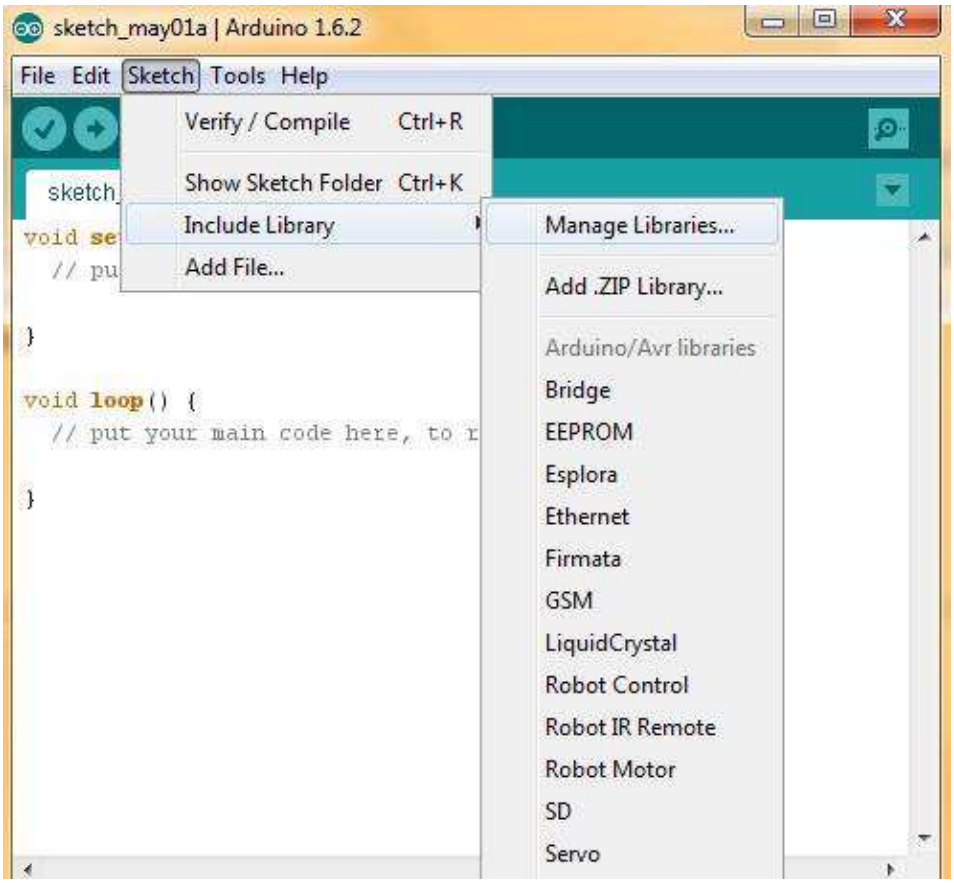
Arduinon kotisivuilla on ajantasainen ohje kirjaston sisällyttämisestä luonnoksen alkuun Learning-välilehdessä Reference-otsikon alla. Opinnäytetyön teon hetkellä verkko-osoite sinne on http://www.arduino.cc/en/Reference/Libraries . (Arduinon kotisivut 2015.) Ohjeiden ja Arduino IDE:n valmiiksi sisältämien kirjastojen (engl. standard libraries) kuvauksien lisäksi osoitteessa on listattuna kasvava määrä yleisesti jaossa olevia kirjastoja (engl. contributed libraries), jotka ovat alan harrastajien luomia.

Arduino IDE:n valmiiksi sisältämän kirjaston liittäminen luonnokseen on yksinkertaista. Käyttääksesi kirjastoa luonnoksessa, valitse ohjelman yläpalkista Sketch → Include Library → listasta haluttu kirjasto. Kuvassa 11 on havainnollistettu servomoottorin kirjaston lisääminen luonnokseen. Kun haluttu servomoottorin kirjasto on valittu, luonnoksen alussa tulee lukemaan *#include <Servo.h>* .

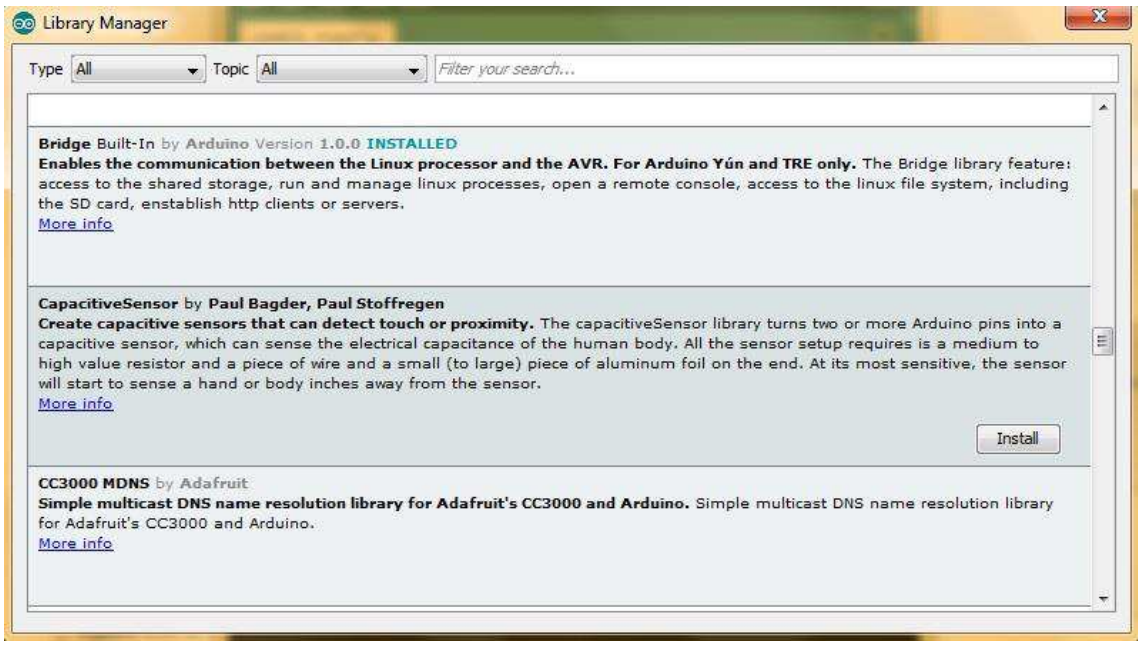


KUVA 11. Servomoottorin kirjaston lisääminen luonnokseen

Arduino IDE:tä täydentävien kirjastojen asentamiseen on myös selkeät ohjeet Arduinon kotisivuilla. Opinnäytetyön teon hetkellä verkko-osoite sinne on http://www.arduino.cc/en/Guide/Libraries . (Arduinon kotisivut 2015.) Haluttu kirjasto voidaan valita Manage Libraries -toiminnolla kuvien 12a ja 12b mukaisesti, tuoda zip-tiedostona tai asentaa manuaalisesti sovelluksen ollessa suljettuna. Jokaisesta toimenpiteestä on olemassa selkeät ohjeet päivityksineen Arduinon kotisivuilla, josta ne on suositeltava tarkastaa ennen asennusta. Asennuksen jälkeen kirjasto näkyy kuvassa 11 näkyvän Include Library -valikon listassa alimmaisena ja on sieltä valittavissa käyttöön.



KUVA 12a. Täydentävän kirjaston asentaminen Arduino IDE:n Manage Libraries -toiminnolla, osa 1/2



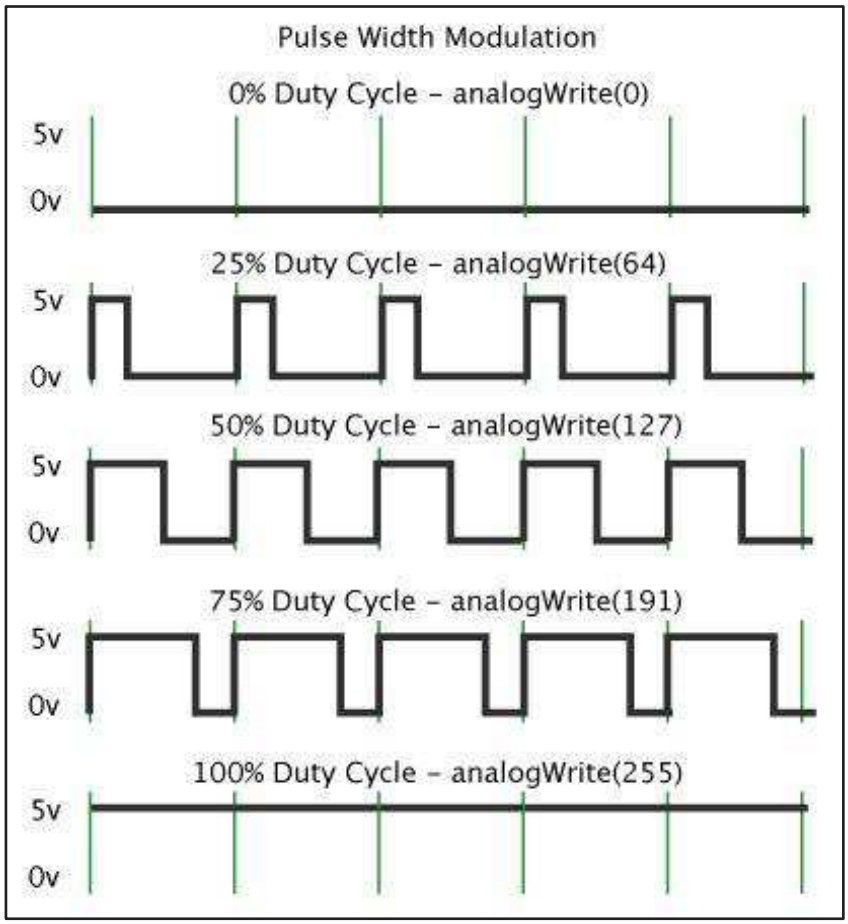
KUVA 12b. Täydentävän kirjaston asentaminen Arduino IDE:n Manage Libraries -toiminnolla, osa 2/2

7 PULSSINLEVEYSMODULAATIO (PWM)

Pulssinleveysmodulaatio eli PWM (engl. Pulse Width Modulation) on tapa saada analogisia tuloksia digitaalisin keinoin. Siinä hallitaan kuormaan syötettävää jännitettä säätämällä kanttiaaltomuotoisen jännitesyötön jaksojen pituudet tarpeen mukaan, toisin sanoen sillä simuloidaan eri käyttöjännitteitä. PWM-signaalin ylhäälläoloaika (päällä) suhteessa alhaallaoloaikaan (pois päältä) muodostaa riittävän suurella taajuudella toistettuna halutun jännitteen ja näin määrittää kytketyn laitteen toiminnan. Taajuus pidetään yleensä vakiona. Arduino Uno (R3) -mikrokontrollerialustan liitinnastoissa 3, 9, 10 ja 11 taajuus on 490 Hz ja liitinnastoissa 5 ja 6 taajuus on 980 Hz. (Arduinon kotisivut 2015; Hirzel 2015.)

Kuvassa 13 vihreät pystyviivat esittävät tiettyä säännöllistä ajanjaksoa, esimerkiksi kahta mikrosekuntia. Muodostunut kuvio on suoraan verrannollinen syötettävään jännitteeseen joka voi olla mitä tahansa täyden jännitteen (5 V) ja nollan (0 V) väliltä. Yhden ajanjakson ja ylhäälläoloajan suhdetta kutsutaan pulssisuhteeksi (engl. duty cycle) ja sitä kuvataan prosenttiluvulla, joka tavallaan kertoo kuormaan syötettävän jännitteen määrän. (Hirzel 2015.)

Pulssisuhteesta voidaan myös käyttää nimityksiä paloaikasuhde, käyttöjakso ja toimintajakso. Signaalin ylhäälläoloaikaa kutsutaan pulssin leveydeksi. Pulssin leveyttä muokkaamalla saadaan portaattomasti analogisia tuloksia. Arduinon funktio *analogWrite()* muuttaa ulos annettua PWM-arvoa välillä 0-255, mikä vastaa jännitettä välillä 0-5 V kuvan 13 mukaisesti. (Lazaridis 2009; Wheat 2011, 133; Shirriff 2013; Hirzel 2015.)



KUVA 13. PWM-signaali eri pulssisuhteilla (Hirzel 2015)

Pulssinleveysmodulaatiota käytetään usein himmentämään LED-valoja, luomaan audiosignaaleja ja ohjaamaan moottoreita, erityisesti servoja. PWM-ohjauksessa on tavallisesti 3-johtoinen kytkentä, jossa yksi johto on maa, toinen PWM-signaali ja kolmas käyttöjännite. On mahdollista käyttää myös 4-johtoista PWM-kytkentää, jossa neljäs johto palauttaa esimerkiksi takometrin signaalin moottorin kierrosnopeuden tarkastamiseksi. Pulssinleveysmodulaation käyttö säästää energiaa verrattuna esimerkiksi lineaariregulaattorilla säädettyyn jännitteeseen. Lineaariregulaattorilla säädettäessä syöttöjännitteen "ylimääräinen" teho haihdutetaan lämpöenergiana pois, kun vastaavasti pulssinleveysmodulaatiota käytettäessä pulssisuhdetta (päällä/pois päältä) muutellaan niin että saadaan ulostuloon haluttu keskiarvo jännitteelle. (Lazaridis 2009; Wheat 2011, 147; Shirriff 2013.)

Esimerkkejä pulssinleveysmodulaation käytöstä:

Ohjataan LED-valon kirkkautta pulssinleveysmodulaation avulla. Periaatteessa lediä vilkutetaan erittäin nopeasti ja kirkkauden vaihtelu mahdollistetaan säätämällä pulssien

päällä ja pois päältä -tilojen suhdetta eli pulssisuhdetta (engl. duty cycle). Asetetaan pulssisuhteeksi 25 %, jolloin LED-valo on päällä neljäsosan ajasta ja pois päältä kolme neljäsosaa. Ihmissilmä ei erota vilkuttamista, vaan näyttää siltä, että LED-valo palaa tasaisesti 25 % kirkkaudella. (Banzi 2011, 54.)

Ohjataan 180° kääntyvää servomoottoria PWM-ohjauksella. Pulssin leveys (ylhäälläoloaika) määrää servomoottorin tavoitekulman, jota kohti ohjauspiiri ohjaa moottoria. Yksi millisekunti ohjaa esimerkin servon äärivasemmalle, 1,5 ms keskiasentoon ja 2 ms äärioikealle. Ohjauspiirillä annettua ohjausjännitettä verrataan potentiometriltä mitattuun servon asennosta riippuvaan referenssijännitteeseen. Jos näiden välillä on eroa, ohjataan moottoria siten, että jännitteet ovat lopulta samat. (Lazaridis 2009.)

8 ADC/DAC

8.1 Analogia-digitaalimuunnin

Sensorit mahdollistavat ympäristön havainnoinnin reagoimalla fyysisiin muutoksiin sähköisesti. Usein tämä reaktio on resistanssin muutos, esimerkiksi venymäliuska-anturin sisältämän metallijohteen resistanssi muuttuu taivutuksen aiheuttaman venytyksen mukaan. Sensori siis muuntaa analogisen (fyysisen) muutoksen toiseksi analogiseksi (sähköiseksi) muutokseksi. Arduino-mikrokontrollerialustan sisältämä mikroprosessori ei ymmärrä analogisia arvoja, vaan toimii digitaalisella, ykkösistä ja nollista koostuvalla kielellä. Analogia-digitaalimuunninta eli ADC (engl. Analog to Digital Converter) käytetäänkin nimensä mukaisesti muuntamaan analogiset arvot digitaaliseksi jotta mikrokontrollerialustan suoritin voi niitä hyödyntää. Se on integroitu mikrokontrolleriin yhtenä sen oheislaitteista. (Premeaux & Evans 2011, 2.)

Arduino Uno (R3) -mikrokontrollerialustan 10-bittinen analogia-digitaalimuunnin on yhteydessä kuuteen analogiseen input-liitinnastaan (pinnit A0-A5). Se havainnoi tietyn pinnin jännitettä ja muuntaa jännitteen biteistä koostuvaksi lukuarvoiksi välillä 0-1023, joka nähdään tuloksena työmuistiin (RAM) määritellyssä muuttujassa. Arduino Uno (R3):n sisältämän muuntimen muunnosnopeus on mikrokontrollerin teknisten tietojen (2014, 237) mukaan 13 - 260 µs. Muunnosnopeudeksi (engl. conversion time) kutsutaan sitä aikaa, joka analogia-digitaalimuuntimelta kuluu muunnoksen suorittamiseen. (Premeaux & Evans 2011, 2; ATmega328 Datasheet 2014, 237.)

Yksinkertaistettuna, analogia-digitaalimuunnin suorittaa vain vertailun liitinnastan jännitteen ja määritellyn referenssijännitteen välillä. Referenssijännitteen suuruinen jännite kuvastaa suurinta mahdollista lukuarvoa (1023). Referenssijännitettä suuremmat jännitteet näkyvät samana, suurimpana mahdollisena arvona riippumatta siitä, kuinka paljon referenssijännitettä suurempia ne ovat. (Premeaux & Evans 2011, 2.)

Analogia-digitaalimuuntimen käyttämä referenssijännite voidaan määritellä kolmella eri tavalla. Ensimmäinen tapa on käyttää piirilevyn päävirtalähteen tuottamaa käyttöjännitettä, mikä on yleensä 3,3 V tai 5 V. Tämä tapa on Arduino Uno -mikrokontrollerialustan oletusasetus, vaikka sen suhteen voi ilmetä ongelmia

syöttöjännitteen laskiessa esimerkiksi paristokäytöllä, jolloin myös referenssijännite laskee. Referenssijännite voi myös hetkellisesti tippua kytkettäessä päälle raskaita toimilaitteita kuten servomoottoreita. (Premeaux & Evans 2011, 2.)

Toinen vaihtoehto on käyttää referenssijännitteenä suhteellisen tasaisena pysyvää mikrokontrollerin sisäistä referenssijännitettä, jonka arvo vaihtelee suorittimesta riippuen. Esimerkiksi Atmel Corporation:n valmistamien mikroprosessorien referenssijännite on 1,1 V ATmega328-mikrokontrollerille ja valittavissa oleva, joko 1,1 V tai 2,56 V ATmega2560-mikrokontrollerille (ATmega328 Datasheet 2014, 237; ATmega2560 Datasheet 2014, 268). Kolmas keino on syöttää oma referenssijännite suoraan mikrokontrollerialustalle AREF-pinnin kautta, mutta tällöin on noudatettava valmistajan ohjeita huollellisesti ja taattava referenssijännitteen vakaus esimerkiksi lineaariregulaattoria käyttäen. (Premeaux & Evans 2011, 2–4.)

8.2 Resoluutio

Analogia-digitaalimuuntimen resoluutio määräytyy käytettävän referenssijännitteen ja bittimäärän mukaan. n-bittinen luku voi saada tiloja 2^n kappaletta. Ensimmäiseksi tilaksi määritellään aina nolla, joten tilan maksimilukuarvo on 2^n –1. Esimerkiksi 8-bittisen luvun maksimilukuarvo on

$$2^8 - 1 = 255$$

8-bittisen luvun minimilukuarvo on nolla. Välillä 0-255 on siis yhteensä 256 eri lukuarvoa. 8-bittinen analogia-digitaalimuunnin voi siis saada tiloja 256 kappaletta.

Arduino Uno (R3) -mikrokontrollerialustassa on 10-bittinen analogia-digitaalimuunnin (ATmega328 Datasheet 2014, 237). Se voi saada tiloja edellä käsitellyn laskutoimituksen mukaisesti 1024 kappaletta. Analogia-digitaalimuuntimen muuntamat lukuarvot voivat siis vaihdella välillä 0-1023. (Mäkelä 2015.)

Tarkastellaan tapausta, jossa referenssijännitteenä käytetään Arduino Uno:n käyttöjännitettä 5 V. Jännitteen maksimiarvo 5 V vastaa analogia-digitaalimuuntimen suurinta mahdollista lukuarvoa 1023. Vastaavasti 0 V vastaa minimilukuarvoa nolla. Voidaan helposti päätellä, että 2,5 V vastaa lukuarvoa 512. Lukuarvoja vastaavat

jännitteen arvot voidaan laskea jakamalla käytettävä referenssijännite mahdollisten tilojen määrällä ja kertomalla saatu tulos kysytyllä lukuarvolla.

Esimerkiksi selvitetään mikä Arduino Uno (R3):n analogiseen sisääntulopinniin tuleva jännite vastaa lukuarvoa 700, kun käytössä on referenssijännitteen oletusasetus (sitä ei siis ole muokattu). Laskutoimitus on

$$5\ \text{V} / 1024 = 4{,}8828\ \text{mV}$$
$$4{,}8828\ \text{mV} \cdot 700 = 3{,}42\ \text{V}$$

Jossa 5 V on referenssijännite (oletusasetuksena on piirilevyn käyttöjännite) ja
1024 on mahdollisten tilojen määrä (riippuu bittimäärästä)

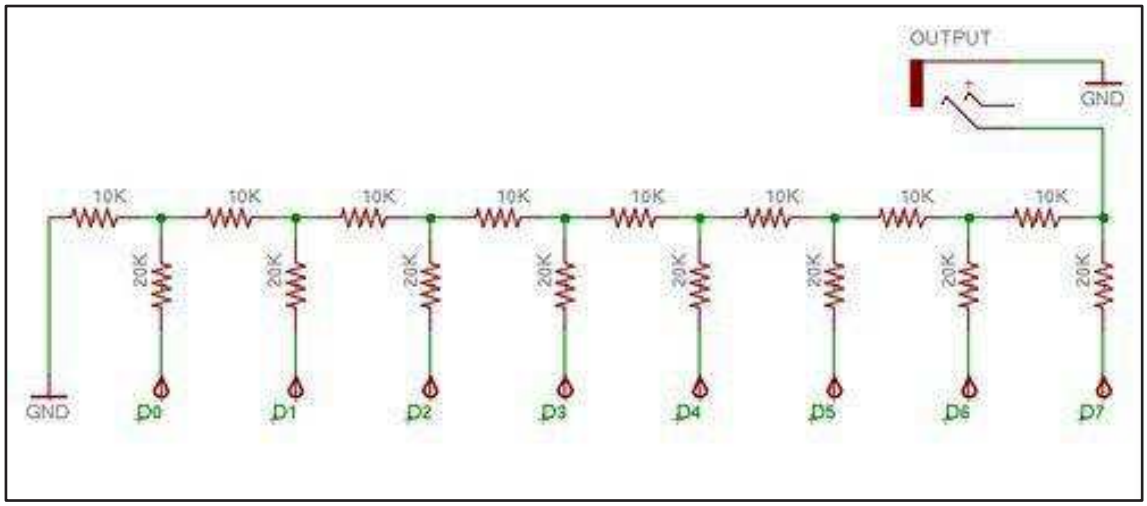
Saatua välitulosta 4,8828 mV kutsutaan kyseisen tapauksen resoluutioksi. Se on tarkkuus, joka vastaa yhtä askelta tilojen määrässä. Tarkkuutta voidaan lisätä kahdella tapaa. Ensimmäinen keino on kasvattaa bittimäärää, jolloin nimittäjä suurenee jolloin saatu tulos eli resoluutio on pienempi eli tarkempi. Askeleita on siis tiheämmin. (Premeaux & Evans 2011, 3; Mäkelä 2015.)

Toinen vaihtoehto on muuttaa referenssijännitettä pienemmäksi, jolloin osoittaja pienenee ja resoluutio jälleen tarkentuu. Jälkimmäinen tapa on mahdollinen lähinnä käytettäessä Arduino Mega 2560 - mikrokontrollerialustaa, jonka ATmega2560-mikrokontrollerin ansiosta valittavissa oleva 2,56 V sisäinen referenssijännite on potentiaalinen vaihtoehto. Arduino Uno (R3) -mikrokontrollerialustan tarjoama 1,1 V sisäinen referenssijännite ei ole ideaali, sillä monet sensorit eivät toimi tällä jännitealueella ollenkaan. (Premeaux & Evans 2011, 3.)

8.3 Digitaali-analogiamuunnin

Toisinaan tarvitaan analogista ulostuloa digitaalisen sijaan, esimerkiksi moottorin pyörimisnopeuden ohjaukseen tai äänien tuottamiseen. Digitaali-analogiamuuntimella eli DAC:lla (Digital to Analog Converter) muunnetaan digitaalinen signaali analogiseksi jännitteeksi. Toisin kuin AD-muuntimia, DA-muuntimia on harvoin integroitu mikrokontrollerin yhteyteen. Tällöin on toteutettava muunnos vaihtoehtoisten ratkaisujen avulla. Niitä ovat esimerkiksi muunnoksen suorittaminen käyttäen suoraan

mikrokontrollerin lähtöliitäntöihin kytkettyä R-2R -vastusverkkoa (kuva 14) tai yksinkertaista ulkoista DAC:a. (Tooley 1990, 190; Tietze 2007, 945, 953.)



KUVA 14. 8-bittisenä DA-muuntimena käytettävä yksinkertainen R-2R -vastusverkko (Cunningham 2008)

Pulssinleveysmodulaatio eli PWM on yksi vaihtoehto tuottaa analogista jännitesignaalia. Sen tuottamien jännitetasojen lukumäärä riippuu bittimäärästä. Arduino Uno (R3) -mikrokontrollerialustan 8-bittinen muunnin kykenee tuottamaan 2^8 eli 256 eri jännitetasoa, mikä yleensä riittää useimpiin sovellutuksiin. Lisää tietoa mikrokontrollerialustojen käyttämistä prosessoreista sekä niiden mahdollisuuksista DA-muuntamiseen on saatavissa mikrokontrollerivalmistajan, kuten Atmel Corporation:n, tuottamista teknisistä tiedoista eli datasheet:sta.

9 TUOTEKEHITYSPROJEKTINA LUOTU ARDUINO-TELINE

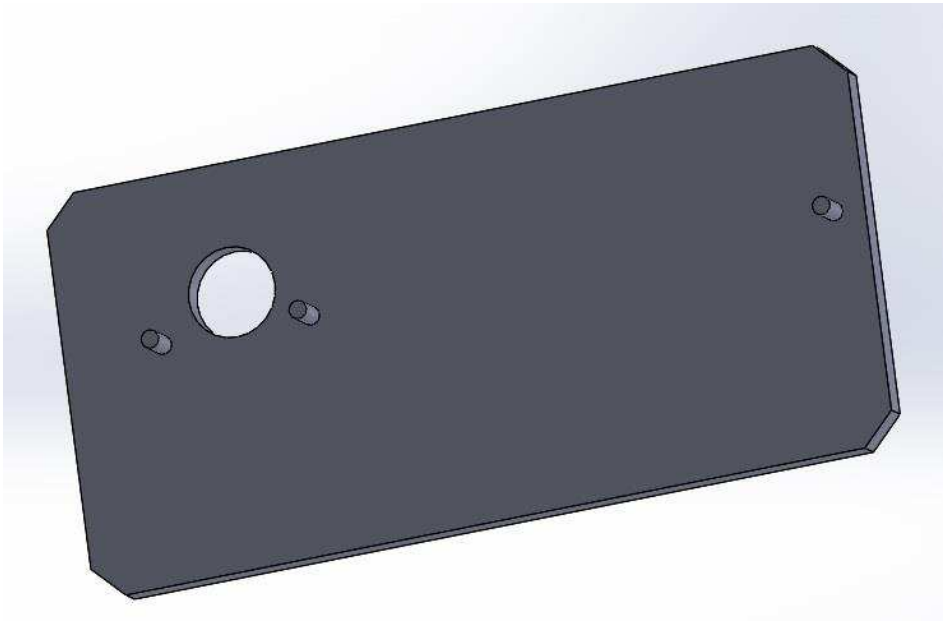
Oppimateriaaliin kuuluvia harjoituksia testattaessa havaittiin rakenteellinen ongelma laboratoriovälineisiin kuuluvassa kytkentäalustassa. Kytkentäalustasta puuttui tukeva ja looginen paikka Arduino-mikrokontrollerialustalle, jolloin harjoituksia suoritettaessa Arduino saattoi liukua ja tipahtaa kytkentäalustan päältä. Tämä luonnollisesti vaikeutti kytkentöjen kiinnittämistä ja pahimmassa tapauksessa mahdollisti komponenttien rikkoutumisen. Lisäksi kytkentäalustan materiaali on sähköä johtavaa terästä, joka on päällystetty ohuella maalikerroksella. Maalipinnan rikkoontuminen voi johtaa oikosulkuun ja komponenttien rikkoutumiseen.

Tuotekehitysprojektin tavoitteeksi asetettiin suunnitella helppokäyttöinen ja turvallinen teline Arduinon kiinnittämiseksi kytkentäalustaan. Telineen tulisi olla helposti irroitettavissa sekä kytkentäalustasta. että Arduinosta. Se ei myöskään saisi sisältää useita irtonaisia osia, jotteivat ne katoaisi. Valmistuskustannukset pyritään pitämään mahdollisimman edullisina eikä telineen asennus kytkentäalustaan saisi aiheuttaa ylimääräisiä työvaiheita tai muutoksia kytkentäalustan rakenteeseen.

Telineen ensimmäinen versio on suunniteltu Arduino Uno - mikrokontrollerialustalle. Työn tilaajan pyynnöstä muokkasin telineen soveltuvaksi myös fyysisesti kookkaammalle Arduino Mega 2560 - mikrokontrollerialustalle tulevaisuuden laitehankintoihin varautuen. Tuloksena kehitetyn telineen kiinnitystapit sopivat siis molemmille Arduinon mikrokontrollerialustoille sekä myös muille piirilevyille jotka ovat valmistettu samalla standardirakenteella. Telineen suorakaiteen muotoiset tuet koskettavat molemmat mikrokontrollerialustat huomioon ottaen piirilevyn pintaa, jossa ei ole juotoksia ja ehkäisevät erityisesti isompaan Arduino Mega 2560 - mikrokontrollerialustaan kohdistuvaa taivutusta kytkentöjä suoritettaessa. Telineen kytkentäalustan puolta suunniteltaessa on tavoiteltu mahdollisimman kestävää mutta helppoa kiinnitystä sekä huomioitu kytkentäalustasta ulkoneva ruuvin kanta.

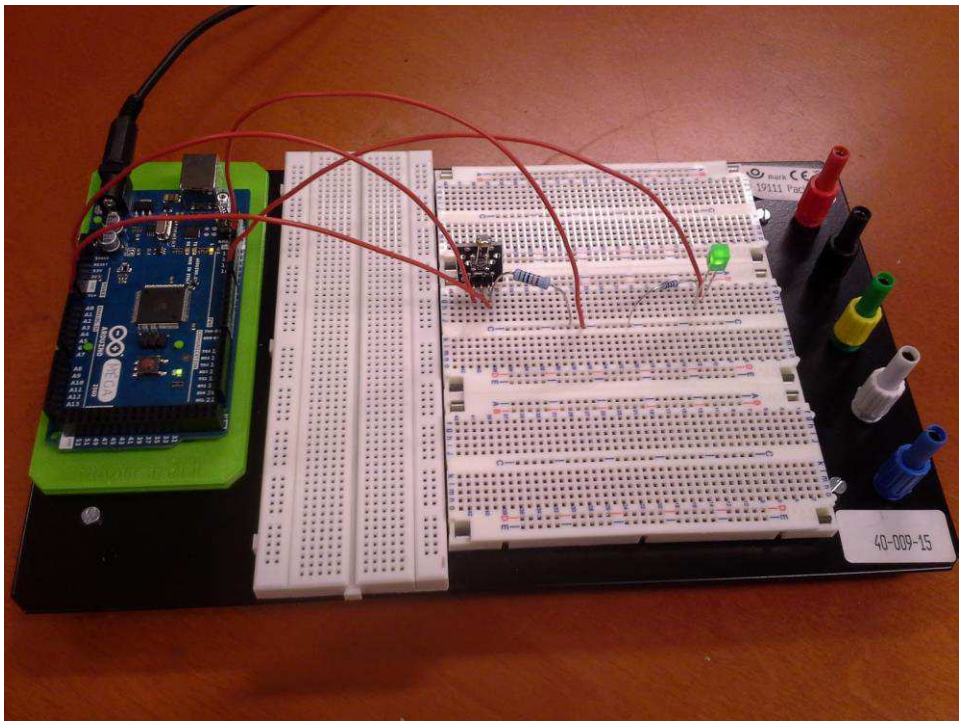
Suunnittelin tarkoitukseen sopivan telineen Dassault Systemes:n SOLIDWORKS-ohjelmistolla. Kuvassa 15 on mallinnettu teline Arduinon puolelta ja kuvassa 16 kytkentäalustan puolelta esitettynä.

KUVA 15. Mallinnettu teline Arduinon puolelta



KUVA 16. Mallinnettu teline kytkentäalustan puolelta

Tampereen ammattikorkeakoululla on käytettävissä useita 3D-tulostimia joita voitiin hyödyntää telineen valmistukseen. Valitsin telineen valmistukseen aikaisempien henkilökohtaisten käyttökokemusten perusteella Ultimaker2 - tulostimen ja materiaaliksi PLA-muovin, joka on halpaa ja sähköä johtamatonta. Teline tulostetaan geometrian vuoksi kahdessa osassa jotka yhdistetään kaksoiskomponenttiliimalla. Vaihtoehtoisesti olisin voinut tulostaa telineen käyttäen Stratasys-tulostinta jolloin tukimateriaalin käyttö olisi mahdollistanut telineen tulostamisen yhtenä kappaleena. Tällöin kuitenkin valmistuskustannukset kasvaisivat suhteettomasti säästettyyn työmäärään nähden. Kuvassa 17 on valmis teline suunnitelman mukaisessa käytössä.



KUVA 17. Arduino-teline, johon on kiinnitetty Arduino Mega 2560 -mikrokontrollerialusta

10 POHDINTA

Opinnäytetyön tavoitteena oli kehittää toimiva oppimateriaalikokonaisuus sulautettujen järjestelmien ja Arduino-mikrokontrollerialustan opetukseen Tampereen ammattikorkeakoulun koneautomaation opintopolun tarpeisiin. Tavoitteeseen päästiin käymällä läpi ja opettelemalla ensin itse huomattava määrä tietoa, josta prosessoimalla muodostui käytäntöön painottuva ja selkeä kokonaisuus opetuskäyttöön.

Harjoituksia on koottu materiaaliin kymmenkunta ja ensimmäiset niistä ovat tarkoituksella suhteellisen yksinkertaisia, jotta jokainen oppija ymmärtää ohjelman pääperiaatteen tavoitteiden mukaisesti. Harjoitukset on myös kehitetty helposti muokattaviksi käytettävissä olevan opetusvälineistön mukaan. Osaan niistä onkin kirjoitettu esimerkkejä vaihtoehtoisista komponenteista ja tehtävän vaikeusasteen lisäämisestä. Oppimateriaaliin kuuluvat harjoitukset sekä niiden kytkennät testattiin toimiviksi kohderyhmän mukaisella opiskelijajoukolla toukokuussa 2015. Testiryhmältä saadun palautteen perusteella harjoitukset viimeisteltiin lopulliseen muotoonsa.

Sulautettujen järjestelmien jatkuvasti kehittyessä on otettava huomioon opetettavan materiaalin ja tuotteiden ajankohtaisuus sekä uuden tiedon päivittyminen. Siitä syystä opinnäytetyössä on pyritty korostamaan internetin ja erityisesti tuotevalmistajien kotisivujen merkitystä luotettavana tiedonlähteenä tulevaisuudessa. Yksi luodulle oppimateriaalille asetetuista tavoitteista olikin muodostaa oppijalle vakaa pohja, johon syventävät opinnot jatkossa perustuvat.

Opinnäytetyönä kehitetyn oppimateriaalin tuloksena oppija perehtyy sulautettuihin järjestelmiin saaden käsityksen siitä, mikä Arduino on ja mihin sitä voidaan soveltaa. Lähtökohtaisesti vastaavaa materiaalikokonaisuutta ei ole ollut työn tilaajan käytössä aiemmin, vaan tieto on ollut hajautetusti saatavilla.

Lopputuloksena muotoutuneessa oppimateriaalissa käydään läpi ohjelmoinnin perusasiat sekä kannustetaan itsenäiseen kokeiluun, sillä jatkuva oppimisprosessi edellyttää aitoa mielenkiintoa parhaiden tulosten saavuttamiseksi. Arduino on monipuolinen työkalu ja varmasti ajankohtainen aihe sulautettujen järjestelmien parissa myös tulevaisuudessa. Seuraava kehityskohde voisi ollakin moniulotteisempien ja laajempien harjoitusten luominen oppimateriaalin jatkumona. Toinen visio on laajentaa

sulautettujen järjestelmien opetusta koskemaan esimerkiksi Rasberry Pi:tä, joka on yhden piirilevyn tietokone.

Oppimateriaalin kehittäminen on harvinainen opinnäytetyön aihe. Sen parissa työskenteleminen avasi aivan uusia ja mielenkiintoisia näkökulmia koneautomaatioon ja jopa yleisesti tekniikkaan. Ei riitä, että itse sisäistää asian, vaan tieto on saatava muodostettua ymmärrettävänä ja havainnollisena kokonaisuutena muiden käyttöön. Työ tarjosi sopivasti haastetta ja muodosti tekijälleen arvokasta kokemusta dokumenttien tuottamisesta. Lopullisen arvion oppimateriaalista voivat luonnollisesti antaa ainoastaan sen käyttäjät, opettajat ja oppijat.

LÄHTEET

Arduino. 2015. Arduino kotisivut. [www-sivu]. Luettu 8.3.2015. http://arduino.cc/en/Reference/HomePage

Atmel Corporation. 2014. ATmega328 Datasheet. [www-dokumentti]. Luettu 13.3.2015. http://www.atmel.com/images/Atmel-8271-8-bit-AVR-Microcontroller-ATmega48A-48PA-88A-88PA-168A-168PA-328-328P_datasheet_Complete.pdf

Atmel Corporation. 2014. ATmega2560 Datasheet. [www-dokumentti]. Luettu 13.3.2015. http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561_datasheet.pdf

Bajdi. 2015. Bajdi kotisivut. [www-sivu]. Luettu 1.3.2015. http://www.bajdi.com/wp-content/uploads/2012/04/nano-uno-mega2560.jpg

Banzi, M. 2011. Arduino – perusteista hallintaan. 2. painos. Suom. Mäenpää, Y. Hämeenlinna: Robomaa.com Oy.

CircuitsToday. 2011. Basics of Microcontrollers. [www-sivu]. Luettu 24.2.2015. http://www.circuitstoday.com/basics-of-microcontrollers

Cunningham, C. 2008. Make:It -kotisivut. Proto-DAC shield for Arduino. [www-sivu]. Luettu 1.5.2015. http://makezine.com/2008/05/29/makeit-protodac-shield-fo/?CMP=OTC-0D6B48984890

Hirzel, T. 2015. Arduinon kotisivut. PWM. [www-sivu]. Luettu 8.3.2015. http://www.arduino.cc/en/Tutorial/PWM

Karvinen, K. & Karvinen, T. 2011. Make: Arduino Bots and Gadgets. 1. painos. Sebastopol: O'Reilly Media.

Koli, H. & Silander, P. 2002. Oppimisprosessin suunnittelu ja ohjaus. 1. painos. Hämeenlinna: Hämeen ammattikorkeakoulu.

Koskinen J. 2004. Mikrotietokonetekniikka. Uudistettu 1. painos. Helsinki: Kustannusosakeyhtiö Otava.

Lazaridis, G. 2009. PCBheaven kotisivut. How RC Servos Work. [www-sivu]. Luettu 8.3.2015. http://pcbheaven.com/wikipages/How_RC_Servos_Works/

Lehtonen T., Lätti E., Similä H. & Tammisto A. 2014. 3D-tulostettu robotti. Mekatroniikan raportti. Tampere: Tampereen ammattikorkeakoulu.

Mäkelä, S. Laboratorioinsinööri. 2015. Haastattelu 15.1.2015. Haastattelija Seipäjärvi, A-E. Tampere.

PICAXE. 2015. PICAXE kotisivut. [www-sivu]. Luettu 26.2.2015. http://www.picaxe.com/What-Is-PICAXE

Premeaux, E. & Evans, B. 2011. Arduino Projects to Save the World. 1. painos. New York City: Apress.

Protostack. 2015. ATmega328 -mikrokontrolleri. [www-sivu]. Luettu 1.5.2015. http://www.protostack.com/microcontrollers/atmega328-pu-atmel-8-bit-32k-avr-microcontroller

Rasberry Pi. 2015. Rasberry Pi kotisivut. [www-sivu]. Luettu 26.2.2015. http://www.raspberrypi.org/help/what-is-a-raspberry-pi/

Shirriff, K. 2013. Ken Shirriff:n blogi. Secrets of Arduino PWM. [www-sivu]. Luettu 8.3.2015. http://www.righto.com/2009/07/secrets-of-arduino-pwm.html

STMicroelectronics. 2015. STMicroelectronics kotisivut. [www-sivu]. Luettu 26.2.2015. http://www.st.com/web/en/home.html

Suominen, R. & Nurmela, S. 2011. Verkko-opettaja. 1. painos. Helsinki: WSOYpro Oy.

Texas Instruments. 2015. Texas Instruments kotisivut. [www-sivu]. Luettu 26.2.2015. http://www.ti.com/ww/en/launchpad/about.html

Tietze, U., Schenk, Ch. & Gamm, E. 2007. Electronic circuits: handbook for design and application. 2. painos. New York: Springer.

Tooley, M. 1990. The Maplin electronic circuits handbook. Uudistettu 1. painos. Lontoo: Butterworth-Heinemann Ltd.

Wheat, D. 2011. Arduino internals. 1. painos. New York City: Apress.

LIITTEET

Liite 1. **Ohjelmoinnin peruskäsitteet**

Lähteet:

Banzi, M. 2011. Arduino – perusteista hallintaan. Liite C/Arduinon pikaopas. 2. painos. Suom. Mäenpää, Y. Hämeenlinna: Robomaa.com Oy.

Arduino. 2015. Arduino kotisivut. [www-sivu]. Luettu 8.3.2015. http://arduino.cc/en/Reference/HomePage

Kattava kuvaus jokaisesta komennosta esimerkkeineen on luettavissa Arduinon kotisivuilla Learning-välilehdessä Reference-otsikon alla. Opinnäytetyön teon hetkellä verkko-osoite sinne on http://arduino.cc/en/Reference/HomePage .

Arduino-ohjelman eli luonnoksen rakenne koostuu kahdesta osasta:

void setup()
Tämä on luonnoksen ensimmäinen osa. Tähän sisältyy kaikki alustamista varten kirjoitettu koodi eli komennot, joilla kortti valmistellaan tulevaa ohjelmasilmukkaa varten. Tämä koodilohko luetaan vain kerran, aina kun piirilevyyn kytketään virta päälle tai kun reset-nappia painetaan. Tätä osaa käytetään esimerkiksi kun asetetaan liitinnastoja input- tai output-tilaan, nollataan alkuarvot, aloitetaan sarjaliikenne tietyllä nopeudella tai sisällytetään luonnokseen mahdollisia tarvittavien käyttölaitteiden kirjastoja. Lue lisää kirjastoista kohdasta “Arduino IDE (kehitysympäristö) - Processing-kieli - Kirjastot”.

void loop()
Tässä osassa sijaitsee ohjelman pääkoodi eli itse toiminto. Siinä olevien komentojen muodostamaa kokonaisuutta toistetaan luonnoksen mukaisesti loppumattomana silmukkana kunnes piirilevystä katkaistaan virta.

Erityissymbolit, joilla jäsennetään koodia, kommentteja tai koodilohkoja:

; (puolipiste)
Jokainen komento tai yksittäinen lauseke päättyy puolipisteeseen. Siitä kääntäjä (se osa Arduinoa, joka muuntaa luonnoksen mikrokontrollerin ymmärtämäksi konekieliseksi ohjelmaksi) tietää, että yksi lauseke päättyy ja toinen alkaa. Tämän unohtaa helposti luonnosta kirjoittaessa.

{} (aaltosulkeet)
Aaltosulkeita käytetään kokoamaan koodi yhdeksi kokonaisuudeksi. Jos esimerkiksi kirjoittaa koodia *loop()*-funktioon, on se suljettava aaltosulkeiden sisään kokonaisuudessaan. Avaava aaltosulje *{* tarvitsee aina parikseen sulkevan aaltosulkeen *}*.

#include
Tätä käytetään kun sisällytetään luonnokseen tarvittavien käyttölaitteiden kirjastoja.

Kommentit voidaan merkitä kahdella tapaa:

//
Kaksi kenoviivaa aloittaa yhden rivin kommentin, joka ohitetaan rivin loppuun saakka.

*/**
...tähän väliin voi kirjoittaa vaikka kuinka paljon,
riviltä toiselle…
/
Kenoviiva ja tähti avaavat kommenttialueen, joka jatkuu riviltä toiselle kunnes alue suljetaan tähdellä ja kenoviivalla. Kommentoinnin lisäksi tämä on kätevä ominaisuus kun luonnoksesta paikallistetaan virhettä, koodin voi pätkä kerrallaan sulkea kommenttialueelle “karanteeniin” ja kokeillen selvittää mikä osa luonnosta ei toimi.

Vakiot:

Arduinossa on joukko valmiiksi määriteltyjä vakioarvoja tiettyjä erityistarkoituksia varten. Niitä ovat esimerkiksi:
HIGH ja *LOW*
Näitä käytetään asettamaan Arduinon tietty liitinnasta päälle (HIGH) tai pois päältä (LOW).

INPUT ja *OUTPUT*
Nämä asettavat liitinnastan toimimaan joko input- tai output-tilassa.

Muuttujat:

Muuttujat ovat Arduinon työmuistissa olevia nimettyjä alueita, joihin voi tallentaa tietoa. Muuttujien tieto voi muuttua niin usein kun on tarvetta. Koska Arduinon prosessori on yksinkertainen, sille on ilmoitettava muuttujan tyyppi kun se esitellään ensimmäisen kerran. Tyyppi kertoo muuttujaan tallennettavan tiedon kokoluokan.

int
Tämä on yleisin Arduinossa käytetty tietotyyppi, joka viittaa sanaan integer eli kokonaisluku. Arduino Uno:ssa se käyttää kaksi tavua muistitilaa ja siihen voi tallentaa numerot -32 768 – 32 767. Luvun saavuttaessa äärirajansa eli minimin tai maksimin, se pyörähtää ympäri aloittaen uuden kierroksen. Tämä tapahtuu molempiin suuntiin laskettaessa. Esimerkkitapaus: 32 767 + 3 ei olekaan 32 770, vaan kierros pyörähtää ympäri ja vastaus onkin -32 765.

Muita käytettyjä tietotyyppejä ovat muun muassa *boolean, char, byte, unsigned int, long, unsigned long, float, double, string* ja *array*.

Ohjausrakenteet:

Arduinossa on joukko avainsanoja, jotka kontrolloivat ohjelman suoritusta.

if...else
Tämä on ehtolause, jonka avulla ohjelmassa voidaan tehdä päätöksiä. if-termiä seuraa suluissa oleva lauseke, joka on kuin väittämä tai testikysymys. Jos lauseke on tosi, ohjelma suorittaa sen mikä seuraa välittömästi if-osaa. Jos väittämä on epätosi, ohjelman suoritus jatkuu kohdasta else, jos sellainen löytyy. On siis mahdollista käyttää if-osaa ilman perässä tulevaa else-osaa.

Muita käytettyjä ohjausrakenteita ovat muun muassa *for, switch case, while, do...while, break, continue* ja *return*.

Aikafunktiot:

Arduinossa on funktioita kuluvan ajan mittaamista ja taukojen luomista varten.

delay(ms)
Pysäyttää ohjelman annettujen millisekuntien ajaksi.

Muita käytettyjä aikafunktioita ovat muun muassa *millis(), micros()* ja *delayMicroseconds()*.

INPUT- ja OUTPUT-funktiot:

Arduinoon sisältyy funktioita, joilla käsitellään sisään tulevaa (input) ja ulospäin menevää (output) dataa.

pinMode(nasta,tila)
Tämä määrittää digitaalisen liitinnastan toimimaan joko input- tai output-tilassa.

int digitalRead(nasta)

Tämä lukee input-nastan tilan ja tallentaa muuttujaan int arvon HIGH jos liittimessä on jännite, ja arvon LOW jos siinä ei ole jännitettä. Liitinnasta täytyy ensin asettaa input-tilaan *pinMode()*-funktiolla, ennen kuin *digitalRead()*-funktio toimii. Jos pinniä ei ole kytketty mihinkään, *digitalRead()*-funktio antaa arvon HIGH tai LOW sattumanvaraisesti. Huomaa, että myös analogisia sisääntulopinnejä voi käyttää digitaalipinneinä, silloin niihin viitataan nimellä A0, A1, jne.

digitalWrite(nasta,vakio)

Tämä vaihtaa digitaalisen liitinnastan joko päälle (HIGH) tai pois päältä (LOW). Liitinnasta täytyy ensin asettaa output-tilaan *pinMode()*-funktiolla, ennen kuin *digitalWrite()*-funktio toimii. Huomaa, että myös analogisia sisääntulopinnejä voi käyttää digitaalipinneinä, silloin niihin viitataan nimellä A0, A1, jne.

int analogRead(nasta)

Tämä lukee analogisessa sisääntulopinnissä vallitsevan jännitteen ja kertoo numeerisen arvon välillä 0-1023, mikä on suoraan verrannollinen jännitteen arvoon 0-5 V. Sen jälkeen se tallentaa numeerisen arvon muuttujaan int. Liitinnasta täytyy ensin asettaa input-tilaan *pinMode()*-funktiolla, ennen kuin *analogRead()*-funktio toimii. Arduino Uno lukee vallitsevan jännitteen nopeimmillaan 10 000 kertaa sekunnissa. Jos pinniä ei ole kytketty mihinkään, analogRead()-funktio antaa arvoja sattumanvaraisesti.

analogWrite(nasta,vakio)

Tämä muuttaa ulos annettua PWM (pulssinleveysmodulaatio) arvoa välillä 0-255, mikä vastaa jännitettä välillä 0-5 V. *analogWrite()*-funktio toimii Arduino Uno:n liitinnastoissa 3, 5, 6, 9, 10 ja 11, koska niihin on rakennettu toiminnon mahdollistava komponentti. PWM-taajuus pinneissä on yleensä 490 Hz, mutta Arduino Uno:n pinneissä 5 ja 6 taajuus on 980 Hz. Huomaa, että PWM-liitinnastat ovat automaattisesti output-tilassa, niitä ei siis tarvitse itse asettaa *pinMode()*-funktiolla. *analogWrite()*-funktio ei myöskään liity analogisiin sisääntulopinneihin, vaan on oma erillinen asiansa.

Muita käytettyjä funktioita ovat muun muassa *shiftOut()* ja *unsigned long pulseIn()*.

Sarjaliikennefunktiot:

Arduino Uno:ssa on sarjaliikenneprotokollaa hyödyntävä USB-portti.

Serial.begin(nopeus)
Tämä valmistaa Arduinon aloittamaan sarjamuotoisen tiedon lähettämisen tai vastaanottamisen. Arduino IDE:n monitoroinnissa nopeutena on tavallisesti 9600 bps (bittiä sekunnissa), mutta muitakin nopeuksia on mahdollista käyttää.

Serial.print(data)
Serial.print(data,kryptaus)
Tämä lähettää dataa sarjaportille. Tiedon kryptaus on valinnainen vaihtoehto.

Serial.println(data)
Serial.println(data,kryptaus)
Tämä toimii kuten *Serial.print(data),* paitsi se lisää kappaleenvaihdon ja rivinvaihtokoodin datan luettavuuden lisäämiseksi. Tiedon kryptaus on valinnainen vaihtoehto.

int Serial.read()
Tämä lukee tavu kerrallaan sisään tulevaa sarjamuotoista dataa.

Serial.flush()
Koska dataa voi tulla sarjaporttiin nopeammin kuin ohjelma ehtii sitä käsitellä, Arduino säilyttää kaiken sisään tulevan datan puskurissa. Jos haluaa tyhjentää puskurin ja antaa sen täyttyä tuoreella tiedolla voi käyttää tätä funktiota.

Liite 2. **Harjoitukset**

Lähteet:

Banzi, M. 2011. Arduino – perusteista hallintaan. 2. painos. Suom. Mäenpää, Y. Hämeenlinna: Robomaa.com Oy. (Osittain mukailtu.)

Harjoitus 1: Hello world (vilkkuva LED)

Tavoite:

Vilkuttaa LED-valoa.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- LED-valo

Tarvittavat funktiot/komennot:

- const int
- void setup()
- void loop()
- pinMode()
- digitalWrite()
- delay()

Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

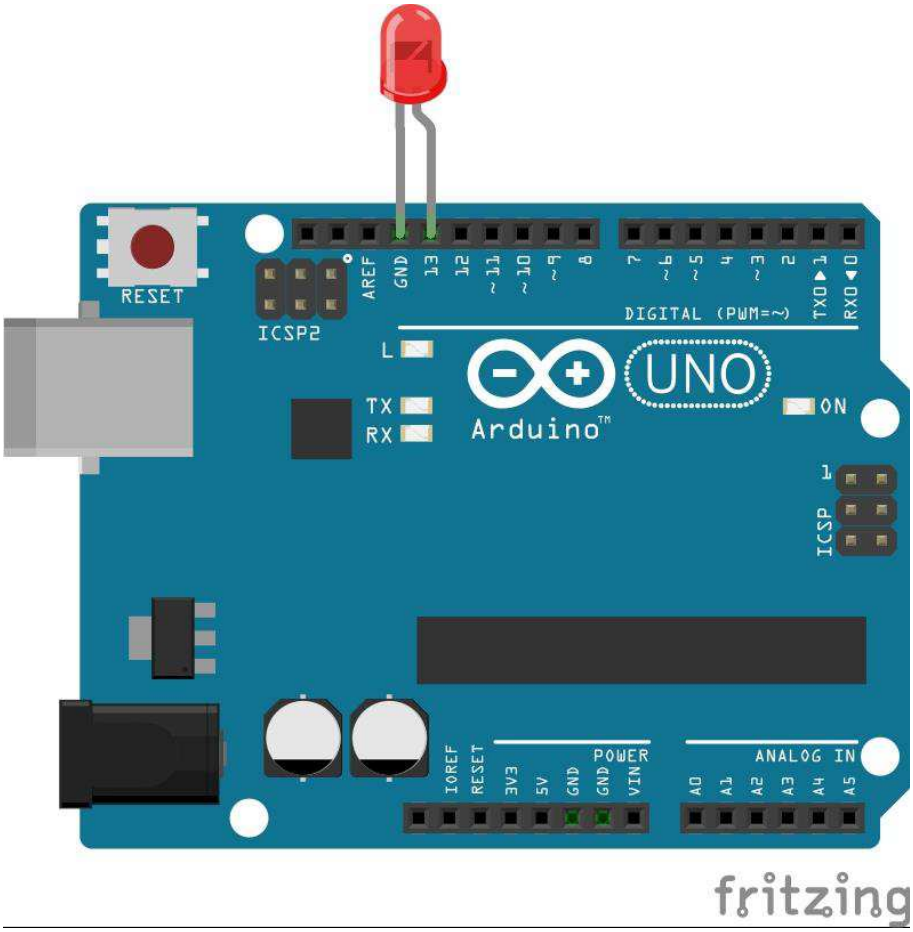
Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.

- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 2: Pidetään LED päällä painonapilla

Tavoite:

Pidetään LED-valoa päällä painamalla nappia. Käyttämällä *digitalRead()*-funktiota "kysytään" Arduinolta onko nappi painettuna vai ei. LED-valo palaa vain kun nappi on alas painettuna.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 10 kΩ vastus
- LED-valo
- painonappikytkin

Tarvittavat funktiot/komennot:

- const int
- int
- void setup()
- pinMode()
- void loop()
- digitalRead()
- if ()
- digitalWrite()
- else

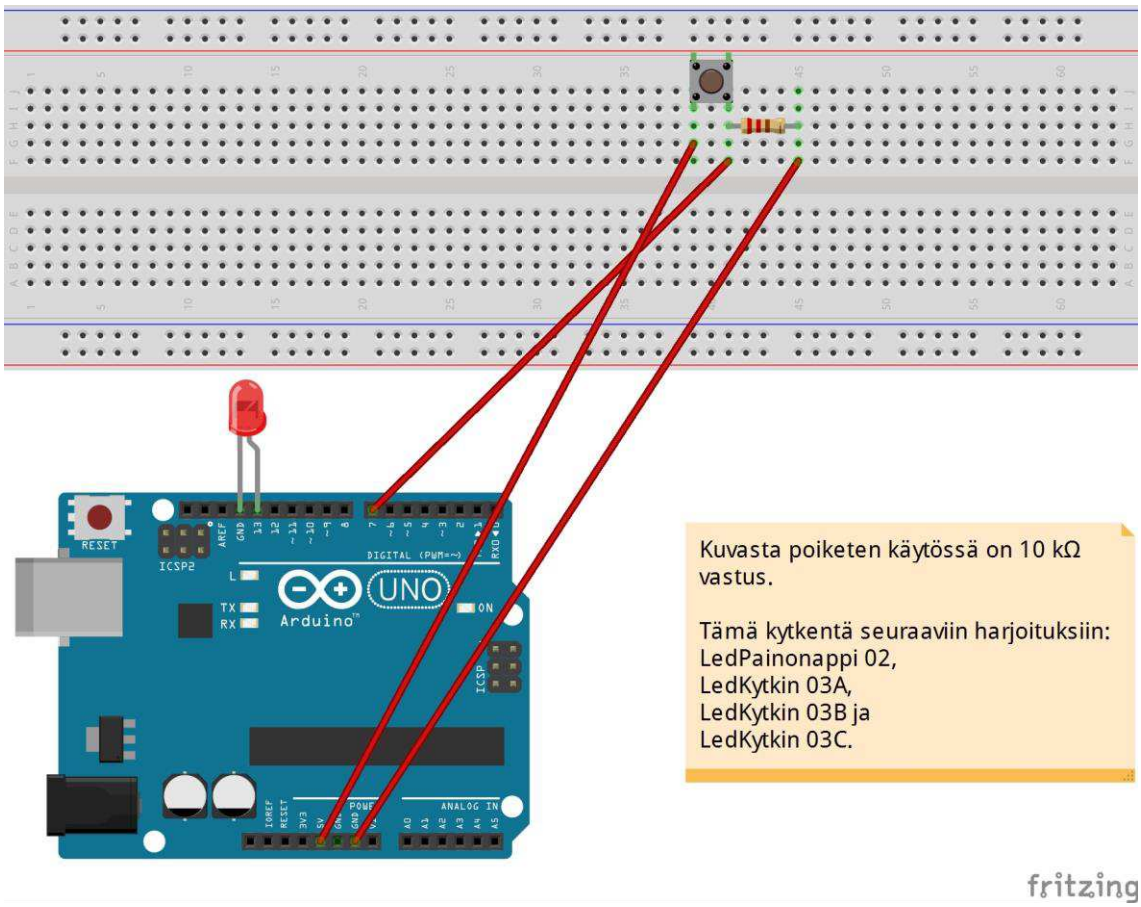
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 3A: Kytketään LED päälle ja pois osa 1/3

Tavoite:

Kytketään LED-valo painokytkimellä ja pidetään se päällä kun kytkin vapautetaan. Painamalla nappia uudelleen LED:n pitäisi sammua. Käyttämällä *digitalRead()*-funktiota "kysytään" Arduinolta onko nappi painettuna vai ei. Ottamalla käyttöön *state*-muuttuja ohjelma muistaa, onko LED laitettava päälle vai pois päältä kun painokytkin painetaan uudelleen. Huomaat harjoituksen lopuksi että luonnosta tarvitsee parantaa, se tehdään harjoituksessa 3B.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 10 kΩ vastus
- LED-valo
- painonappikytkin

Tarvittavat funktiot/komennot:

- const int
- int
- int state
- void setup()
- pinMode ()
- void loop()
- digitalRead()
- if()
- state
- digitalWrite()
- else

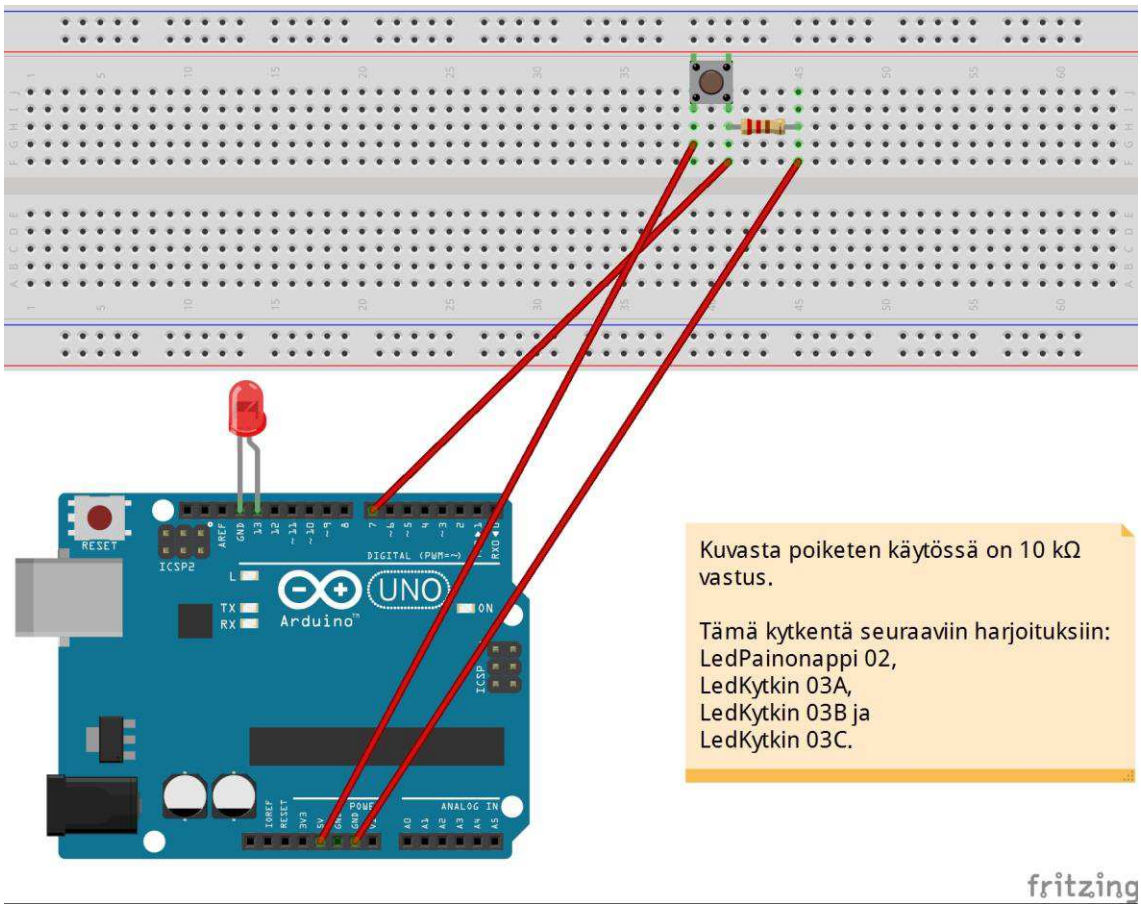
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 3B: Kytketään LED päälle ja pois osa 2/3

Tavoite:

Kytketään LED-valo painokytkimellä ja pidetään se päällä kun kytkin vapautetaan. Painamalla nappia uudelleen LED:n pitäisi sammua. Käyttämällä *digitalRead()*-funktiota "kysytään" Arduinolta onko nappi painettuna vai ei. Ottamalla käyttöön *state*-muuttuja ohjelma muistaa, onko LED laitettava päälle vai pois päältä kun painokytkin painetaan uudelleen. Lisätään loopin loppuun nykyisen arvon vertailu entiseen. Huomaat harjoituksen lopuksi että luonnosta tarvitsee vieläkin parantaa, se tehdään harjoituksessa 3C.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 10 kΩ vastus
- LED-valo
- painonappikytkin

Tarvittavat funktiot/komennot:

- const int
- int
- int state
- void setup()
- pinMode ()
- void loop()
- digitalRead()
- if()
- state
- digitalWrite()
- else

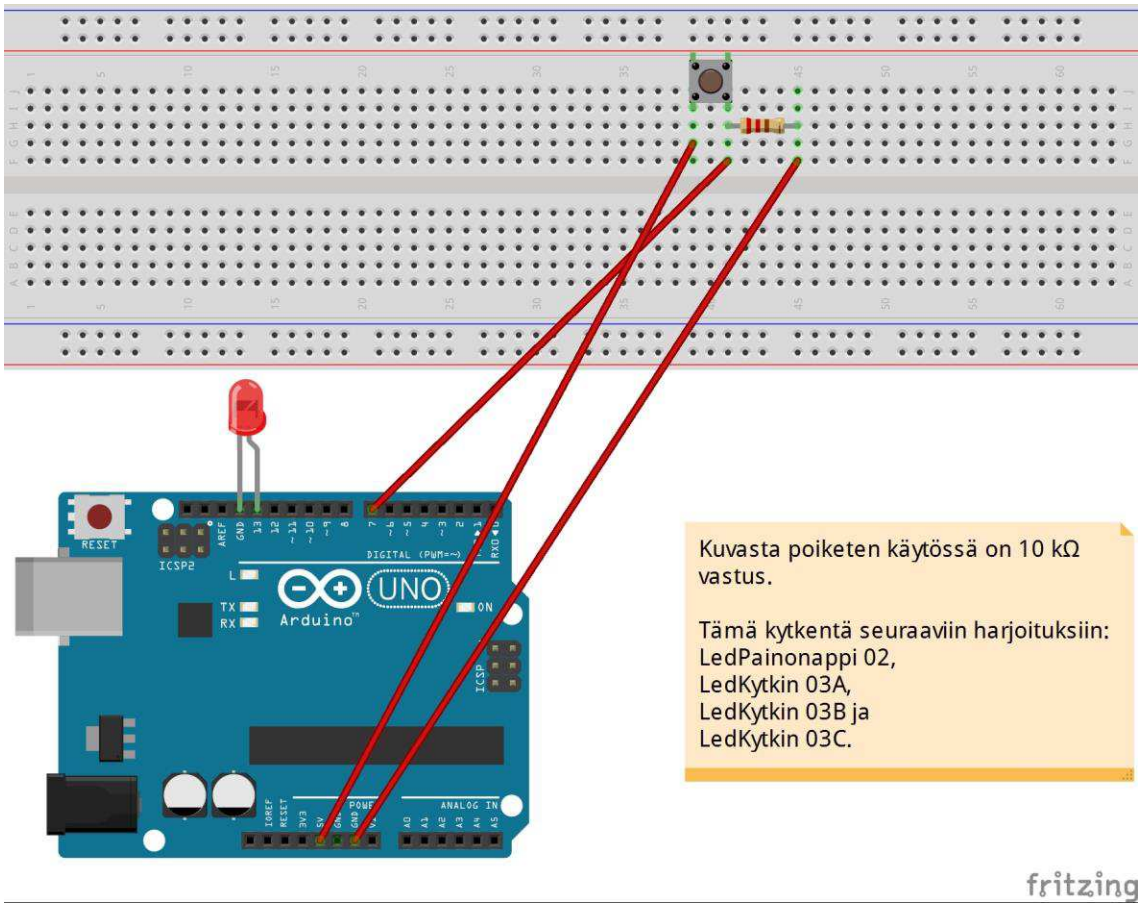
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 3C: Kytketään LED päälle ja pois osa 3/3

Tavoite:

Kytketään LED-valo painokytkimellä ja pidetään se päällä kun kytkin vapautetaan. Painamalla nappia uudelleen LED sammuu. Käyttämällä *digitalRead()*-funktiota "kysytään" Arduinolta onko nappi painettuna vai ei. Ottamalla käyttöön *state*-muuttuja ohjelma muistaa, onko LED laitettava päälle vai pois päältä kun painokytkin painetaan uudelleen. Lisätään loopin loppuun nykyisen arvon vertailu entiseen. Lisätään myös viivettä katkomisen (bouncing) estämiseksi.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 10 kΩ vastus
- LED-valo
- painonappikytkin

Tarvittavat funktiot/komennot:

- const int
- int
- int state
- void setup()
- pinMode ()
- void loop()
- digitalRead()
- if()
- state
- delay()
- digitalWrite()
- else

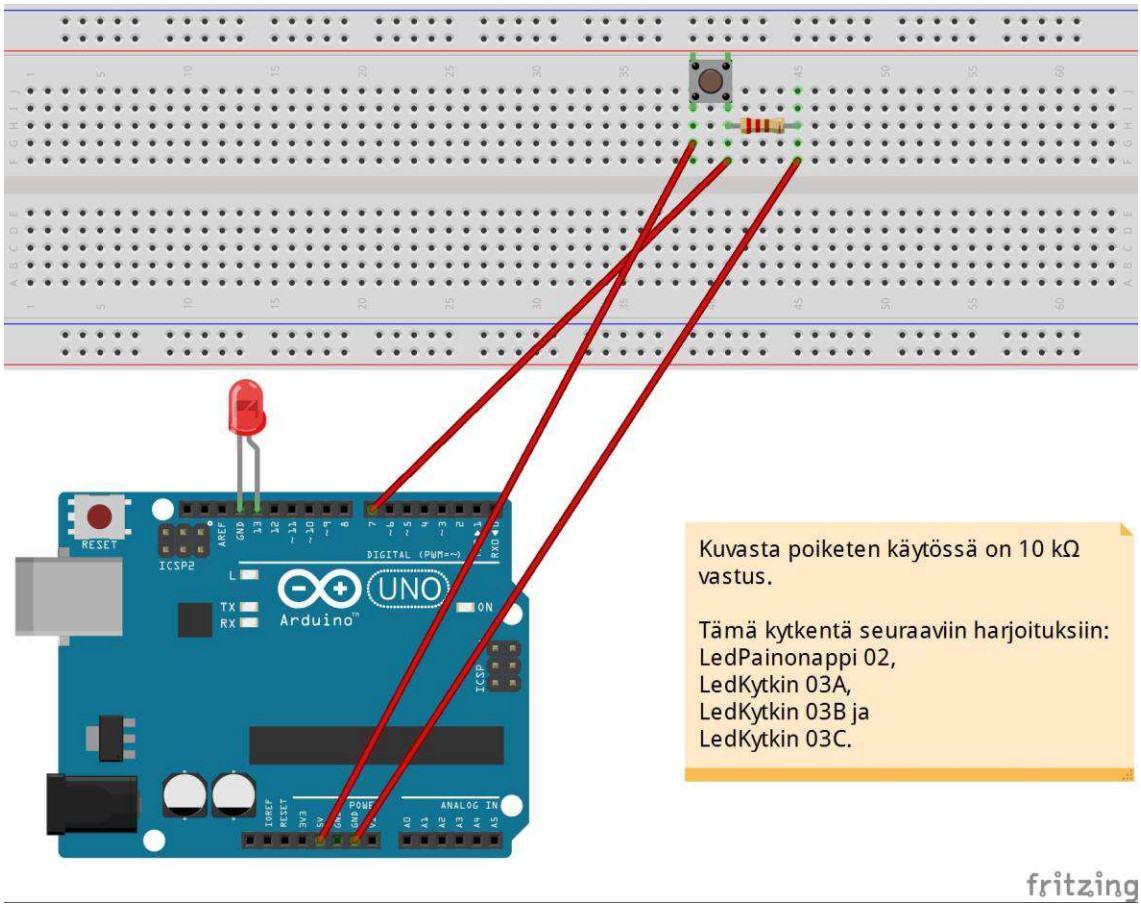
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 4: LED:n himmennys (PWM)

Tavoite:

Himmennetään LED-valoa kirkkaasta sammutettuun käyttäen pulssinleveysmodutaatiota (PWM, Pulse Width Modulation). Lisää luonnokseen hieman viivettä, jotta ehdit huomata muutoksen. Muistathan, että pulssinleveysmodulaation skaala on 0-255, jossa 255 tarkoittaa täyttä kirkkautta ja nollassa LED on pois päältä.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 270 Ω vastus
- LED-valo

Tarvittavat funktiot/komennot:

- const int
- int
- void setup()
- pinMode ()
- void loop()
- for()
- lisäysoperaattori ++
- analogWrite()
- delay()
- vähennysoperaattori --

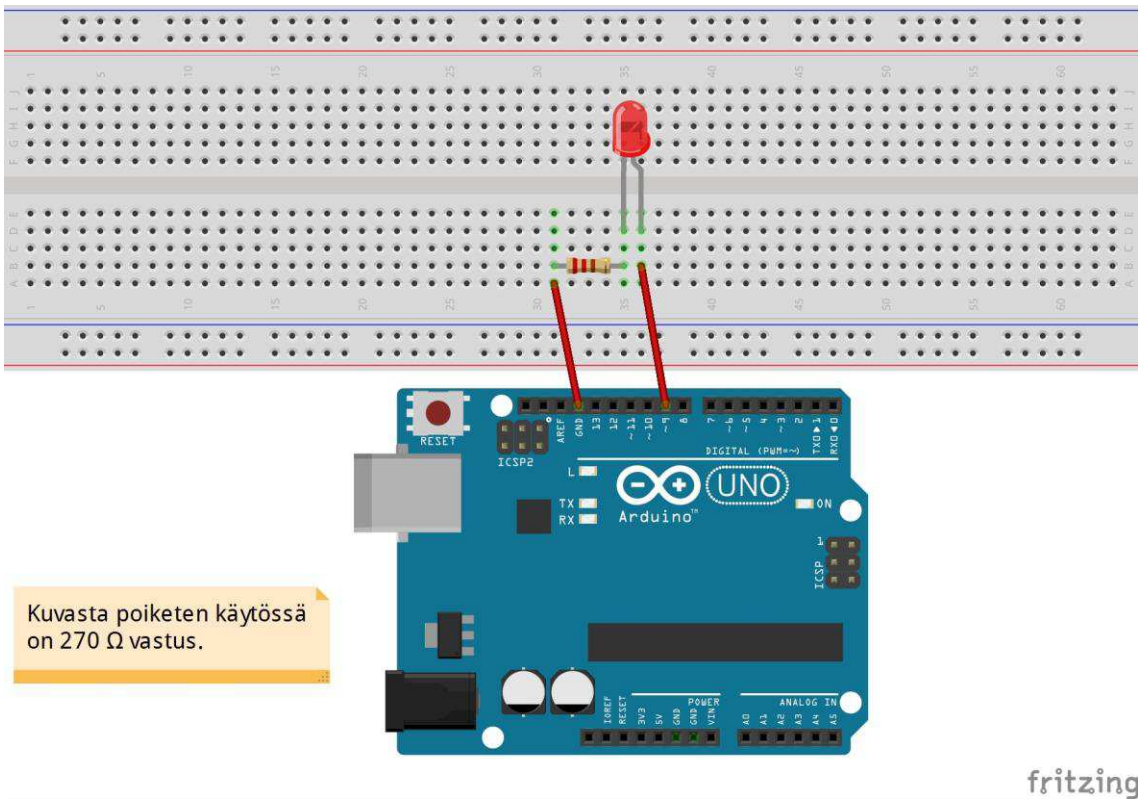
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 5: LED:n kirkkauden säätö (PWM)

Tavoite:

Sytytetään LED-valo kytkimellä ja pidetään se päällä kun kytkin vapautetaan. Jos kytkintä pidetään pohjassa, LED:n kirkkaus muuttuu käyttämällä pulssinleveysmodulaatiota (PWM, Pulse Width Modulation).
Harjoituksen pohjana kannattaa käyttää harjoituksen 3C (Kytketään LED päälle ja pois osa 3/3) luonnosta ja *millis()*-funktiolla selvittää kuinka kauan kytkintä pidetään painettuna. Harjoituksesta 4 (LED:n himmennys) on myös apua.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- yksi 10 kΩ vastus
- painonappikytkin
- yksi 270 Ω vastus
- LED-valo

Tarvittavat funktiot/komennot:

- const int
- int
- unsigned long
- void setup()
- pinMode ()
- void loop()
- digitalRead()
- if()
- JA-operaattori &&
- millis()
- lisäysoperaattori ++

- analogWrite()
- delay()
- else

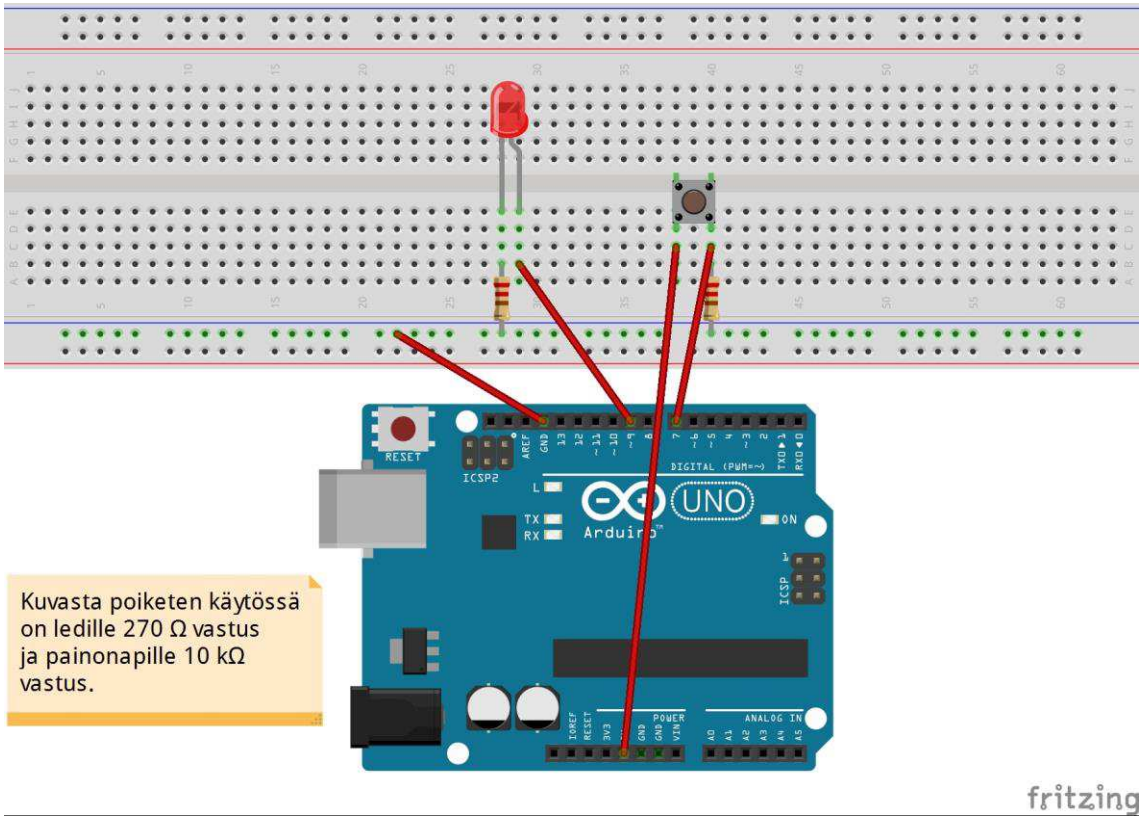
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 6A: LED vilkkuu valon määrän mukaan (analogRead)

Tavoite:

Vilkutetaan mikrokontrollerialustaan integroitua LED:iä (liitin 13) taajuudella, joka riippuu analogisen liitännän A0 antamasta arvosta. Vilkuttaminen toteutetaan sytyttämällä ja sammuttamalla LED sekä lisäämällä väleihin haluttu määrä viivettä. Analogisen tiedon lähteenä voidaan käyttää valoherkän vastuksen sijaan myös muuta analogista sensoria, esimerkiksi ultraäänianturia tai venymäliuskaa.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- analoginen sensori
- analogiselle sensorille tarvittava vastus

Tarvittavat funktiot/komennot:

- const int
- int
- void setup()
- pinMode ()
- void loop()
- analogRead()
- digitalWrite()
- delay()

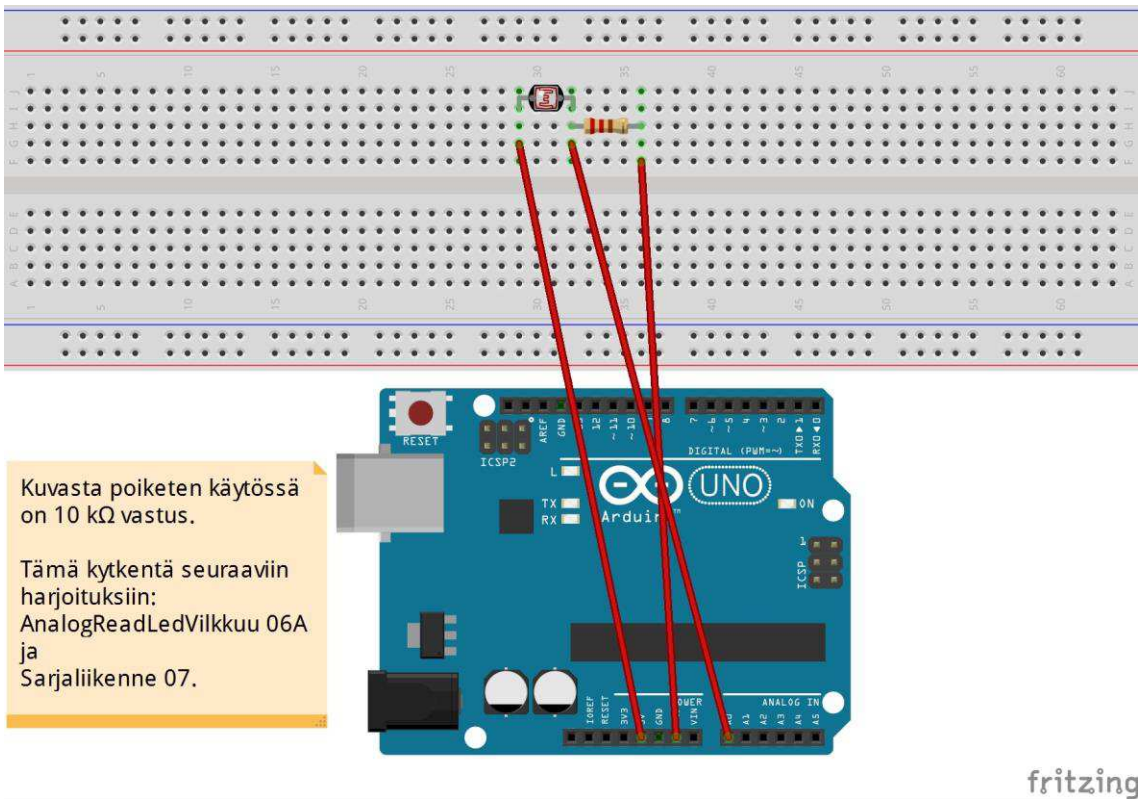
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 6B: RGB-LED:n värien kirkkauden säätö (analogWrite)

Tavoite:

Säädetään RGB-LED:n värien kirkkautta potentiometrin antaman analogisen tulon A0 mukaan. Analogisen tulon skaala (0-1023) jaetaan kolmelle värille (red, green, blue) siten, että jokainen väri muuttuu pulssinleveysmodulaation avulla himmeästä kirkkaaksi tai toisinpäin. Muistathan, että pulssinleveysmodulaation skaala on 0-255.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- potentiometri
- RGB-LED

Tarvittavat funktiot/komennot:

- const int
- int
- void setup()
- pinMode ()
- void loop()
- analogRead()
- if()
- vertailuoperaattorit <= ja >=
- float (liukulukutyyppi)
- analogWrite()
- else if()
- JA-operaattori &&
- else
- delay()

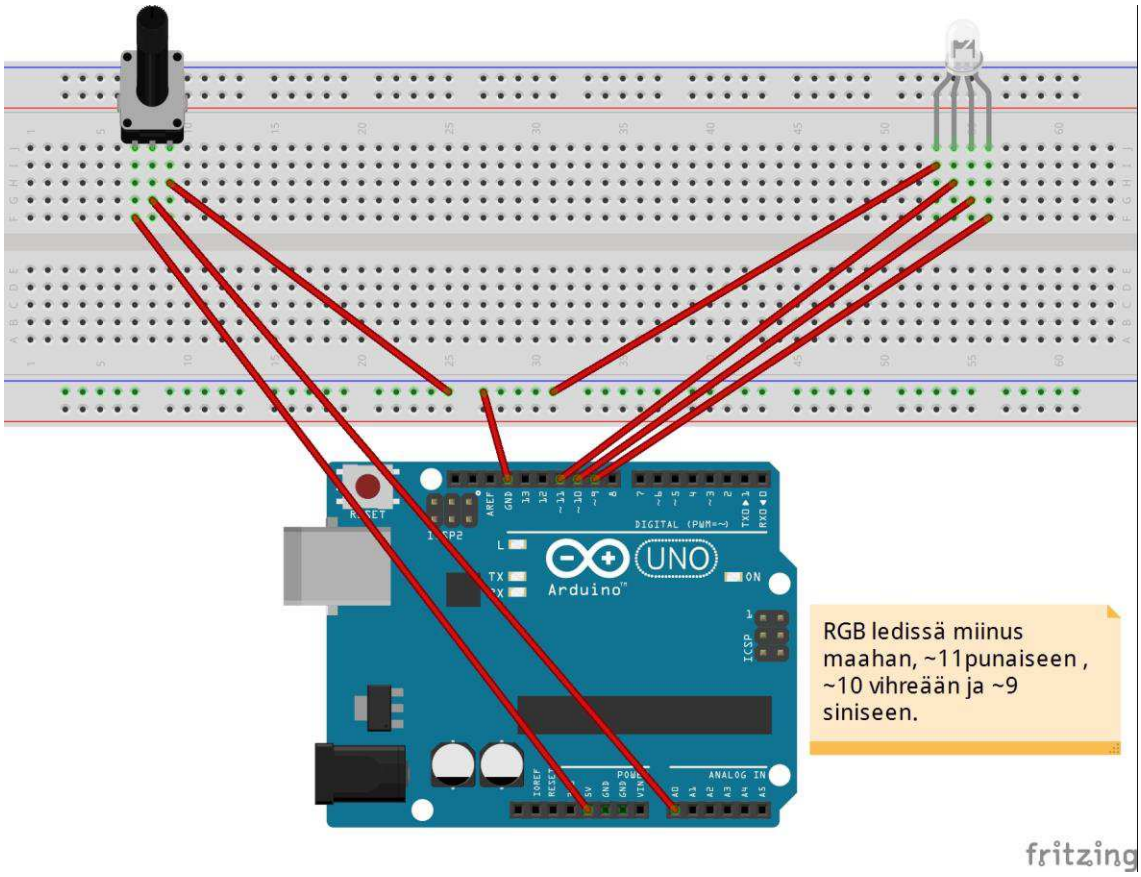
Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.
- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Harjoitus 7: Sarjaliikenne

Tavoite:

Lähetetään tietokoneelle arvot, jotka on luettu analogiselta input-liittimeltä A0. Analogisena sensorina voidaan valovastuksen sijaan käyttää myös muuta analogista sensoria, esimerkiksi ultraäänianturia tai venymäliuskaa. Varmista, että klikkaat "Serial Monitor" sen jälkeen kun olet ladannut ohjelman Arduinolle.

Tarvittavat komponentit:

- Arduino UNO
- USB type B - kaapeli
- Kytkentäalusta
- valmiiksi katkottuja hyppylankoja
- analoginen sensori
- analogiselle sensorille tarvittava vastus

Tarvittavat funktiot/komennot:

- const int
- int
- void setup()
- Serial.begin()
- void loop()
- analogRead()
- Serial.println()
- delay()

Funktioista ja komennoista lisää tietoa: http://arduino.cc/en/Reference/HomePage

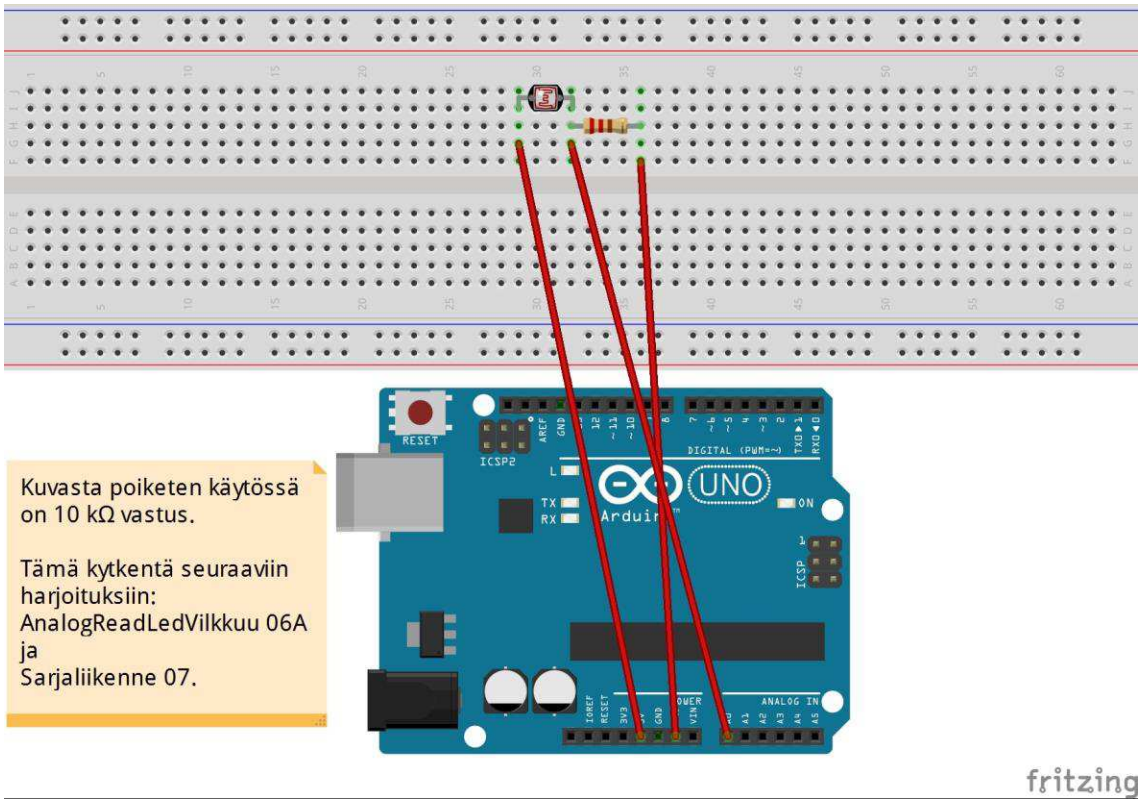
Muistathan:

- Koodin kirjoittaminen on tarkkaa. Muun muassa kirjaimien koot ja välilyönnit ovat merkityksellisiä. Kirjoita koodi juuri siinä muodossa, jossa se on ohjeistettu.

- Jokainen koodilohko rajataan koodissa omaksi nipukseen aaltosulkeilla {}
- // aloittaa kommenttirivin
- jokaisen toiminnallisen lausekkeen perään kirjoitetaan puolipiste ;

Kokeile luonnoksen muodostamista ja harjoituksen tekemistä ensin itse, vastaus on seuraavalla sivulla.

Kytkentäohje:



Liite 3. **Harjoitusten ratkaisut**

Seuraavaksi on esitetty oppimateriaaliin luotujen harjoitusten ratkaisut.

```
// Harjoitus 01 : Hello world (vilkkuva LED)

const int LED = 13; // LED yhdistettynä
                    //digitaaliseen liitinnastaan 13

void setup(){
  pinMode(LED, OUTPUT); // asettaa digitaalisen liitinnastan output-tilaan
}

void loop(){
  digitalWrite(LED, HIGH); // sytyttää ledin
  delay(1000);
  digitalWrite(LED, LOW); // odottaa sekunnin
  delay(1000);
}
```

```
// Harjoitus 02 : Pidetään LED päällä painonapilla

const int LED = 13;  // liitin LEDiä varten
const int BUTTON = 7;  // input-liitin, johon nappi on kytketty

int val = 0;  // val-muuttujaa käytetään input-liittimen tilan tallentamiseen

void setup(){
  pinMode(LED, OUTPUT);  // kerrotaan Arduinolle että LED on output-tilassa
  pinMode(BUTTON, INPUT);  // ja että BUTTON on input-tilassa
}

void loop(){
  val = digitalRead(BUTTON);  //luetaan input-arvo ja tallennetaan se
  // tutkitaan onko input HIGH
  // (pistetään kytkin päälle)

  if (val == HIGH){
    digitalWrite(LED, HIGH);  // sytytetään LED
  } else {
    digitalWrite(LED, LOW);
  }
}
```

```
// Harjoitus 3A: Kytketään LED päälle ja pois osa 1/3

const int LED = 13; // LED-liitin
const int BUTTON = 7; // kytkimen input-liitin
int val = 0; // alustetaan val tallentamaan input-liittimen tilatieto
int state = 0; // 0 = LED pois päältä, 1 = LED päällä

// int tarkoittaa että aiotaan tallentaa kokonaisluku, integer = kokonaisluku

void setup() {
  pinMode (LED, OUTPUT); // LED output tilaan
  pinMode (BUTTON, INPUT); // BUTTON input-tilaan
}

void loop() {
  val = digitalRead(BUTTON); // luetaan input-arvo ja tallennetaan se
  // tarkistetaan missä tilassa BUTTON on ja muutetaan sen tilaa
  if(val == HIGH) {
    state = 1 - state;
  }
  if (state == 1) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

```
/*
 Harjoitus 3B: Kytketään LED päälle ja pois osa 2/3
 Kytketään LED-valo painokytkimellä ja pidetään se päällä kun kytkin vapautetaan.
 Painamalla nappia uudelleen LED:n pitäisi sammua.
 Lisätty nykyisen arvon vertailu entiseen
  -> LED:n tila muuttuu vain silloin kun BUTTON saa arvon HIGH oltuaan ensin LOW.
*/

const int LED = 13; // LED-liitin
const int BUTTON = 7; // kytkimen input-liitin
int val = 0; // alustetaan val tallentamaan input-liittimen tilatieto
int old_val = 0 ; // val-muuttujan edellinen eli vanha arvo
int state = 0; // 0 = LED pois päältä, 1 = LED päällä

// btw: int tarkoittaa että aiotaan tallentaa kokonaisluku, integer = kokonaisluku

void setup() {
  pinMode (LED, OUTPUT); // LED output tilaan
  pinMode (BUTTON, INPUT); // BUTTON input-tilaan
}

void loop() {
  val = digitalRead(BUTTON); // luetaan input-arvo ja tallennetaan se
  // tarkistetaan onko jotain muuttunut kytkimen tilassa
  if((val == HIGH) && (old_val == LOW)) {
    state = 1 - state;
```

```
  }
  old_val = val; // val on nyt vanha, joten tallennetaan se

  if (state == 1) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

```
/*
 Harjoitus 3C: Kytketään LED päälle ja pois osa 3/3
 Kytketään LED-valo painokytkimellä ja pidetään se päällä kun kytkin vapautetaan.
 Painamalla nappia uudelleen LED sammuu.
 Lisätty nykyisen arvon vertailu entiseen
 -> LED:n tila muuttuu vain silloin kun BUTTON saa arvon HIGH oltuaan ensin LOW.
 Lisätty viivettä katkomisen (bouncing) estämiseksi.
*/

const int LED = 13; // LED-liitin
const int BUTTON = 7; // kytkimen input-liitin
int val = 0; // alustetaan val tallentamaan input-liittimen tilatieto
int old_val = 0 ; // val-muuttujan edellinen eli vanha arvo
int state = 0; // 0 = LED pois päältä, 1 = LED päällä

// int tarkoittaa että aiotaan tallentaa kokonaisluku, integer = kokonaisluku

void setup() {
  pinMode (LED, OUTPUT); // LED output tilaan
  pinMode (BUTTON, INPUT); // BUTTON input-tilaan
}

void loop() {
  val = digitalRead(BUTTON); // luetaan input-arvo ja tallennetaan se
  // tarkistetaan onko jotain muuttunut kytkimen tilassa
  if((val == HIGH) && (old_val == LOW)) {
```

```
    state = 1 - state;
    delay(30); // viive millisekunneissa
  }
  old_val = val; // val on nyt vanha, joten tallennetaan se

  if (state == 1) {
    digitalWrite(LED, HIGH);
  } else {
    digitalWrite(LED, LOW);
  }
}
```

```
/*
 Harjoitus 4: LED:n himmennys (PWM)
 Himmennetään LED-valoa kirkkaasta sammutettuun
 käyttäen pulssinleveysmodutaatiota (PWM, Pulse Width Modulation).
 Tässä harjoituksessa LED on kytkettävä Arduino Uno:n (versio R3) liitinnastoista
 yhteen seuraavista : 9, 10, 11. Saat itse valita mihin niistä.
 Tämä, koska niihin on rakennettu harjoituksen mahdollistava komponentti.
*/
const int LED = 9; // LED:n liitinnasta, vaihda jos joku muu
int i = 0; // käytämme tätä muuttujaa laskurina alas ja ylöspäin

void setup() {
  //pinMode(LED, OUTPUT); // LED output-tilaan KOODI TOIMII ILMAN TÄTÄKIN
}
void loop(){
  for (i = 0; i < 255; i +=5) { // edetään 0:sta 254:ään viiden välein
                                // eli kirkastetaan LED:iä
  analogWrite(LED, i); // asetetaan ledin kirkkaus
  delay(30); // viive millisekunneissa koska muuten ei näkisi muutosta paljain silmin
  }
  for (i = 255; i > 0; i -=5) { // edetään 255:sta 1:een eli himmennetään LED:iä
  analogWrite(LED, i); // asetetaan ledin kirkkaus
  delay(30); // viive millisekunneissa
  }
}
```

```
/*
 Harjoitus 5: LED:n kirkkauden säätö (PWM)
 Sytytetään LED-valo kytkimellä ja pidetään se päällä kun kytkin vapautetaan.
 Jos kytkintä pidetään pohjassa, LED:n kirkkaus muuttuu
 käyttämällä pulssinleveysmodulaatiota (PWM, Pulse Width Modulation).
 Tässä harjoituksessa LED on kytkettävä Arduino Uno:n (versio R3) liitinnastoista
 yhteen seuraavista : 9, 10, 11. Saat itse valita mihin niistä.
 Tämä, koska niihin on rakennettu harjoituksen mahdollistava komponentti.
 Mukana viivettä katkomisen (bouncing) estämiseksi.
 Harjoituksen pohjana kannattaa käyttää harjoituksen 3C (Kytketään LED päälle ja pois osa 3/3)
 luonnosta ja millis()-funktiolla selvittää kuinka kauan kytkintä pidetään painettuna.
*/

const int LED = 9; // LED-liitin
const int BUTTON = 7; // kytkimen input-liitin

int val = 0; // tallentaa input-liittimen tilatiedon
int old_val = 0 ; // tallentaa val-muuttujan edellisen eli vanhan arvon
int state = 0; // 0 = LED pois päältä, 1 = LED päällä
int brightness = 128; // tallennetaan kirkkausarvo, arvo voi olla väliltä 0-255, jossa
                      //255=täysi kirkkaus ja 0=LED pois päältä
unsigned long startTime = 0; // ajanhetki jolloin kytkimen painaminen alkaa

void setup() {
  pinMode (LED, OUTPUT); // LED output tilaan
  pinMode (BUTTON, INPUT); // BUTTON input-tilaan
```

```
}

void loop() {
  val = digitalRead(BUTTON); // luetaan input-arvo ja tallennetaan se
  // tarkistetaan onko jotain muuttunut kytkimen tilassa

  if ((val == HIGH) && (old_val == LOW)) {
    state = 1 - state; // muutetaan OFF-tila ON-tilaksi tai päinvastoin

    startTime = millis(); // millis() on Arduinon kello (tämä rivi muistaa milloin painokytkintä vii
    delay(10);
  }
  // tarkistetaan onko kytkintä pidetty painettuna
  if ((val == HIGH) && (old_val == HIGH)) {

    // jos on pidetty painettuna enemmän kuin 500 ms
    if (state == 1 && (millis() - startTime) > 500) {

      brightness++; // lisätään kirkkautta yhdellä
      delay(10); // pieni viive estää kirkkauden nousevan liian nopeasti

      if (brightness > 255) { // 255 on maksimikirkkaus
      brightness = 0; // jos ylitetään 255 -> palataan arvoon 0
      }
    }
  }
```

```
  old_val = val; // val on nyt vanha, joten tallennetaan se

  if (state == 1) {
    analogWrite(LED,brightness); // sytytetään LED nykyisellä kirkkaustasolla
  } else {
    analogWrite(LED, 0); // sammutetaan LED
  }
}
```

```
/*
 Harjoitus 6A: LED vilkkuu valon määrän mukaan (analogRead)
 Vilkutetaan mikrokotrollerialustaan integroitua LED:iä (liitin 13) taajuudella,
 joka riippuu analogisen liitännän A0 antamasta arvosta. Analogisen tiedon lähteenä
 voidaan käyttää myös muuta analogista sensoria, esimerkiksi
 ultraäänianturia tai venymäliuskaa.
*/
const int LED = 13; // LED:n liitin

int val = 0; // muuttuja, johon tallennetaan sensorilta saatava arvo

void setup() {
  pinMode(LED, OUTPUT); // LED output-tilaan
}

void loop() {
  val = analogRead(0); // luetaan arvo sensorilta, käytössä liitinnasta A0 (vaihda jos muu)
  digitalWrite(LED, HIGH); // sytytetään LED
  delay(val); // odotetaan val-arvon verran
  digitalWrite(LED, LOW); // sammutetaan LED
  delay(val);
}

/* Kokeile saatko LED:n vilkkumaan sitä nopeammin, mitä enemmän valaistusta on
(valoherkän sensorin läpi päästämä jännite kasvaa kun valon määrä kasvaa).
Onnistuu kun laitat arvoksi 1033 - val. val pitäisi olla välillä 0-1023 joten lyhyin
```

```
vilkutusväli pitäisi olla 10 ms ja pisin 1033 ms.
Harjoitukseen voisi lisätä painonapin kumman moodin valitsee, tulee if-funktiolla vaikkapa!
*/
```

```
  /*
  Harjoitus 6B: RGB-LED värin kirkkaus (analogWrite)
  Asetetaan RGB-LED:n värin kirkkaus analogisen liitännän A0 määrittelemän arvon mukaan.
*/

const int RED = 11; // Red:n liitin (muuta jos on eri)
const int GREEN = 10;
const int BLUE = 9;
//const int Sensori = A0; // Sensorin liitin, KOODI TOIMII ILMAN TÄTÄ LAUSEKETTAKIN

int val = 0; // muuttuja, johon tallennetaan sensorilta saatava arvo, alustetaan nollaan

void setup() {
  /*pinMode(RED, OUTPUT); // RED output-tilaan    ON NÄKÖJÄÄN TURHA LAUSEKE NÄMÄ KAIKKI MYÖS,
  pinMode(GREEN, OUTPUT);                         ARDUINO ON KEHITTYNYT.
 pinMode(BLUE, OUTPUT);*/
//  pinMode(Sensori, INPUT); // Sensori input-tilaan (Jos tarvii) EI NÄEMMÄ TARVI
}

void loop() {
  int val = analogRead(A0); // luetaan arvo sensorilta, käytössä liitinnasta A0

 if (val <= 340)
{
  float kirk1 = val*(255.0/340.0);
```

```
  analogWrite(RED, kirk1); // sytytetään LED sensorin määrittelemään kirkkausarvoon
  analogWrite(GREEN, 0);
  analogWrite(BLUE, 0);
}
else if (val >=341 && val <= 682)
{
  float kirk2 = (val-340.0)*(255.0/340.0); // vähennetään edelliset pois, alkaa nollasta joka väri
  analogWrite(RED, 0);
  analogWrite(GREEN, kirk2);
  analogWrite(BLUE, 0);
}
else
{
  float kirk3 = (val-683.0)*(255.0/340.0);
  analogWrite(RED, 0);
  analogWrite(GREEN, 0);
  analogWrite(BLUE, kirk3);
}
delay(10); // viive millisekunneissa jotta toimisi sulavammin
}


/*
1024/3 = 341 // skaala jaettuna kolmelle ledille
255/341 = 0,75
esim 320*0,75 = 240 (0-340)
```

```
VOIDAAN VAIHTOEHTOISESTI TOTEUTTAA MYÖS:
x*1/2*3/4 = (3*x)/8 (341-682)

(3*x)/16 (683-1023)

Seuraava variaatio tästä tehtävästä voisi olla vaikka toteuttaa kaikki sateenkaaren värit!
*/
```

```
/*
 Harjoitus 7: Sarjaliikenne
 Lähetetään tietokoneelle arvot, jotka on luettu
 analogiselta input-liittimeltä A0. Analogisena sensorina voidaan
 valovastuksen sijaan käyttää myös muuta analogista sensoria,
 esimerkiksi ultraäänianturia tai venymäliuskaa.
 Varmista, että klikkaat "Serial Monitor" sen
 jälkeen kun olet ladannut ohjelman Arduinolle.
*/

const int SENSOR = 0; // sensorivastuksen input-liitin, käytössä liitinnasta A0 (vaihda jos muu)

int val = 0; // muuttuja, johon sensorilta luettu arvo tallennetaan

void setup() {
  Serial.begin(9600); // avataan sarjaportti, jonka kautta voidaan lähettää
                      // tietoa tietokoneelle 9600 bittiä sekunnissa
}

void loop() {
  val = analogRead(SENSOR); // luetaan arvo sensorilta

  Serial.println(val); // tuupataan arvo sarjaporttiin
  delay(100); // viive millisekunneissa jotta sujuu mukavammin
}
```

```
// Kun olet ladannut ohjelman Arduinoon, klikkaa Serial Monitor -painiketta IDE:ssä
//(ikkunan oikea ylänurkka).
// Numerosarja pitäisi alkaa rullata ikkunan alaosassa.
```