Opinnäytetyö (AMK)

Esittävä taide

Nukketeatteri

2014

Tuomas Kotamaa

VAIHTOEHTOISTA TEATTERITEKNIIKKAA

– Arduino ja Raspberry Pi

TURUN AMMATTIKORKEAKOULU
TURKU UNIVERSITY OF APPLIED SCIENCES

OPINNÄYTETYÖ (AMK) | TIIVISTELMÄ

TURUN AMMATTIKORKEAKOULU

Esittävä taide | Nukketeatteri

2014 | Sivumäärä

Ohjaajat: Esa Pukero, Veli-Pekka Teponoja

Tuomas Kotamaa

VAIHTOEHTOISTA TEATTERITEKNIIKKAA

Tässä kirjallisessa opinnäytetyössä kuvataan yhden kokeellisen teatteriesityksen prosessia. Lähdimme syksyllä 2013 kokeilemaan millaisilla tavoilla voisimme muovata olemassa olevia teatteriteknisiä laitteistoja ja tuoda tekniikkaa näyttämölle osaksi lavastusta. Kokeilujen myötä rakensimme kevät talveksi 2014 esityksen ”Proto”, jossa käytimme kokeilujen pohjalta kolmea nykyaikaista automataa. Yksi automatoista oli nukketeatterissa opittua käyttäen rakennettu robotti, ja kaksi muuta oli vastuussa esityksen äänistä ja valoista. Kokeilujen myötä automatojen pohjiksi selviytyivät avoimet laitteistot Arduino ja Raspberry Pi. Käsittelemmekin tässä kattavasti mitä nämä laitteistot ovat. Tämä kirjallinen opinnäytetyö sivuaa myös avoimen laitteiston ideaa yleisesti pyrkiessään selvittää suuntaviivoja vaihtoehdoille teatterin tekniikan muovaamisessa. Kokeilujen perusteella näyttäisi siltä että Arduinon ja sulautettujen järjestelmien opetteluun tarkoitetuilla avoimilla laitteistoilla on monia käyttöjä teatterissä.

ASIASANAT:

Nukketeatteri, robotiikka, Arduino, Raspberry Pi, Open Source

BACHELOR´S THESIS | ABSTRACT

TURKU UNIVERSITY OF APPLIED SCIENCES

Performing Arts | Puppetry

2014 | Total number of pages

Instructors: Esa Pukero, Veli-Pekka Teponoja

Tuomas Kotamaa

ALTERNATIVE THEATER TECHNOLOGY

This written thesis describes the making of one experimental theater performance. We set out in the autumn of 2013 to try out in what ways we could shape existing theater technical equipment and technology to bring them onto stage into the set design. From the experiments, we built a winter spring 2014 presentation, "Proto", in which we used the experiments on the basis of three modern automatas. One of the automatas was a robot, built using techniques learned in puppetry, and the other two were responsible for the sounds of the performance, and lights. From the experiments we unraveled that Open Hardware Arduino and Raspberry Pi would be the platforms for the automatas. In the text we are going to handle comprehensively what these hardware are. This written thesis will also touch on the idea of open systems in general trying to resolve alternatives in shaping theater technology. As a result from the experiments it seem that Arduino and embedded system have many uses in theater.

KEYWORDS:

Puppetry, robotics, Arduino, Raspberry Pi, Open Source

SISÄLTÖ

LIITTEET

KUVAT

1 JOHDANTO

Lopputyöni taiteellisen osan idea syntyi, kun halusin löytää tavan tuoda teatterin teknologiaa johonkin yhteyteen näyttämöllä niin, että tuloksena olisi tahatonta teatteria ja sattumuksia. Idean perusteet muistuttivat muinaisista riitteistä, joissa käytettiin automataa, eli mekaanisia avuja ihmeiden esittämiseen. Automatalla tarkoitetaan mekanismia, joka suorittaa jonkin toimituksen. Esimerkiksi mekaaninen vieterilelukin on automata; se kummastuttaa ja viihdyttää. Muualla kuin uskonnollisten huijauksien yhteydessä automata-nimitystä käytetään ainakin ohjelmoinnin parissa. Myös ohjelmaa, joka suorittaa jonkin tehtävän nimitetään automataksi. Ohjelmat toimivat automaagisesti ja ovat mekanismeja, joten miksi ei? Kun olin vielä kuullut, että automata-nimitystä käytetään myös ohjelmoinnin parissa koin että se missä automata nykyisin esiintyy olisi kiinnostava kokeilu esitykselle.

Seuraava vaihe lopputyön suunnittelussa vei minut tutustumaan suomalaiseen kansaperinteeseen kummittelusta, kun yritin löytää riittejä läheltä niin paikallisesti kuin ajallisestikin. Yllätyin - tavallaan - siitä kuinka suomalaisessa kummitteluperinteessä kummittelu oli kaiken kansan nähtävyys. Tavallaan yllättymisellä tarkoitan sitä, että olin tietoinen monituhatvuotisesta perinteestä lavastaa ihmeitä yliluonnolliseen lankeavien kummasteltavaksi, mutta suomalainen suoruus tässä tahattomuuden kimarassa toi hymyn huulille. Ideoita kansanperinteestä ongittuani lähdin rakentamaan kummittelusta käsikirjoitusta ja tuomaan taruja nykyaikaan.

Open Source, tai avoimen lähdekoodin sovellus, on myös ohjelmoinnin parissa käytettyä termistöä ja tarkoittaa ohjelmaa, jonka alkuperäistä koodia ei ole kätketty tai muulla tavoin tehty ihmiselle mahdottomaksi lukea. Termiä on lavennettu sittemmin koskemaan myös laitteistoja ja montaa muutakin asiaa, kuten joogaa. Open Hardware, eli avoin laitteisto, tarkoittaa laitetta, jonka toimintaa, tai toimintaperiaatetta ei ole kätketty. Käytännössä tämä tarkoittaa että laitteen piirikaavio ja sen toimintaan mahdollisesti liittyvät ohjelmat ovat vapaasti tarkasteltavissa, ja niin hyvin dokumentoitu että niitä pystyy ymmärtämään asiaan perin-

pohjaisesti vihkiytymättä. Laitteistot saa itse myös rakentaa, niitä ainoastaan ei saa silloin markkinoida laitteiston alkuperäisellä nimellä. (Arduino 2014a)

Open Source on myös filosofia, jota ajavat monet avoimuuden eettisyyttä korostavat järjestöt, kuten Open Source Iniative, jonka huolena on Open Sourcen käyttäminen brändinä. (Open Source 2014) Kun avoimuuteen liitetään vielä vapaus, päästään matkaan idean alkujuuria kohti. Ei ihan alkujuurille, mutta muutama vuosikymmen taaksepäin aikaan, jolloin avoimuutta alettiin kaivata ohjelmointiin liittyvissä tehtävissä. Vapaan ja avoimen lähdekoodin ohjelmista käytetään myös nimitystä FOSS, eli Free and Open Source Software. Eettisyyden sijaan tässä kyse on siitä, ettei pyörää kannata keksiä aina uudestaan. Eli kun jokin ohjelma on jo kertaalleen kirjoitettu, ei sen kirjoittamisen prosessin kertaamiseen ole syytä. Vapaan ja avoimen lähdekoodin ohjelmiston koodin on tarkoitus olla ihmisen luettavissa että sitä saa muokata ja yleensä myös levittää vapaasti, usein parhaaksi katsomallaan tavalla. Open Sourcea ja FOSS:ia nimitetään niiden soveltamien vaihtoehtoisten tekijänoikeuskäytäntöjen vuoksi myös Copyleftiksi. (GNU 2013a)

Open Sourcen avointen laitteistojen ja avoimen lähdekoodin sovellusten avulla rakensimme esityksen “Proto”, joka esitettiin demona Turun AMK:n Taideakatemiassa Naruteatterissa helmikuussa 2014. Esityksen rakentaminen alkoi syyskuussa 2013 käsikirjoituksen suunnittelemisella. Esitysidean alkukehittelyn jälkeen seurasivat tutustuminen teatterin tekniikan mahdollisuuksiin esitysidean puitteissa ja ideat kokeiluille Open Source:n ja robotiikan parissa. Nuken rakentaminen ja liittäminen esityksen osaksi seurasi käsikirjoituksen valmistumista syksyllä, kun idea nuken käyttötavasta suhteessa esitysideaan alkoi hahmottua. Valo- ja äänisuunnitelmista ääni alkoi rakentua jo syksyllä ja oli ensimmäinen työryhmää rakentava voima. Valosuunnitelma taas alkoi rakentua, kun laitteistot valoajoihin alkoivat olla kokeilukunnossa marras- joulukuun vaihteessa. Valosuunnitelman rakentamisen yhteydessä aloitettiin myös esityksen harjoittelu.

2 KÄSIKIRJOITUS

Tarinan ideasta aloin hahmottelemaan dramaturgiaa ja ohjaussuunnitelmaa. Tarinan syntymistä valaiseva kynttilä paloi heti alusta alkaen kahdesta päästä, kun tarinaan piti sopia erikoisten teknisten ratkaisujen näkyminen, ja tarinan piti myös tuottaa tekniikalle ongelmia ratkaistavaksi. Pohdinnan tuloksena alkoi hahmottua tarina kummitustutkijasta ja hänen näennäisen tieteellisistä tutkimuslaitteistaan. Pohdinnan tulos vaikutti kunnolliselta, koska tarinassa kummitustutkijasta yhdistyivät teknisyys ja teatterillisuus. Itseään toteuttavan kummitustutkijan repertuaariin huijarina kuuluu teknisten laitteiden parissa päteminen ja show-henkisyys yliluonnollisen ilmapiirin luomiseksi näennäisen tieteellisyyden nimissä.

Käsikirjoituksen kirjoittaminen alkoi syksyllä, kun osallistuin näytelmän kirjoittamiseen tähtäävälle dramaturgian kurssille. Löysin kummittelusta kertovasta kansanperinteestä useita hienoja tarinoita, joista yksi oli ylitse muiden viimeaikaisuutensa ja kuuluisuutensa vuoksi. Siitä alkoi kehittyä idea tutkimuskohteesta, joka on aivan ilmeinen, mutta joka ei tottele kummitustutkijan tutkimuksia. Kummitustutkijasta alkoi kehittyä kunnianhimonsa riivaama huijari, joka aloitettuaan ei saata luovuttaa maineensa menettämisen pelosta. Hänestä alkoi kehittyä myös vallanhimoinen, kun kummitusten tutkimiseen liittyvästä taikauskosta oli tullut hänelle valuuttaa.

Tarina kertoo viimeisestä yrityksestä ennen luovuttamista. Kummitustutkija on käyttänyt jo omat resurssinsa kokeillessaan saada kummituksia esiin niin teknisten kuin hengellistenkin apujen välityksellä. Tarina näyttää kummitustutkijan viimeisessä tutkimuksessaan, joka on kuten aiemmatkin, mutta jossa tutkija on jo epätoivoinen ja valmis mihin vain. Kummitustutkija kokeilee ensin tekniset avut, kuten ennenkin, ja sitten suorittaa rituaalin, jonka tarkoituksena on saada kummitus nukessa esiin, kuten ennenkin. Lopulta, kun mikään ei tälläkään kertaa ollut onnistunut, kummitustutkija tarjoaa kummitukselle majapaikaksi omaa kehoaan ja myy näin näennäisesti sielunsa tarkoituksenaan saada pois pääsy tutkimuksesta mainettaan menettämättä.

Käsikirjoitus alkoi omana tekstinäni mutta kehittyi kahta eri kautta tätä näytelmää varten ryhmätyönä tehdyksi. Minun osakseni muodostui ryhmätekstin tekemisessä kirjata ideat muistiin ja tehdä niistä kokonaisuus. Ensimmäinen poikkeama omasta tekstistä tuli, kun pyysin tuntemaani räppäriä osallistumaan esityksen musiikin tekemiseen. Nukketeatterissa ei ainakaan minun opiskeluaikanani ollut kuultu räppiä, joten räppi tuntui kiinnostavalta musiikin genreltä jo muutenkin kokeelliseen esitykseen. Räpissä sanoitukset tosin ovat niin suuressa osassa, että niistä tuli heti osa käsikirjoituksen tarinankerrontaa ja käsikirjoituksesta ryhmätyö. Toinen poikkeama käsikirjoituksessa tuli robotiikan tultua mukaan esityksen osaksi. Robotin käytöksen esittäminen koodina käsikirjoituksessa tuli oleelliseksi, kun haluttiin että robotin toiminta on ymmärrettävässä yhteydessä esityksen dramaturgian kanssa. Ja koska en koodannut robottia yksin, vaan teimme robotin koodia ryhmätyönä, oli toinenkin syy kutsua käsikirjoitusta ryhmätekstiksi kehittynyt. Robotin koodin kommentoidulla esittämisellä käsikirjoituksen yhteydessä hyödyimme sen, että saatoimme käsikirjoituksen esittämällä selvittää miten robotti toimii esittelemättä itse robottia toimimassa näyttämöllä. Robotin koodin liittäminen käsikirjoituksen osaksi oli oma mielenkiintoinen prosessinsa ja kokeilimme monia eri tapoja esittää koodi käsikirjoituksessa. Kun koodin esittämiselle tietyssä kohdin käsikirjoitusta ei robotin toiminnan kannalta ollut merkitystä, lopulta parhaaksi tavaksi paljastui tehdä koodista erillinen liite käsikirjoituksen loppuun. (Liite 1)

3 KUMMITUSTUTKIJAN LAITTEISTO

Kummitustutkimuslaitteiden etsiminen alkoi tutustumalla mahdollisuuksiin muokata tavallisia teatteriohjaimia kummitustutkijan käyttöön. Lähtökohtana idea oli hyvä mutta ongelmaksi muodostui ohjainten mahdollinen rikkoutuminen niitä muokatessa. Paremmalta idealta tuntui siirtää tarinan tapahtumat huomisen kaltaiseen lähitulevaisuuteen ja tehdä laitteet alusta asti itse. Itse tekemisestä alkoi hahmottua idea Arduinon käyttämisestä tutkimuslaitteiston sydämenä, sillä prototyypittelyyn tarkoitettuna laitteistona Arduinon mahdollisuuksia rajasi vain mielikuvitus. Arduinolla olisi voinut helposti vaikka rakentaa valo- ja ääniajot itse tehdyillä ohjaimilla. Tai automoida koko tekniikan, miten me lopulta teimmekin tuodessamme automatat nykyaikaan.

Kun päätös Arduinon käyttämisestä oli tehty ja samoin joitain luonnoksia siitä mitä sillä voisi kokeilla, osallistuin kurssille, jolla perehdyimme Arduinoon pohjautuvaan robotiikkaan. Robotiikan kurssilla tutustuimme robotin rakentamiseen ja ohjelmoimiseen Arduino Unolla ja Parallaxin Boe-Bot rakennussarjalla. Robotin sydämenä toimi Arduino Uno Parallaxin Motor Shieldin kanssa ja kurssin aikana perehdyimme siihen, miten Boe-Botin osista saa rakennettua ja ohjelmoitua yksinkertaisen ja pienen, kahdella pyörällä kulkevan, ledejä vilkuttavan ja infrapunalla ympäristöään aistivan robotin. (vrt. Parallax 2014) Opintojeni aikana ei nukketeatterissa ollut nähty robotiikkaakaan, ja kurssin aikana alkoi tuntua kiinnostavalta että esityksen osana olisi robotiikkaakin, vaikka aluksi tarkoitus oli ollut vain tutustua Arduinon mahdollisuuksiin. Tarinan puitteissa näytti kuitenkin kiinnostavalta että robotista sai esitykseen itsenäisesti toimivan tutkimuslaiteen, joka etsi näyttämöltä merkkejä kummittelusta. Boe-Bot tosin on hyvin pienitehoinen, eikä soveltunut esitykseen pienuutensa vuoksi, joten käytin osan kurssista etsien osat suurempitehoiseen robottiin. Boe-Botin kantavuus ei olisi riittänyt paljon siihen kiinnitettyjä ledejä enempään mutta etsimällä toiset osat kantavuuden sai riittäväksi ja robotista niin tehokkaan että se pystyi selviytymään jo pienistä esteistäkin. Boe-Botissa käytetty infrapunasensori ei myöskään olisi toiminut teatterissa. Infrapunasensori toimii valon ja lämmön taajuuk-

silla, eikä olisi toiminut teatterin vaihtuvissa valotilanteissa. Sensoriin olisi tullut valon ja lämpötilan vaihteluista häiriöitä, eikä robotti olisi pystynyt havainnoimaan ympäristöään. Ultraääni selvisi käytännöllisimmäksi vaihtoehdoksi. Ultraääni on korkeaa ääntä, jota ihminen ei kuule, eikä sitä esiinny teatterissa. Ultraäänisensori mittaa etäisyyttä äänen heijastuksina, eli kaikuluotaamalla. Ainoaksi esteeksi ultraäänisensorin käyttämiselle teatterissa jäivät voimakkaasti ääntä eristävät materiaalit, joita näyttämöllä saattaisi olla. Kohdistuessaan voimakkaasti ääntä eristävään materiaaliin ultraääni vaimentuu, eikä heijastu takaisin sensorille, jolloin sensori ei toimi. Me emme kuitenkaan löytäneet teatterista niin ääntä eristäviä materiaaleja, että niistä olisi ollut haittaa sensorille.

Päätös Arduinon käyttämisestä tekniikka-ajoissa DMX järjestelmän osana syntyi robotin osia etsiessä. Osia etsiessä selvisi, että Arduinolla on melko yksinkertaista ohjata DMX-pohjaista himmennintä. Arduinon vanhalla mallilla, Duomillanovella, DMX-ohjaukseen tarvittavat kolme johtoa olisi saattanut liittää suoraan Arduinon pinniliittimiin, mutta uudemmalla Arduino Unolla väliin tarvitsi DMX Shieldin. Idea lähti kehittymään edelleen ja kummitustutkijalle alkoi hahmottua konsoli, jolla hän hallitsisi ympäristöään ja tekisi hämmentäviä analyysejään. Konsolin muiksi osiksi Arduino Unon ja DMX Shieldin lisäksi tuli pieni tietokone, Raspberry Pi, ja kosketusnäyttö. Tietokoneen liittämisellä konsoliin hyödyimme sen, että kummitustutkija saattoi ohjata myös äänet konsoliltaan. Osien tilaaminen alkoi robotiikan kurssilla mutta ne saapuivat vasta nykykuvanveiston jatkokurssilla jota robotiikka oli edeltänyt. Kokonaisuudessaan laitteistojen suunnittelemisesta tilattujen osien saapumiseen kesti pari kuukautta. Tilaukset olisivat tulleet muuna vuodenaikana nopeammin, mutta koska pääsimme tekemään tilauksen vasta joulukuun alussa tuli osa osista joulun vuoksi vasta loppiaisen jälkeisellä viikolla. Nykykuvanveiston jatkokurssin aikana robotti rakennettiin ja testattiin, sekä konsoli laitettiin kasaan ja sen toimintaa alettiin suunnitella.

Idea alussa oli että pyrkisimme käyttämään yhteisöllisesti tuotettuja laitteistoja ja avoimen lähdekoodin sovelluksia niin pitkälle kuin pystyisimme. Pian jouduimme kuitenkin pohtimaan missä raja kulkee. Robotin rakentamisessa Ar-

duinolla ei ollut mitään epäselvää mutta konsolin rakentaminen ei ollut niin yksinkertaista. Arduino ja sen Motor Shield olivat kummatkin avoimia laitteistoja ja kaikki muut robotin osat olivat vain komponentteja. Raspberry Pi:kin oli avoin laitteisto mutta siihen ei ollut muita osia avoimina laitteistoina. Kaikki muut konsolin osat, kuten kosketusnäyttö ja kaikki kaapelit pitikin hankkia suurilta, kaupallisilta valmistajilta emmekä siksi saaneet vakuuksia niiden alkuperästä tai dokumentaatioita niiden toimintaperiaatteista.

Robotti

Robotin rakentaminen siihen kuntoon että sen liikkumista ja valoja saattoi kokeilla, kävi käden käänteessä, mutta robotin saaminen toimimaan yhdellä virtalähteellä virheettömästi vaati yllättäen ongelmanratkaisua. Robotti oli ensin rakennettu niin että sen kummallakin laitteistolla, Arduinolla ja Motor Shieldillä, oli omat virtalähteensä ja moottoreiden virrankulutus niiden käynnistyessä jäi huomattavaksi kokeilun ja erehdyksen kautta. Huomasimme, että käytettäessä yhtä virtalähdettä moottorit käynnistyessään aiheuttivat häiriön piiriin ja Arduino jäi ilman virtaa. Ratkaisu tähän löytyi moottoreille rakennetuista häiriönpoistopiireistä, jotka varautuivat ennen moottorien käynnistymistä ja pitivät näin huolen siitä että moottorien käynnistyessä kaikki laitteet saivat tasaisesti virtaa. Häiriönpoistopiirit rakennettiin kolmesta kapasitorista moottoria kohtia kolvamalla ne sarjaan ja kunkin moottorin napoihin. Käynnistämisen yhteydessä varautuvat kapasitorit huolehtivat moottorien käynnistämisen aiheuttamasta häiriöstä piirissä antamalla laitteistolle hetkellisesti ylimääräistä virtaa. (vrt. Adafruit 2012)

Kun robotti oli saatu toimimaan niillä osilla millä se oli suunniteltu, eli yhdellä virtalähteellä, ultraäänikuuloineen, rgb-ledeineen ja nelivetoisesti, saattoi sen käytöksen ohjelmoiminen alkaa. Tehdessämme päätöksiä robotin luonteesta päätimme, että se vilkuttaisi iloisesti valojaan liikkuessaan eteenpäin ja voisi valita useasta eri vaihtoehdosta mitä tehdä kun sillä olisi este eteenpäin liikkumiselle. Robotti aisti ympäristöään yhdellä ultraäänisensorilla keulastaan, eli se kuulee eteenpäin ja osaa toimia kuulemansa perusteella. Kuullessaan edes-

sään esteen se käyttää hetken harkitsemiseen ja vaihtaa sitten suuntaa. Päätimme ohjelmoida robotin harkitsemaan ettei se olisi kokoajan aktiivinen ja antaisi tilaa muulle esitykselle. Ohjelmoimalla robottiin satunnaismuuttujia ja viiveitä saimme robotin käytöksen lähemmäksi nukketeatterillista animointia ja robotista älykkäämmän oloisen kuin mitä se oikeastaan edes oli.

Robotin käytöksen suunnittelemisessa ja robotin liittämisessä esityksen osaksi saimme kokeilla täysin vapaasti, sillä robotiikkaa on käytetty näyttämöllä vielä hyvin vähän. Näyttämöteknisiä laitteistoja on motoroitu tavalla tai toisella jo satoja vuosia, tuhansiakin, mutta kun puhutaan siitä että robotti olisi esillä näyttämöllä, esimerkkejä ei juuri ole. Yksi tuore ja robotiikan käyttämistä hyvin äärimmäisyyksiin vienyt esimerkki löytyi vuodelta 2010. Peter Torpeyn maisterintutkintotyö MIT:stä kertoo tarinan Simon Powersista, joka väistääkseen kuolemaa siirtää tietoisuutensa koneisiin. Esityksen pääosassa ovat näyttämöllä toimivat robotit, visuaaliset elementit ja ääni. Esitys pohjautui Tod Machoverin oopperaan “Death and the Powers”. (Torpey, 2009, 52) Esityksen alussa päähenkilö Simon Powers joutuu koneisiin ja koko esityksen ajan häntä esitetään koneellisesti. Näyttelijä on näyttämön ulkopuolella ja esiintyy koneiden välityksellä. Näyttelijälle on mahdollistettu havaitseminen sensoreita käyttämällä, ja hän kuulee ja näkee audio-visuaalisten laitteiden avulla. Näyttelijänä hän on vain koneiden välityksellä läsnä näyttämöllä mutta tarkoituksenaan silti välittää kanssanäyttelijöilleen ja yleisölle oma osansa. Hän voi jäljitellä koneiden välityksellä sitä millainen hänen läsnäolonsa olisi näyttämöllä kuin hän olisi paikalla fyysisesti. (Torpey, 2009, 55)

Teatteri ja Copyleft

Kuten johdannossa jo mainitsin, Open Source on oikeastaan yhden eettisen filosofian brändi, jolla voidaan määritellä esimerkiksi kehityssuunniltaan ja lisensoinniltaan vapaata tai avointa ideaa. Idea voi olla mitä tahansa: softaa, piirilevy, tietokone, helikopteri, tai teatteriryhmän toimintaperiaate. Esimerkiksi Gnu, Linux, Arduino, Raspberry Pi, ja monet muut ideat ovat kaikkien vapaasti jatko-

kehiteltävissä ja käytettävissä. Niistä käytetään nimitystä Open Source, koska se kuvaa ymmärrettävästi sitä mistä niissä on kyse. Open Source ohjelmistoista käytetään nimitystä FOSS, kun halutaan korostaa sitä että avoin lähdekoodi on sekä ilmainen että kunnioittaa käyttäjänsä oikeuksia. (Open Source 2014) FOSS käsittää vain ohjelmistoja, ja Open Source vain vähän FOSS:ia laajemmin tekniikkaa ja ideoita. Copyleftistä puhuttaessa taas käsitetään koko aineettoman pääoman alue. Copyleftin tarkoitus onkin ilmaista, että kyse on niin-ja-niin vapaasta ja avoimesta asiasta, johon on sovellettu niitä-ja-niitä tekijänoikeuksia. (GNU 2013b)

Arduino on elektroniikan nikkaroinnin ja ohjelmoinnin prototyypittelytyökalu ja avoin laitteisto. Yksinkertaisuudessaan se on vain lelu elektroniikasta kiinnostuneille, mutta helpottaa monien tehtävien tekemistä, jotka jäisivät tekemättä siksi että niiden aloittamisen kynnys on kovasti korkea. Arduinosta ja muista Arduinon kaltaisista työkaluista käytetään myös nimitystä “Maker Board”, sillä ne tekevät tekemisestä helpompaa. Erityisesti kokeilevissa ja luovissa tehtävissä Arduinon yksinkertaisuudesta ja helppokäyttöisyydestä on paljon iloa. Arduinon perusmalli on pieni, noin tulitikkuaskin kokoinen laite. (Kuva 1.) Laite liitetään tietokoneeseen USB:llä ja käynnistetään sen omalla ohjelmalla tietokoneelta. Ohjelma sisältää Arduinon oman integroidun kehitysympäristön ja valmiita esimerkkejä siitä mitä Arduinolla voi tehdä. Arduinon sydän on sen mikrokontrolleri, jota ohjelmoimalla Arduinon saa suoriutumaan monista elektroniikan prototyypittelylle yleisistä tehtävistä ja monista muista monimutkaisemmistakin tehtävistä. Pienivirtaisia projekteja voi rakentaa Arduinolla sellaisenaan ja se ottaa virran niille USB:stä, suurempivirtaisissa Arduinoon voi liittää virtalähteen, jolloin virtaa riittää suurenkin kulutukseen. Arduinoon onkin liitettävissä mielikuvituksekas määrä lisälaitteita, jotka mahdollistavat myös esimerkiksi langattoman viestinnän radiotaajuuksilla tai verkon käyttämisen.

Raspberry Pi on pieni tietokone ja sekin avoin laitteisto. (Kuva 2.) Pienuudestaan ja markkinoiden edullisimmasta hinnastaan huolimatta Raspberry Pi on melko tehokas ja monipuolinen. Sen prosessorina on 700MHz Arm-prosessori ja siinä myös on tehokas grafiikkaprosessori. Lisäksi sen tulitikkuaskia vähän

suurempalle piirilevylle mahtuu kaksi USB-paikkaa, SD-kortti paikka, äänikortti, verkkokortti ja HDMI-, sekä komposiittivideo-liitännät. Rapsperry Pi:n käyttöjärjestelmänä käytetään esimerkiksi Gnu/Linuxia ja yksi Raspberry Pi:n yleisimmistä käyttötarkoituksista tietokonenikkaroinnin lisäksi on toimia multimediapalvelimena.

Avoimen laitteiston robotiikka taas ei varsinaisesti ole kiteytettävissä yhteen asiaan, vaan kyse on ennemminkin tavasta tehdä robotiikkaa yhteisöllisesti. Laitteistoina robotiikkaa voi ostaa valmiina rakennussarjoina, jotka on tehty avoimia laitteistoja, kuten Arduinoa tai Raspberry Pi:tä silmällä pitäen. Yhteisöillä on avoimen laitteiston robotiikkaa varten neuvovia ja projekteja dokumentoivia foorumeita ja yhteisöllisesti tehtyjä ohjelmistoja, jotka ovat kaikkien käytettävissä ja muokattavissa. Robotiikka ei kuitenkaan välttämättä vaadi erottelua suljetun ja avoimen akselilla, koska robotteja tehdään pääosin elektronisista komponenteista, joiden toimintaperiaatteen julkisuus on välttämätöntä niiden oikein käyttämiselle, eikä komponenteista rakennettujen robottien toiminta välttämättä vaadi ohjelmistoja. Ohjelmistojen osa tulee mukaan oikeastaan vasta kun robotti on suunniteltu niin että sen laitteistojen toiminta vaatii ajureita, tai kun robottiin halutaan tekoäly, tai sen halutaan jäljittelevän ohjelmoinniltaan elämää, jolloin puhutaan tekoelämästä.

4 ARDUINO

Arduinon sydän on sen ohjelmoitava mikrokontrolleri. Arduinoita on useita eri malleja, ja eri malleihin on lukematon määrä erilaisia lisälaitteita lukemattomiin eri tarkoituksiin, lukemattomia eri tehtäviä helpottamaan. Arduinon perusmallista, Arduino Unosta, löytyy mikrokontrollerin lisäksi USB-liitäntä tietokoneeseen yhdistämistä varten ja pinniliitäntöjä, joilla prototyypittelyyn rakennetut piirit yhdistetään Arduinoon. Muista malleista löytyy esimerkiksi puolivalmiita peliohjaimia ja verkkolaitteita. Pinneihin kiinnitettävistä valmiista lisälaitteista käytetään nimitystä Arduino Shield. Niistä löytyy esimerkiksi puolivalmiita langattomia verkkolaitteita ja moottoriohjaimia. Kaikki Arduinon laitteistot ovat avoimia laitteistoja, eli niiden tekniset piirrokset ovat julkisia ja laitteistot saa myös rakentaa itse komponenteista niin halutessaan. Arduinon dokumentaatiot ovat luettavissa ja Arduinon ohjelmat ovat ladattavissa Arduinon verkkosivuilta osoitteesta arduino.cc. Samasta osoitteesta löytyvät linkit Arduinon foorumeille ja kehittäjäyhteisöihin. (Arduino 2014b)

4.1 Arduinon ohjelmointi

Kun Arduino on liitetty USB-kaapelilla tietokoneeseen, sen mikrokontrollerin voi ohjelmoida tietokoneelta. Arduinon ohjelmoimista varten tietokoneelle asennetaan Arduinon oma avoimen lähdekoodin ohjelmointiympäristö. Arduinon ohjelmointiympäristön ja muiden sen kaltaisten ohjelmistoympäristöjen nimitys on “Integrated Development Enviroment”, eli IDE. Kun Arduinon IDE on asennettu tietokoneeseen, Arduinon ohjelmoiminen voi alkaa. Ohjelmointiympäristön ikkunassa on alue kirjoitettavalle koodille ja selkeät painikkeet koodin varmentamista ja Arduinoon lataamista varten. Ikkunan valikoista taas löytyvät esimerkiksi ohjelmien tallennus, esimerkkejä ohjelmista ja Arduinon toiminnan seuraamiseen käytettävälle sarjaportille monitori. Myös Arduinoon mahdollisesti liitettävien Shieldien omat kirjastot tulevat näkyviin IDE:hen niiden asentamisen jälkeen.

Arduinon IDE on myös Arduinon GUI, eli sen 'Graphical User Interface', sen graafinen käyttöliittymä. Jos Arduinoa ei halua käyttää IDE:ltä, voi sitä ohjelmoida ainakin linuxilla myös suoraan tekstipäätteeltä ilman graafista käyttöliittymää. Kiinnostuksesta tutustuttuani IDE:n lähdekoodiin haluan kertoa siitä että se on kirjoitettu Javalla, ja perustuu sekä Java ohjelmointiin käytettyyn IDE:hen, Eclipseen, että Javan 'Java Development Toolkit':iin, eli jdt:hen. Eclipse ja jdt kuuluvat vapaan ja avoimen lähdekoodin sovellusten pariin, mutta Java ohjelmointikielenä taas ei ole kokonaan vapaa tai avoin, vaan osittain kaupallisessa kehityksessä. Arduinon IDE:n sisuksien tuntemuksesta on kuitenkin hyötyä lähinnä vasta kun itse IDE:tä haluaa muokata esimerkiksi mukauttaakseen sitä henkilökohtaisemmaksi tai tehdäkseen Arduinolle tietokoneelta sarjaportin kautta toimivan ohjaimen.

Arduinon ohjelmointikieli

Arduinolla on oma ohjelmointikielensä, joka perustuu C/C++:aan. Kieli toimii Arduinon IDE:ssä ja Arduinon ohjelmointia kutsutaan nimellä Programming. Kielen referenssiaineisto löytyy Arduinon verkkosivuilta, eli Arduinon sivuilla on esimerkkejä kielen käyttömahdollisuuksista ja sen ilmeisimmät konventiot selitettynä. Arduinon IDE:n lähdekoodiin tutustumalla ja IDE:tä käyttämällä selviää että Arduinon IDE:ssä Programmingilla ohjelmointia on avustamassa toinen kieli, xml. Xml avustaa Programmingia kirjoitettaessa tekstin sisennyksissä ja muissa ohjelmointikielten dogmaattista luonnetta pehmentävissä tehtävissä.

Programmingin alla Arduinon IDE:ssä on Arduinon kääntäjä, avr-g++. Avr-g++ kääntää Programmingin Arduinon mikrokontrollerin ymmärtämälle konekielelle binäärikoodiksi, eli ykkösiksi ja nolliksi. Avr-g++ on nimenomaan Arduinojen käyttämille Atmel AVR mikrokontrollereille tarkoitettu kääntäjä ja Atmelin mikrokontrollereita voi käyttää myös muilla kääntäjillä. Pinnan alla Arduinon IDE:ssä valmis koodi riisutaan ensin xml:stä ja muista Programmingin ominaisuuksistaan, jonka jälkeen avr-g++ käsittelee koodin C/C++:ana C-koodin kirjastonsa kautta ja kääntää sen konekielelle mikrokontrollerille. (Arduino 2014a) Kääntä-

misen yhteydessä IDE “debuggaa” koodin, eli tarkistaa sen virheiden varalta. Jos virheitä esiintyy, IDE ilmoittaa missä kohdin koodia niitä on ja mitä ne ovat. Kun koodi on virheetöntä, IDE:stä voi hiiren klikkauksella ladata koodin Arduinoon, missä se aloittaa toimintansa.

Programming toimii kuin C/C++, johon se pohjautuu. Programming-ohjelman aluksi määrittellään mitä headereitä ja kirjastoja se käyttää. Headerit ovat esikirjoitettuja ohjelman osia, joiden tarkoitus on säästää aikaa ohjelmoitaessa, kun ohjelman alkua ei tarvitse määritellä kokonaan joka kerta uudestaan. Kirjastotkin käsittävät ohjelmaa varten valmiiksi kirjoitettuja ohjelman osia, mutta yleisemmin. Headereiden ja kirjastojen määrittelyn jälkeen määritellään mitä osia laitteistosta käytetään ja nimetään miten niitä halutaan käyttää, eli esimerkiksi avataan sarjaportti, että voidaan seurata Arduinon toimintaa tietokoneelta tai nimetään miltä pinniltä ledi ottaa virtaa. Tämän jälkeen, tai samassa yhteydessä seuraavat muut määritelmät, joita ohjelmassa käytetään. Arduinolla on määritelmien jälkeen kaksi tapaa joilla se voi toimia: se voi suorittaa jonkin tehtävän kerran tai se voi suorittaa tehtävän toistuvasti. Näiden kahden tavan merkitseminen on seuraava vaihe ohjelmaa. Kerran toistuvan suorituksen koodi kirjoitetaan määritelmien perään. Toistuvan suorituksen, tai laajemman ohjelman koodi taas kirjoitetaan omiin lausekkeisiinsa, joissa se on valmis toistumaan alusta loppuun niin kauan kuin Arduino saa virtaa.

Arduinon foorumi

Arduinon verkkosivuilla on sen oma foorumi, missä Arduinosta kiinnostuneet voivat keskustella ongelmalähtöisestä oppimisestaan prototyypittelyn ja nikkaroinnin parissa. Lisäksi internetissä on useita muita Arduinolle omistettuja foorumeita, joilla keskustelut Arduinosta ovat usein osa mitä tahansa laajempaa keskustelua elektroniikasta ja ohjelmoinnista. Foorumeilta löytyy ohjeita ja neuvoja, kun useat niitä seuraavat Arduinosta kiinnostuneet kertovat omista kokemuksistaan ja kartuttavat foorumeiden tietokantoja. Kuten foorumeilla yleensä, jos aiheesta ei jo löydy aiempaa tietoa, voi kukin aiheen keksinyt lisätä omansa

foorumeille, joko kertomuksena huomioistaan tai kysymyksinä, avun pyyntöinä, miksei kritiikkinäkin.

4.2 Arduinon laitteisto

Arduino on avoin laitteisto, jonka voi niin halutessaan myös rakentaa itse. Valmiiden Arduinojen ja Arduinon lisäosien myymisen kannattavuus perustuu niiden massatuotantoon; eli Arduinojen tekeminen suurissa määrissä on merkittävästi edullisempaa kuin niiden valmistaminen yksitellen. Itsevalmistettua Arduinoa ei kuitenkaan saa markkinoida Arduinon nimellä, vaikka se olisi Arduinon piirikaavion perusteella tehty. Itsevalmistetusta Arduinosta käytetään esimerkiksi nimeä Freeduino. Freeduinojakin massatuotetaan ja valmistajien erilaisilla Freeduinoilla on omat nimensä. Arduinoja, kuten eri Freeduinojakin saa valmiina monia erilaisia. Arduinon perusmallin nimi on Arduino Uno. Muiden Arduinon mallien ominaisuuksiin kuuluvat erityiset lisäosat tai toisaalta perusosienkin poissaolo. Erityisistä lisäosista on hyötyä kun Arduinon käyttötarkoitus tiedetään ja esimerkiksi Motor Shield on voitu rakentaa suoraan osaksi Arduinoa. Kun osat ovat jo Arduinolla, ei Arduinoon tarvitse lisätä Shieldiä tai tarvitse rakentaa välttämätöntä piiriä, jolla se toimisi halutulla tavalla. Perusosien poissaolosta taas on hyötyä kun niille ei ole tarvetta. Riisutuimmassa Arduinon laitteistossa ei ole kuin liitännät johdoille ja itse mikrokontrolleri. Riisutuimmuudesta on hyötyä tietysti myös laitteiston koolle. Pienin Arduino on postimerkin kokoinen.

Arduinon Unon ostaessaan voi valita kahdesta: yhdessä mikrokontrolleri on integroituna piirilevyyn ja toisessa se on piirilevystä irrallaan omassa anturassaan. Anturaan istutetun mikrokontrollerin etuja ovat, että mikrokontrollerin hajotessa sen voi vaihtaa ja mikrokontrollerin voi siirtää laitteistosta toiseen. Toisessa piirilevyyn integroidun mikrokontrollerin mallissa taas on se haitta, että mikrokontrollerin hajotessa koko Arduino on käytännössä rikki, korjauskelvoton, ja joutuu hankkimaan uuden Arduinon. Piirilevyyn integroitu mikrokontrolleri on tosin pienempi. Kuitenkin kun ottaa huomioon kuinka helposti mikrokontrolleri saattaa hajota, ei voi olla ajattelematta että Arduinojen myymisen katteella saat-

taa olla massatuotannon lisäksi muitakin muodostumisen keinoja. Tosin voittohan Arduino ei tee.

Arduinon mikrokontrolleri

Eri Arduinojen malleissa on eri mikrokontrollereita. Mikrokontrollereita on useita malleja, mutta ne eivät vaadi perehtymistä ellei aio ryhtyä muokkaamaan Arduinon lähdekoodia muokatakseen Arduinon toimintaa, sillä niiden toimintaperiaate on sama. Mikrokontrollerit toimivat Arduinon Programminginin ehdoilla ja IDE hoitaa kaiken itse ohjelman kirjoittamista lukuun ottamatta.

Mikrokontrollereissa on pieni flash-muisti, minne konekielinen ohjelma tallentuu. Programming käyttää Arduinon ohjelmointiympäristöön ohjelmoituja kirjastoja, avr-kääntäjää ja laiteajureita kääntääkseen kirjoitetun koodin tiedostosta mikrokontrollerin ymmärtämälle konekielelle. Ja näin Arduino on ohjelmoitavissa suorittamaan monimutkaisiakin toimintoja. Mikrokontrollereiden flash-muisti on kuitenkin pieni ja siten rajallinen ottamaan muistiin laajempaa ohjelmaa, kuten tekoälyä.

Arduinon Shieldit

Arduinon perusmallissa pinniliittimiä on rajallinen määrä, eikä Arduino esimerkiksi myöskään sovellu sellaisenaan paljon virtaa käyttäviin laitteistoihin. Arduinoon onkin siksi valmiita lisäosia, jotka suoriutuvat tehtävistä, joihin Arduino ei muuten riittäisi. Näitä lisäosia kutsutaan Shieldeiksi. Käytännössä Shieldit ovat valmiita piirejä yleisimmistä tapauksista, joissa Arduino tarvitsee jonkin tietyn piirin suoriutuakseen jostain tehtävästä. Arduino ei valmista itse kaikkia Shieldejä, vaan yleisimmin Shieldit on suunnitellut ja valmistanut joku Arduinon suuresta yhteisöstä. Käytännössä tämä toimii kysynnän ja tarjonnan lain kautta; kun jokin piiri huomataan niin yleiseksi tai hyödylliseksi Arduinolle että sitä kannattaa alkaa valmistaa massatuotantona, sitä aletaan valmistaa massatuotantona.

4.2.1 Arduino Uno ja Adafruitin Motor Shield robotissa

Robotissa käytimme Adafruit Industriesin suunnittelemaa ja valmistamaa Motor Shieldiä. Arduino Unoon yhdistetyn Motor Shieldin tarkoitus on ohjata virtaa robotin moottoreille. Arduino Uno ei pystyisi käsittelemään moottoreiden useiden amppeerien virtavaatimusta, joten Motor Shield ja siihen yhdistetty virtalähde toimittavat virran moottoreille. Arduinon tehtäväksi jää ohjata kuinka paljon moottorit saavat virtaa Motor Shieldiltä, ja miten. Arduino siis huolehtii moottoreiden käynnistämisestä, pyörimisnopeudesta ja -suunnasta, ja sammuttamisesta.

Robotin alustana toimi Open Source robotiikkaa valmistavan DFRobotin nelivetoinen alustasarja. Alustaan kuului neljä vaihteistomoottoria, renkaat, johtoa ja alumiininen alusta, johon renkaat ja moottorit kiinnitettiin. DFRobotin nelivetoinen alustasarja on suunniteltu Arduinolla käytettäväksi ja mitoitettu Arduinolle. Alustasarjan ja Arduinon lisäksi robotti käytti DFRobotin URM37 ultraäänisensoria kuulonaan havaitakseen ympäristöään. Sensorin toiminta perustuu etäisyyden tunnistamiseen. Se lähettää ultraäänipulssin ja kuuntelee pulssin heijastusta. Pulssin heijastuksen paluuajasta Arduino laskee kuinka pitkä matka sensorista on siihen mistä ultraäänipulssi on heijastunut takaisin. Motor Shieldin lisäksi robotissa oli siis Arduinoon liitettynä myös sensori.

Sensoriin perustuva robotin toiminta Motor Shieldin kanssa toimi seuraavasti: Arduino on yhdistetty virtalähteeseen, Motor Shieldiin ja sensoriin. Motor Shield on yhdistetty moottoreiden virtalähteeseen ja moottoreihin. Käynnistyessään robotti liikkuu eteenpäin, kun Arduino ohjaa Motor Shieldiä antamaan virtaa moottoreille ohjelman mukaisesti. Sensori havainnoi mitä sen edessä on ja kommunikoi Arduinon kanssa; Arduino taas suorittaa ohjelmaa sensorin etäisyyshavainnon mukaisesti ja esimerkiksi ohjaa Motor Shieldiä laittamaan moottorit pyörimään toiseen suuntaan tai pysähtymään, jos robotti on ajamassa jotain päin. Tällä periaatteella ympäristöään harkitsevasti havainnoivasta robotista tuli esitykseen ensimmäinen automatamme.

4.2.2 Arduino Uno ja SKPangin DMXShield

DMX-ohjauksen laitteistona toimi Arduino Uno ja SKPangin valmistama DMXShield. Kokonaisuus oli hyvin yksinkertainen: Arduinoon liitettiin Shieldi DMX-liittimineen ja Arduinon kirjastoihin lisättiin Shieldin toimimisen vaatima kirjasto. Kirjaston kanssa tosin ilmeni ongelmia, kun sen asentamisessa selvisi, ettei sitä oltu ylläpidetty useisiin vuosiin. Ongelma oli kuitenkin pieni ja liittyi Arduinon päivityksiin: Shieldin kirjastosta piti vaihtaa yhden Arduinon päivityksen vuoksi vanhentuneen headerin nimi toiseksi, jonka jälkeen kirjasto toimi jälleen normaalisti. (ks. Tinkerit 2009) Laitteiston kanssa ilmeni toinenkin ongelma, kun himmentimen toiminta piti saada häiriöttömäksi. Rakentamamme laitteiston ominaisuuksiin ei vielä kuulunut häiriönpoisto ja kokeillun ja erehdyksen kautta ohjelmoidessamme valoja huomasimme että jäännösvirrat järjestelmässä saivat valot jäämään päälle tai palamaan kirkkaampina kuin niiden olisi kuulunut. DMX-järjestelmä on ketjuuntuva, eli useita DMX-laitteita voi ketjuttaa. DMX:n häiriönpoistajaa kutsutaan terminaattoriksi ja yksinkertaisuudessaan se on laite, joka laitetaan DMX-ketjun päähän syömään virran, joka muuten pyrkisi jatkamaan kulkuaan ketjua eteenpäin.

Terminaattori

Rakensimme terminaattorin ledeistä, DMX-liittimestä, vastuksesta ja avaimenperälampaasta. Terminaattoriksi olisi riittänyt, että kolvaamme DMX-urosliittimeen 120 ohmisen vastuksen sen signaali- ja maapinnin väliin, jolloin signaali olisi maadoittunut vastuksen läpi. (ks. Blue room 2008) Taiteellisista syistä päätimme kuitenkin lisätä terminaattoriin signaalipinnin ja maan väliin kaksi lediä ja koteloida koko Terminaattorin kuumaliimalla muovisen söpön avaimenperälampaan sisään. Näin, kun DMX-signaali maadoittuu Terminaattorissa, lampaan silmät vilkkuvat ledivaloa ja Terminaattori kulkee kätevästi mukana avaimenperänä.

Rajattomat ohjelmointimahdollisuudet

Tutustuessamme Arduinon käyttämiseen DMXShieldin kanssa selvisi useita eri tapoja ohjelmoida valot. Yleisesti teatterissa käytettyjen liukukytkimien rakentaminen Arduinoon oli yksi mahdollisuus. Ohjelmoinnilta se olisi vaatinut ainoastaan, että nimeämme kunkin kytkimen ohjelmaan ja määrittelemme ohjelmassa mitä kytkimen käyttämisestä tulisi seurata. Seuraava vielä yksinkertaisempi vaihtoehto oli rakentaa ohjelma, missä valot toimivat ajastuksella. Ohjelmasta saattoi tehdä rajattoman monimutkaisen; ohjelmoimme esimerkiksi valot huojumaan niin, että niiden huojunnan nousuvaihe oli eri kuin sen laskuvaihe. Valojen huojuessa niiden kirkkauden saattoi määrittää kahdeksan bittisesti, 256:n vaiheen tarkkuudella täsmällisesti ja huojunnan nousun ja laskun keston sekunnin tuhannesosina. Ajastuksella teimme valo-ohjelmaan huojuntaa, välähdyksiä ja satunnaisarvoja käyttäviä funktioita. Ajastuksella automoiduista valoista tuli esitykseen toinen automatamme. (Liite 2)

Seuraavaan vaihtoehtoon Arduinon käyttämisestä DMX-ohjauksessa liittyi jo tietokoneen käyttäminen. Tutustuessamme ohjelmistoihin, joita Arduinon ohjaamiseen olisi saattanut käyttää, huomasimme että Arduinoa saattoi käyttää hyvin monimutkaisten ohjausjärjestelmien kanssa. Löysimme Arduinoa ja Raspberry Pi:tä käyttävän avoimeen laitteistoon ja avoimen lähdekoodin sovelluksiin perustuvan projektin, joka tarjosi valmiit ohjelmistot ja asennusmediat. Raspberry Pi:n kanssa Arduinoa olisi voinut käyttää DMX-ajoihin palvelimena, jota ohjataan joko Raspberry Pi:ltä tai verkon kautta miltä tahansa tietokoneelta. (ks. Open Lighting 2014) Tutustuttuamme ohjausjärjestelmiin selvisi että ohjaukseen käytettäviä ohjelmistoja oli myös useita, joiden joukossa oli myös yleisesti multimedian ohjelmoinnissa käytetty kaupallinen multimedian visuaaliseen ohjelmointiin tarkoitettu MAX, ja Artnet, jolla esimerkiksi voi rakentaa IP-osoitteisiin perustuvia ohjauksia. Ohjausjärjestelmiin pintapuolisesti tutustuminen riitti kuitenkin meille ja koimme niiden tarjoaman hyödyn ylimitoitetuksi tekemäämme esitykseen. Esityksen valoista tai multimedioista ei ollut tarkoitus tehdä niin monimutkaisia että niiden ohjaaminen olisi vaatinut visualisointeja, tai niin laajoja että eri laitteistoja olisi tullut sijoiteltua verkkoon.

5 RASPBERRY PI

Raspberry Pi on tietotekniikan opiskeluun tarkoitettu pieni mutta täysiverinen tietokone. Sen laitteistoon kuuluu melko tehokas prosessori ja siihen integroitu hyvin tehokas grafiikkaprosessori. Lisäksi piirilevyltä löytyvät integroitu verkkokortti ja äänikortti. Raspberry Pi:n mukana ei tule oheislaitteistoja, mutta se tarvitsee virtalähteekseen mikro-USB:n, eli vaikka kännykän laturin, ja lisäksi kannattaa siihen liittää näyttö, näppäimistö, hiiri ja verkkokaapeli verkkoyhteyttä varten. Käyttötarkoituksesta riippuen Raspberry Pi:n voi koota eri tavoilla, ja siitäkin löytyy myös pinniliittimiä vaikka robotiikan rakenteluun. Raspberry Pi:n dokumentaatiot ovat luettavissa Raspberry Pi:n verkkosivuilla osoitteessa raspberrypi.org. Samasta osoitteesta löytyvät linkit Raspberry Pi:n foorumeille ja kehittäjäyhteisöihin, sekä ladattavat ohjelmat. (Raspberry Pi 2014)

Raspberry Pi:n käyttöjärjestelmäksi voi valita useista eri vapaan ja avoimen lähdekoodin Gnu/Linux-käyttöjärjestelmistä. Käyttöjärjestelmät on yhteisöllisesti suunniteltu Raspberry Pi:tä varten, eli ohjeita ja apua ongelmanratkaisuun saa yhtälailla kuin Arduinon kanssa nikkaroidessaankin. Ohjelmistotkin Raspberry Pi:lle ovat vapaita ja avoimia, eli myös ilmaisia. Eroksi muihin yleisimpiin linuxeihin jää vain, että Raspberry Pi:n käyttöjärjestelmän voi odottaa olevan yksinkertaistettu ja riisuttu verrattuna kotikäyttöön tarkoitettuun linuxiin. Raspberry Pi:n käyttöjärjestelmästä käytetäänkin nimitystä sulautettu järjestelmä, eli se on rakennettu se sulauttaen Raspberry Pi:hin, vain kyseistä laitteistoa varten.

Gnu/Linux ja sulautettu järjestelmä

Gnu/Linux on nimitys käyttöjärjestelmälle, joka on nykyisin kolmanneksi käytetyin maailmassa, ja yleisistä käyttöjärjestelmistä ainoa vapaan ja avoimen lähdekoodin sovelluksiin perustuva. ’Linuxien’ oikeaoppinen nimitys on Gnu/Linux, sillä Linux on niissä vain käyttöjärjestelmän pohjana muun käyttöjärjestelmän rakentuessa vapaan ja avoimen lähdekoodin ohjelmistoista, jotka ovat tavalla tai toisella sidoksissa Gnu-projektiin. Linuxeista käytännössä kaikki ovat ilmaisia

ja saatavissa verkosta ladattavina distroina. Vaikka linuxin asentaminen valmiista distroista on tehty hyvin yksinkertaiseksi, on matkassa monta muttaa: linuxit ovat ehdottomasti tietokoneharrastajien toisilleen tekemiä, eikä ainakaan asetusten tekeminen ole aina yksinkertaista. Kuitenkin linux on hyvin yksinkertainen ja looginen käyttöjärjestelmä. Yleisimmät linuxin ongelmat on ratkaistu jo kauan sitten, ja kun jokin ei toimi, yhteisö opettaa korjaamaan ongelman. Linuxia käyttämällä myös oppii nopeasti sen käyttämistä tukevat perusteet tietokoneiden toimintaperiaatteesta ja ohjelmoinnista.

Raspberry Pi:n käyttöjärjestelmiksi tarjotaan yleisimmistä linuxeista esimerkiksi Debian Gnu/Linux pohjaista Raspbiania. Raspbian on Debianin yhteisön rakentama linux Raspberry Pi:lle ja tekee Raspberry Pi:stä täysiverisen Debian-järjestelmän. Debianin vahvuuksiin kuuluu Debianin repository, eli ohjelmistopalvelin, joilta Debianin päivittäminen ja ohjelmistojen asentaminen käy hyvin yksinkertaisesti verkosta. Debianin ohjelmistopalvelimien ohjelmat käsittävät hyvin suuren osan kaikista FOSS-sovelluksista ja kaiken järjestelmän päivittämiseen liittyvän. Käytännössä tämä tarkoittaa, että mitään asennettavia ohjelmia ei tarvitse hankkia, vaan järjestelmä integroidaan ohjelmistopalvelimen kanssa ja ohjelmistot liitetään ohjelmistopalvelimelta järjestelmään. Debianin dokumentaatiot ovat luettavissa osoitteessa debian.org. Samasta osoitteesta löytyvät linkit Debianin foorumeille ja kehittäjäyhteisöihin, sekä ladattavat ohjelmat. (Debian 2014) Raspbianin dokumentaatiot taas ovat luettavissa osoitteessa raspbian.org. Samasta osoitteesta löytyvät linkit Raspbianin foorumeille ja kehittäjäyhteisöihin, sekä ladattavat ohjelmat. (Raspbian 2014)

Raspberry Pi:n käyttöjärjestelmät ovat sulautettuja järjestelmiä, vaikka esimerkiksi Raspbian ei juuri eroakaan kotikäytössä olevasta Debian järjestelmästä. Sulautettua järjestelmää voi ajatella käyttöjärjestelmänä, joka on jossain muussa yhteydessä käytössä kuin kotikoneella tai palvelimella ja juuri sitä laitteistoa varten rakennuttu, eli se on sulautettu laitteistoon. Raspberry Pi:n asennus tapahtuu lataamalla Raspberry Pi:n verkkosivuilta asennustiedosto ja siirtämällä sen sisältö SD-kortille. Raspberry Pi suosittelee ensikertalaisille NOOBS-pakettia, joka sisältää helppokäyttöisen graafisen asentimen ja useita eri asen-

nusvaihtoehtoja. NOOBS-paketin siirtäminen kortille on tehty mahdollisimman yksinkertaiseksi. Asennustiedosto on pakattu .zip:iksi ja se täytyy purkaa, mutta sitten tiedostot paketista voikin vain vaivattomasti kopioida kortille. Kun tiedostot on kopioitu, asennuksen voi aloittaa laittamalla kortin kasattuun Raspberry Pi:hin kiinni ja laittamalla Raspberry Pi:hin virran päälle. Asennuksen alettua näytölle tulee valikko ja ohjeet.

NOOBS:illa asennuksen alettua ensimmäiseksi valitaan millainen asennus halutaan tehdä. Vaihtoehtoina on asennus yleisimmistä linuxeista kotitietokonemaiseen käyttöön, asennus multimediakäyttöön, riisuttu linux tietokonenikkarointiin, tai ARM:n oma käyttöjärjestelmä. Valinta tapahtuu klikkaamalla hiirellä. Asennus on tästä eteenpäin automaattinen ensimmäiseen käynnistykseen asti. Asennus kestää joitain kymmeniä minuutteja ja asennuksen ajan voi seurata sen etenemistä, kun asennusohjelma näyttää kuinka paljon asennuksesta on tehty ja kertoo samalla ohjeita siitä miten asennus otetaan käyttöön, kun se on valmis. Kun asennus on valmis, asennusohjelma käynnistää Raspberry Pi:n uudestaan. Ensimmäisellä käynnistyksellä asennusohjelmasta valitaan miten käyttöjärjestelmää halutaan käyttää ja tehdään joitain muita asetuksia.

Valo, multimedia ja ääni

Kuten yllä mainostinkin, Raspberry Pi:lle löytyy myös linux, joka on rakennettu DMX:llä tapahtuvaa valo-ohjelmointia varten. Open Lighting on projekti, jonka pyrkimyksenä on tehdä valaistukseen liittyvistä järjestelmistä Open Sourcea. Open Lighting projekti käsittää eri ohjelmistoja, joilla voi ohjata DMX järjestelmiä ja myös valmiita ohjaimia, jotka toimivat projektin ohjelmilla. Open Lightingin ohjelmistopaketin nimi on OLA, eli Open Lighting Architecture. Vapaan ja avoimen lähdekoodin ohjelmistona se on ilmaiseksi saatavissa projektin verkkosivuilta. OLA toimii myös Raspberry Pi:llä ja Raspberry Pi:lle onkin saatavissa valmiiksi tehty, Rasbianiin pohjautuva linux, jossa OLA on valmiina käyttöön heti käyttöjärjestelmän asentamisen jälkeen. OLA:n dokumentaatiot ovat luettavissa osoitteessa opendmx.net. Samasta osoitteesta löytyvät linkit Open Lightingin

foorumeille ja kehittäjäyhteisöihin, sekä ladattavat ohjelmat. (Open Lighting 2014)

Raspberry Pi:n käyttäminen valoajojen ohjaimena tarjoaa useita eri vaihtoehtoja, joista OLA:n käyttäminen on yksi. OLA:n kanssa Raspberry Pi:llä voi tehdä valo-ohjelmat, joiden käynnistäminen ja ohjaaminen tapahtuu esityksessä Raspberry Pi:ltä. OLA:n kanssa Raspberry Pi voi toimia myös palvelimena, johon otetaan yhteys verkosta toiselta tietokoneelta, jolla valo-ohjelmien tekeminen, käynnistäminen ja ohjaaminen tapahtuu. OLA toimii myös Artnetin kanssa, mikä mahdollistaa myös valojen ohjaamisen IP-osoitteiden kanssa. Vaihtoehtona OLA:n käyttämisen lisäksi on esimerkiksi ohjaimen tekeminen itse.

Raspberry Pi:n grafiikkaprosessori pystyy FullHD toistoon, joten esimerkiksi projektorien liittäminen Raspberry Pi:hin onnistuu hyvin. Muita multimedian vaatimuksia, esimerkiksi kineettisiä osia, kuten robotteja, Raspberry Pi:hin voi tehdä sen pinniliittimen kautta tai sen USB porttien välityksellä. Raspberry Pi on niin pienivirtainen, että sen voi liittää osaksi multimedia-ohjausta myös paristolla toimivana. Palvelimena Raspberry Pi ei pysty suoriutumaan raskaasta verkkoliikenteestä, mutta omalle kotisivulle tai multimediapalvelimeksi se riittää. Eli käytännössä erilaisten IP-osoitteita käyttävien ohjausten tekeminen Raspberry Pi:llä onnistuu, koska liikkuvan datan määrän voi ennalta arvioida.

Ääniajoonkin Raspberry Pi:n käyttäminen tarjoaa laitteistona useita vaihtoehtoja: äänen voi ottaa HDMI:n kautta, analogisen stereoplugin kautta tai USB:ltä. Raspbianin mukana tulee myös valmiiksi asennettu linuxin ääniohjaukseen tarkoitettu ohjelmisto Jack, millä Raspberry Pi:n voi liittää mideihin. Jackillä voi yhdistää useita eri äänilähteitä mideiltä, ulkoisilta äänikorteilta ja laitteiston sisäisesti, sekä ohjelmoida että ohjata näitä haluamallaan tavalla. Jack:iin on myös useita lisäohjelmia, joilla esimerkiksi voi muokata tai analysoida ääntä. Myös samalla kun ääntä soittaa. Ohjelmistojen puolesta mahdollisuudet ovat rajattomat. Erilaisia ohjelmia soittimiksi on lukemattomia ja niitä voi tehdä itse. Protossa käytimme muokattua VLC-soitinta, johon teimme soittolistan esityksen kappaleista. Kun vielä laitoimme soittolistan käynnistymään tietokoneen käynnistymisen yhteydessä, oli meillä esitykseen kolmaskin nykyaikainen automata.

6 LOPUKSI

Proton automoidusta tekniikasta tuli hyvin toimiva ja dynaaminen osa esitystä. Alussa huolenani esityksen ohjaajana oli ollut että esityksen automaatiot vaatisivat näytteleminseltä jotain aivan kummallista. Automoidun tekniikan kanssa esityksen tekeminen ei vaatinut kuitenkaan näytteleminseltä koreografioimista, kun ajastuksin toimivien funktioiden sisäänajot oli ohjelmoitu limittäin vaihtuviksi. Automatojen kanssa näyttämöllä oleminen ei myöskään ollut hankalaa sen puolesta että automatat olisivat vieneet liikaa huomioita tai toimineet väärin. Kokonaisuutena automoitu tekniikka ei tuonut esiintymiselle yhtään lisätyötä, vaan kokonaisuus sujui kuin tanssi. Tai kävi kuin kellon koneisto. Erityisesti esityksen videoituamme teimme videosta huomion, että valo- ja ääniautomaatiot toimivat hyvin videonkuvassa, ja voisinkin kuvitella että erityisesti jos esitys on tarkoitus taltioida videolle, pitkälle viedyistä automaatiosta on vain hyötyä. Myös videolle näyttelemiselle, jolloin näyttelijä voi hyödyntää automaatioiden ajastuksia.

Automaatiosta oli hyötyä esiintymisen vapaudellekin, kun muutoin valo- ja ääniohjauksia olisi joutunut aktiivisesti hallitsemaan kummitustutkijalle rakennetulta konsolilta. Vastaavuudessa olisi tosin mielenkiintoista kokeilla esiintymistilanteessa myös miten haastavaa aktiivinen tekniikan ohjaaminen olisi, ja myös millaista olisi esiintyä kun tekniikka olisi automoitu vielä vähän pidemmälle, niin että esiintyjä ei vaikuttasi sen ajastuksiin ollenkaan. Proton valoautomaatiot käynnistettiin esiintyjän toimesta, kun hän oli siihen valmis, ja musiikkikappaleet esiintyjän toimesta kun hän oli tekstissä oikeassa kohdassa. Robotti taas oli ohjelmoitu suorittamaan valo-ohjelman kanssa kestoltaan yhtä pitkän rutiinin ja lopettamaan toimintansa vähän ennen kuin valo-ohjelma ajoi valot alas. Robotin osalta vastaavuudessa olisi mielenkiintoista rakentaa se reagoimaan esiintyjän sille antamiin ärsykkeisiin ja saada aikaan suunniteltua dialogia koneen ja esiintyjän välillä. Tämä oli meillä tarkoituksena jo tähän esitykseen mutta emme ehtineet suunnitella osia tilaukseen. Konsolin rakentamisesta näyttämölle osaksi lavastusta taas kuvittelisin olevan hyötyä erityisesti sooloesityksiä tehtäessä,

kun esitykselle ei ole välttämättä tarvetta hankkia henkilöä huolehtimaan tekniikasta, vaan esiintyjä voi ajaa tekniikan näyttämöltä.

Koska lähes kaikki yllä käsitellyistä laitteistoista ja komponenteista on kätevimmin saatavissa vain verkosta, on hyvin merkityksellistä suunnitella ja aikatauluttaa osien tilaaminen. Suunnitteluun, aikatauluttamiseen ja myös budjetointiin tulee huomioida mistä osat tulevat postitusaikojen ja mahdollisten tullien vuoksi. On tavallista että osat saapuvat useasta eri maasta. Yhteistilaukset tulevat edullisemmaksi kuin tilata osia yksi kerrallaan, ja kuitenkin yksi kerrallaankin tilaaminen saattaa olla edullisempaa kuin tilata osat maahantuojilta tai ostaa erikoisliikkeistä.

Osien tilaamisen suunnitelmallisuuden lisäksi on merkityksellistä että rakennus ja ohjelmointitehtävät on hyvin suunniteltu. Proton tapauksessa teimme osien valinnan ja tilaamisen yhdessä, mutta robotin ja konsolin rakentaminen tehtiin yksilöittäin. Päätimme parhaaksi antaa sen rakentaa laitteistot kenellä oli eniten motivaatiota ja hyödyimme siitä sen että koko ajan ainakin yksi tiesi missä tilassa laitteisto oli ja tiesi mistä lähteä ongelmia kokeilun ja erehdyksen kautta ratkomaan. Sama päti myös ohjelmointiin. Ohjelmoinnin pohjatyöt, kuten kirjastojen hankinnat ja esimerkkeihin tutustuminen tehtiin yhdessä, mutta lopulta ohjelmointien rakentaminen ja viimeistely tehtiin yksilöittäin. Hyödyt ohjelmien rakentamisesta näin olivat samat kuin laitteistojenkin rakentamisessa. Motivaatioon ja asiantuntijuuteen kannattaa panostaa.

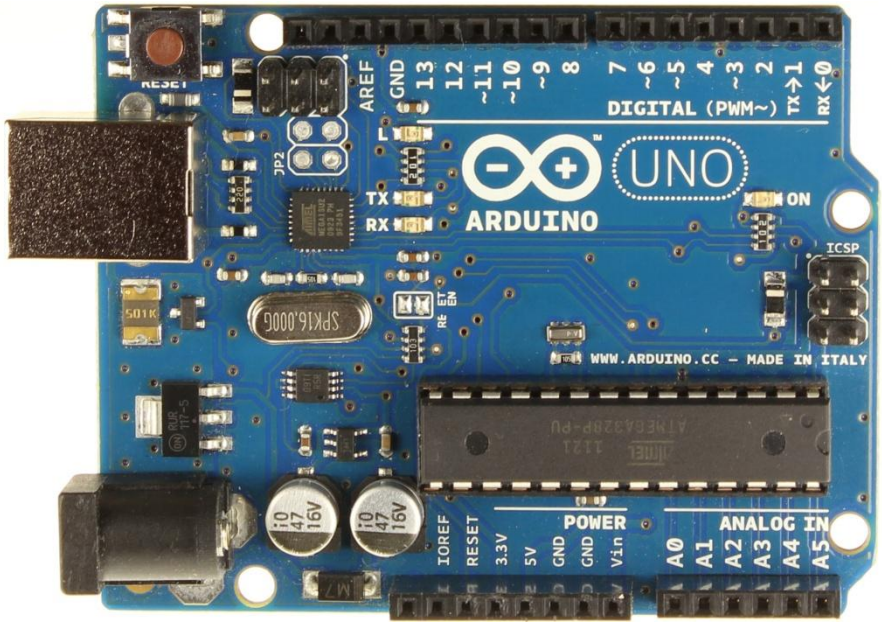
Kokeilu ja erehdys oli yksi kokonaisuutta kantava teema ja piti mielenkiintoa yllä. Teippaamisen katsominen parhaaksi tavaksi rakentaa konsoli oli kokeilun ja erehdyksen kautta saaduista tuloksista huvittavin. Olimme jo ehtineet tehdä hienon koteloinnin ruuvinpaikkoineen kaikkineen, kun huomasimme, ettei kevyt laitteisto tarvitse kuin palat kaksipuoleista teippiä piirilevyjen ja tukilevyn väliin. Robotissa sensorin kalibrointi taas vaati monta kokeilua ja erehdystä kun sensorin oikea toiminta vaati tarkat sensorin toimintaetäisyyksien määritykset ohjelmaan. Näitä ennen saimme seurata ihmettyneinä sekavasti tai vain hetkittäin toimivaa robottia. Myös tekniikkaohjaukset vaativat omat kokeilunsa ja erehdyksensä, kun etsimme parasta tapaa ajoittaa ja käynnistää valot ja äänet. Ar-

duinolle löytyi esimerkiksi erilaisia itse tehtyjä Arduinon IDE:hen perustuvia GUI:ta, joilla Arduinon toimintaa saattaisi ohjata myös Raspberry Pi:ltä mutta valmista DMX-ajoon Arduinon IDE:stä muokattua ohjainta ei kuitenkaan löytynyt valmiina. Lähimmäksi pääsi Python-ohjelmointiin perustuva Arduinon käyttämän DMXSimplen:n toimintaa ohjaava ohjelmisto, joka toimi vain tekstipäätteeltä. Emmekä lähteneet tekemään GUI:ta itsekään, vaikka ehkä olisimme osanneetkin.

Protoon rakennetuissa automatoissa saimme esiin monta automatoille perinteistä ominaisuutta. Yksi automatoistamme liikkui ja esiintyi, toinen huolehti valoshowsta ja kolmas soitti esityksen äänet automoidusti. Perinteisemmin tehtynä olisimme ehkä itkettäneet automatoillamme pyhimyksen kuvaa tai automoineet jonkin salaoven. Ja voimmehan tehdä niin ensi kerralla, tai joku muu voi niin tehdä, sillä toivon tässä antaneeni ainakin suuntaviivoja sille miten automaatioita voi yksinkertaisesti tehdä, jos en neuvonut kädestä pitäen. Arduinon kaltaisiin protyypittelytyökaluihin ja sulautettuihin järjestelmiin liittyvä tekniikka edistyy näin vuonna 2014 niin huimaa vauhtia, ettei tässä tekstissä mielestäni ollut merkitystä antaa täsmällisempiä esimerkkejä tai esimerkiksi keskustella ohjelmoinnin konventioista enempää. Aikahaarukka kuitenkin siihen että kaikki on uudenlaista ja parempaa ei ole vuosi, ei edes kuukausi, vaan ennemminkin pari viikkoa. Ja nähdäkseni paras tapa pysyä mukana on olla kokemassa yhdessä tekeminen.

Teatterin tekniikkaa ajatellen pitäisin jatkossakin silmällä juuri Arduinon ja sulautettuja järjestelmiä opettavien avointen laitteistojen mahdollisuuksia. Näiden lisäksi ehkä avoimet mobiililaitteistot saattavat tuoda hyvinkin pian jännittäviä lisiä, kun erilaisista kaapeloinneista päästään eroon ja tekniikasta tulee lisääntyvästi langatonta. Kineettistä multimediaa, kuten robotiikkaa tulee myös olemaan mielenkiintoista pitää silmällä. Ubiikkia on hienoa nähdä teoksina. Jonain päivänä piankin toivoisin näkeväni näyttämöllä esityksessä yleisön mobiililaitteiden ja internetin kanssa interaktiivisesti kyberavaruutta tiedostavan automatan, joka voi näyttää kineettisesti ja multimedialla mitä emme muuten verkolta näkisi.

7 KUVAT



Kuva 1. Arduino Uno Rev3. (Arduino 2014c)

Kuva 2. Raspberry Pi Rev1. (Raspberry 2014b)

LÄHTEET

Arduino 2014a. FAQ. Viitattu 14.3.2014 http://arduino.cc/ > Support > FAQ

Open Source 2014. The Open Source Definition. Viitattu 14.3.2014 http://opensource.org > The Open Source Definition

GNU 2013a. What is free software? Viitattu 14.3.2014 http://www.gnu.org > Philosophy > What is free software?

Parallax 2014. Boe-Bot Robot. Viitattu 14.2.2014 http://www.parallax.com/product/boe-bot-robot

Adafruit 2012. Adafruit Motorshield FAQ. Viitattu 14.3.2014 http://learn.adafruit.com/adafruit-motor-shield/faq

Torpey, P.A. 2009. Disembodied Performance. Abstraction of Representation in Live Theater. 52, 55. Massachusetts Institute of Technology. Viitattu 14.3.2014 http://web.media.mit.edu/~patorpey/publications/torpey_sm_thesis_2009_disembodied_performance.pdf

GNU 2013b. What is Copyleft? Viitattu 14.3.2014 http://www.gnu.org/copyleft

GNU 2013c. Licences. Viitattu 14.3.2014 http://www.gnu.org > Licences

Arduino 2014b. Arduino. Viitattu 14.2.2014 http://www.arduino.cc

Tinkerit 2009. DMXSimple.Viitattu 14.3.2014 http://code.google.com/p/tinkerit/wiki/DmxSimple

Blue room 2008. Terminator. Viitattu 14.3.2014 http://www.blue-room.org.uk/wiki/DMX_Terminator

Open Lighting 2014. Open Lighting Project. Viitattu 14.3.2014 http://opendmx.net/index.php/Open_Lighting_Project

Raspberry Pi 2014a. Raspberry Pi. Viitattu 14.3.2014 http://www.raspberrypi.org/

Debian 2014. Yleismaailmallinen käyttöjärjestelmä. Viitattu 14.3 2014 http://www.debian.org/

Raspbian 2014. Raspbian. Viitattu 14.3.2014 http://raspbian.org/

Arduino 2014c. Arduino Uno Rev3. Viitattu 14.3.2014 http://arduino.cc/ > Products > Arduino Uno > Arduino Uno R3 Front

Raspberry 2014b. Revision 1 Raspberry Pi Viitattu 14.3.2014 http://www.raspberrypi.org/ > FAQs > Revision 1 Raspberry Pi

Robotin lähdekoodista esimerkki

```
#include <Wire.h>
#include <Adafruit_MotorShield.h>
#include "utility/Adafruit_PWMServoDriver.h"

Adafruit_MotorShield AFMS = Adafruit_MotorShield();
Adafruit_DCMotor *FR = AFMS.getMotor(1);
Adafruit_DCMotor *FL = AFMS.getMotor(2);
Adafruit_DCMotor *BL = AFMS.getMotor(3);
Adafruit_DCMotor *BR = AFMS.getMotor(4);

int URPWM = 11; // PWM Output 0－25000US,   Every 50US represent 1cm
int URTRIG=12; // PWM trigger pin

int ledR = 7;
int ledG = 8;
int ledB = 3;

unsigned int Distance=0;
uint8_t EnPwmCmd[4]={0x44,0x02,0xbb,0x01};    // distance measure command

int perc = 3;

void setup() {
  Serial.begin(9600);           // set up Serial library at 9600 bps
  AFMS.begin();  // create with the default frequency 1.6KHz
  // Set the speed to start, from 0 (off) to 255 (max speed)
  FL->setSpeed(50);
  FL->run(FORWARD);
  FL->run(RELEASE);

  FR->setSpeed(50);
  FR->run(FORWARD);
  FR->run(RELEASE);

  BL->setSpeed(50);
  BL->run(FORWARD);
  BL->run(RELEASE);

  BR->setSpeed(50);
  BR->run(FORWARD);
  BR->run(RELEASE);

  pinMode(ledR, OUTPUT);
  pinMode(ledG, OUTPUT);
  pinMode(ledB, OUTPUT);
  PWM_Mode_Setup();
}

void PWM_Mode_Setup(){
```

```
  pinMode(URTRIG,OUTPUT);                  // A low pull on pin COMP/TRIG
  digitalWrite(URTRIG,HIGH);               // Set to HIGH
  pinMode(URPWM, INPUT);                   // Sending Enable PWM mode
command

  for(int i=0;i<4;i++){
     Serial.write(EnPwmCmd[i]);
   }
}

void loop()

{
        //Serial.println(millis());
        if(millis()<900000) // 900 000ms = 15min
        {
                digitalWrite(URTRIG, LOW);
                digitalWrite(URTRIG, HIGH); // reading Pin PWM will output pulses
                unsigned long DistanceMeasured=pulseIn(URPWM,LOW);

                if(DistanceMeasured==50000){             // the reading is invalid.
                        Serial.print("Invalid");
                }
                else{
                        Distance=DistanceMeasured/50;          // every 50us low level
stands for 1cm
                }
                Serial.println(Distance);
                if(Distance>10&&Distance<1400)
                {
                        driveForward();
                        randomLights();
                }
                else
                {
                        driveStop();
                        randomAway();
                        driveTurnLeft();
                        driveStop();
                }
                resetSpeed();
                int rndVal = random(101);
                //Serial.println(rndVal);
                if(rndVal<perc)
                {
                        int loopC = 0;
                        digitalWrite(ledR,0);
                        digitalWrite(ledG,0);
                        digitalWrite(ledB,0);
                        do
                        {
                                Serial.println("Thinking");
                                think();
```

```
                        loopC++;
                    }while(loopC<3);
                    randomAway();
                    resetSpeed();
                }
                delay(50);
        }
        else
        {
                driveFullStop();
                digitalWrite(ledR,0);
                digitalWrite(ledG,0);
                digitalWrite(ledB,0);
                //blink(ledR);
                //blink(ledG);
                // Shutdown procedure
        }
}

void randomAway()

{
  int choice = random(4);
  switch(choice)
  {
    case 0: driveTurnLeft(); break;
    case 1: driveTurnRight(); break;
    case 2: driveTankCW(); break;
    case 3: driveTankCCW(); break;
    default: driveTurnLeft(); break;
  }
}

void randomLights()

{
  int rv = random(256);
  int gv = random(256);
  int bv = random(256);
  analogWrite(ledR,rv);
  analogWrite(ledG,gv);
  analogWrite(ledB,bv);
}

void driveForward()

{
  FR->run(FORWARD);
  FL->run(FORWARD);
  BR->run(FORWARD);
  BL->run(FORWARD);
}
```

```
void driveStop()

{
  FR->run(RELEASE);
  FL->run(RELEASE);
  BR->run(RELEASE);
  BL->run(RELEASE);
}

void driveTurnLeft()

{
    FR->run(BACKWARD); //Hitaamin
    FL->setSpeed(255);
    FL->run(BACKWARD);
    BR->run(BACKWARD); //Hitaamin
    BL->setSpeed(255);
    BL->run(BACKWARD);
    delay(500);
}

void driveTurnRight()

{
    FR->run(BACKWARD);
    FR->setSpeed(255);
    FL->run(BACKWARD); //Hitaamin
    BR->run(BACKWARD);
    BR->setSpeed(255);
    BL->run(BACKWARD); //Hitaamin
    delay(500);
}


void driveTankCW()

{
  FR->run(BACKWARD);
  FL->run(FORWARD);
  BR->run(BACKWARD);
  BL->run(FORWARD);
  delay(500);
}

void driveTankCCW()

{
  FR->run(FORWARD);
  FL->run(BACKWARD);
  BR->run(FORWARD);
  BL->run(BACKWARD);
  delay(500);
}
```

```
void resetSpeed()

{
  FR->setSpeed(50);
  FL->setSpeed(50);
  BR->setSpeed(50);
  BL->setSpeed(50);
}


void driveFullStop()

{
  FR->run(RELEASE);   FR->setSpeed(0);
  FL->run(RELEASE);    FL->setSpeed(0);
  BR->run(RELEASE);   BR->setSpeed(0);
  BL->run(RELEASE);    BL->setSpeed(0);
}

void think()

{
	driveFullStop();
	for(int d=0;d<256;d++)
	{
		analogWrite(ledR,d);
		delay(10);
	}
	delay(200);
	for(int j = 255; j>0; j--)
	{
		analogWrite(ledR,j);
		delay(10);
	}
}

void blink(int led)

{
	for(int i = 0; i<10; i++)
	{
		Serial.println("Honk");
		digitalWrite(led, HIGH);
		delay(50);
		digitalWrite(led,LOW);
	}
}

void fadeOut(int led)

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

```
        {
                analogWrite(led,i);
        }
}
```

Valo-ohjauksen lähdekoodista esimerkki

```
#include <DmxSimple.h>

boolean toggle = false;

int l4k1        = 1;    // Oikea taka
int l2k2        = 2;    // Keski taka
int l11k3       = 3;    // Oikea etu
int l8k4        = 4;    // Oikea keski
int l9k5        = 5;    // Keski keski

void setup() {


  DmxSimple.usePin(3);
  DmxSimple.maxChannel(6);  // Määritetään maksimi kanavien määrä
  // Valitaan kanavat ja alustetaan ne olemaan pois päältä.
  DmxSimple.write(l4k1, 0);
  DmxSimple.write(l2k2, 0);
  DmxSimple.write(l11k3,0);
  DmxSimple.write(l8k4, 0);
  DmxSimple.write(l9k5, 0);
  Serial.begin(9600);
}

void loop()


{
  // vihreä 15
  // fresu 40
  // taka oikea 40
  // keski taka ja etu oikea 70
        // 1 260 000 ms = 21 min
        //        900 000 ms = 15 min
        //       180 000 ms =  3 min
  // lights on
  int steps = 70;
  //unsigned long asd = calcDelay(100,180000);
  //Serial.println(asd);
  //toggle = true;
  if(!toggle)
  {
    // Isku 1: "Kaikkiloppuu...", laitteisto kytketään päälle
    for(int i=0; i<steps; i++)
    {
                if(i<41)
                {
                  //DmxSimple.write(l4k1,i);
                  DmxSimple.write(l9k5,i);
                }
```

```
            //DmxSimple.write(l2k2,i);
            DmxSimple.write(l11k3,i);
            if(i<15)
            {
              DmxSimple.write(l8k4,i);
            }
            delay(3428);
            //delay(100);
            Serial.println(i);
}
Serial.println("Taka oikea fader");
// Isku 2: Rap musa loppuu
for(int i = 0; i<41;i++)
{
  DmxSimple.write(l4k1,i);
  delay(1000);
}
int l1Huou = 0;
long startHuoju1 = millis();
do
{
  l1Huou = random(35,56);
  DmxSimple.write(l4k1,l1Huou);
}while((millis()-startHuoju1)<60000);
DmxSimple.write(l4k1,30);

Serial.println("Keski keski fader");
for(int i = 0; i<61; i++)
{
  DmxSimple.write(l2k2,i);
  delay(1200);
}
// Isku 3: "Mitä tämän jälkeen on?"

Serial.println("Fresu blinker");
fresuBlinker(l9k5, 50, 0);
DmxSimple.write(l9k5, 40);
Serial.println("Blinker");

// Isku 4: "Tämän jälkeen sinua ei odota enään mikään."

for(int i=15; i<50; i++)
{
  DmxSimple.write(l8k4,i);
  delay(2571);
}
delay(120000);
blinker(l8k4);
// Isku 5: "MITÄ NYT?"

DmxSimple.write(l11k3,0);
DmxSimple.write(l8k4,0);
delay(90000);
fresuBlinker(l9k5, 1000, 100);
delay(60000);
```

```
    fresuBlinker(l9k5,1000,100);
    DmxSimple.write(l9k5,40);
    // Isku 6: "Ja jos otat mut..."
    for(int i = 60; i>20; i--)
    {
                if(i<41)
                {
                  DmxSimple.write(l4k1,i);
                  DmxSimple.write(l9k5,i);
                }
                DmxSimple.write(l2k2,i);
                delay(2250);
                //delay(100);
                Serial.println(i);
    }
    for(int i = 20; i>0; i--)
    {
      DmxSimple.write(l4k1,i);
      DmxSimple.write(l9k5,i);
      DmxSimple.write(l2k2,i);
      delay(1500);
    }
    DmxSimple.write(l4k1,0);
    DmxSimple.write(l2k2,0);
    DmxSimple.write(l11k3,0);
    DmxSimple.write(l8k4,0);
    DmxSimple.write(l9k5,0);
    Serial.println("Toggle true");
    toggle = true;
  }
}

void blinker(int channel)
{
  long milStart = millis();
  do
  {
    for(int i=0; i<50;i++)
    {
      DmxSimple.write(channel,255);
      delay(random(50));
      DmxSimple.write(channel,50);
      delay(500);
    }
    DmxSimple.write(channel,0);
    delay(5000);
  }while((millis()-milStart)<60000);
}

void fresuBlinker(int channel, int rndMax, int rndMin)

{
```

```
  long milStart = millis();
  do
  {
    for(int i=0; i<20;i++)
    {
      DmxSimple.write(channel,150);
      delay(random(rndMin, rndMax));
      DmxSimple.write(channel,50);
      delay(500);
    }
    DmxSimple.write(channel,0);
    delay(5000);
  }while((millis()-milStart)<30000);
}
```