OULUN SEUDUN
AMMATTIKORKEAKOULU

Hanna Nauska-Vaara

FLOWCODE-OHJELMOINNIN PERUSTEITA JA HARJOITUKSIA

FLOWCODE-OHJELMOINNIN PERUSTEITA JA HARJOITUKSIA

Hanna Nauska-Vaara
Opinnäytetyö
Kevät 2012
Automaatiotekniikan koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Automaatiotekniikka

Tekijä(t): Hanna Nauska-Vaara
Opinnäytetyön nimi: Flowcode-ohjelmoinnin perusteita ja harjoituksia
Työn ohjaaja(t): Kari Jyrkkä
Työn valmistumislukukausi ja -vuosi: Kevät 2012 Sivumäärä: 35 + 3 liitettä

Tämä opinnäytetyö tehtiin Oulun seudun ammattiopiston Myllytullin yksikölle. Työn tavoitteena oli tehdä oppimateriaali, jota tieto- ja tietoliikennealan Poweritiimin opettajat voivat käyttää aloittaessaan flowcode-ohjelmoinnin perusteiden opettamisen. Lisäksi tavoitteena oli testata materiaalin toimivuutta yhden ryhmän kanssa. Koska flowcodea ei ole yksikössämme aiemmin käytetty, haluttiin materiaalin sisältävän sekä teoriaa että ohjelmointiharjoituksia.

Ensin työssä perehdyttiin flowcodeen ohjelmointikielenä ja rajattiin sulautettuja järjestelmiä koskeva teoria minimiin. Lisäksi täytyi miettiä, kuinka paljon esitellään mikrokontrollerin toimintaa ottaen huomioon ajan rajallisuus ja opetussuunnitelman perusteiden tavoite.

Materiaalissa esitellään flowcode-ohjelmointikielen peruskäskyt ja toiminnot. Lisäksi siinä on sanallisia tiedonhakutehtäviä, joilla pyritään aktivoimaan opiskelijaa ja lisäämään hänen tietomääräänsä aiheesta. Materiaalin laatimisen taustateoriana on konstruktivistinen oppimiskäsitys, joka pitää oppijaa aktiivisena toimijana ja johon on myös materiaalin laadinnassa pyritty.

Työn tavoite saavutettiin pääosin. Materiaali saatiin valmiiksi ja sen toimivuutta ehdittiin kokeilla, jopa kahdella ryhmällä. Oppimateriaalia voidaan käyttää yksikkömme tieto- ja tietoliikennealan aloittavien luokkien opetukseen. Kun materiaalia käytetään, sitä voidaan täydentää aina sen mukaan, mikä huomataan tarpeelliseksi. Lisäksi ajan myötä tiimin opettajat voivat kehittää lisää harjoituksia ja liittää niitä materiaaliin.

Asiasanat:
Ohjelmointi, oppiminen, oppimiskäsitys, oppimateriaali, Flowcode

ABSTRACT

Oulu University of Applied Sciences
Automation Engineering

Author(s): Hanna Nauska-Vaara
Title of thesis:The basics and exercises of the Flowcode programming
Supervisor(s): Kari Jyrkkä
Term and year when the thesis was submitted: Spring 2012 Pages: 35 + 3 appendices

---

This thesis was made to the Myllytulli unit of Oulu Vocational College. The aim of the thesis was to create a learning material which the teachers of the Electrical Engineering Poweri team could use when teaching the Flowcode programming. To test the material's functionality with a team of students was an additional objective. The material consists of theory and programming exercises.

First the Flowcode as a programming language was familiarized in the thesis and the theory of the embedded systems was left to minimum. Secondly the microcontroller's functionality was introduced.

The basic commands and functions of the Flowcode programming language was introduced. Furthermore the learning material included information searches. That activated students and increased amount of information. The learnig theory of the constructivism was selected as the background theory of the learning material, it expresses students as an active author.

The objectives were achieved mainly. The learning material was ready in time and it was tested with two student groups. The learning material can be used in teaching the first year's students of the Electrical Engineering department. The material will be developed further by teachers.

Keywords:
Programming, learning, learnig theory, learning material, Flowcode

ALKULAUSE

Tämä opinnäytetyö on tehty Oulun seudun ammattiopiston Myllytullin yksikön Poweritiimin käyttöön. Haluankin kiittää tiimiäni tuesta ja erityisesti Jouni Lievosta kiinnostuksesta aihetta kohtaan. Lisäksi haluan kiittää ohjaajaani Kari Jyrkkää kärsivällisyydestä, vaikka työni viivästyi. Lopuksi kiitän miestäni Keijo Vaaraa, joka rahoitti opintovapaani ja mahdollisti tämän opinnäytetyön valmistumisen.

Hanna Nauska-Vaara

SISÄLLYS

1 JOHDANTO

Aiheen opinnäytetyöhöni sain omasta tiimistäni. Opetan Oulun seudun ammattiopistossa Myllytullin yksikössä ammattiaineita tieto- ja tietoliikennetekniikan perustutkintoa ensimmäistä vuotta opiskeleville nuorille. Oulun seudun ammattiopisto (OSAO) koostuu 13 yksiköstä, joissa annetaan ammatillista peruskoulutusta seitsemällä eri alalla. Myllytullin yksikkö on monialainen yksikkö ja siellä opiskelee vajaa 1000 nuorta eri aloilla.

Ammatillisten perustutkintojen opetussuunnitelmien perusteet uusittiin vuosina 2008 -2010. Yksikössämme aiemmin käytössä ollut sähköalan perustutkinto, elektroniikka-asentajan koulutusohjelma muuttui vuonna 2009 tieto- ja tietoliikennetekniikan perustutkinnoksi, joka sisältää osaamisalat elektroniikka-asentaja ja ICT-asentaja. Sisällöltään ensimmäisen vuoden ammatilliset opinnot kuuluvat tutkinnon osaan elektroniikan ja ICT:n perustehtävät. Kyseisen tutkinnon osan eräs tavoite on, että opiskelija osaa käyttää mikrokontrollerin kehitysympäristöä (kääntää sekä ladata sen avulla ohjelmia) (1, s. 24).

Myllytullin yksikössä sulautetut järjestelmät ja mikrokontrollereiden ohjelmointi on kuulunut ja kuuluu edelleen jatkavien opiskelijoiden sulautettujen järjestelmien tutkinnon osaan. Tähän asti ohjelmoinnissa on käytetty pääasiallisesti C-kieltä. C-kielen oppiminen vaatii aikaa ja perehtymistä. Aloittavat opiskelijat toisaalta keskittyvät pääasiassa elektroniikan ja ICT:n perustietojen ja taitojen oppimiseen, joten mikrokontrollerin ohjelmointiin tarvitaan C-kieltä helpommin ja nopeammin alkuun päästävä tapa. Flowcode on eräs ratkaisu tähän. Graafisena vuokaaviopohjaisena ohjelmointiympäristönä se sopii monentasoisille opiskelijoille ehkä C-kieltä paremmin.

Tiimissäni oli asiaa jo pohdittu ja kokeilumielessä hankittu kaksi flowcoden aloituspakettia tarkoituksena perehtyä siihen. Aloittaessani opinnot nousi opinnäytetyönaiheen valinta esille ja sovimme, että minä teen opinnäytetyönä materiaalin opiskelijoille. Mietin tarkemmin opetuksen sisällön ja toteutuksen. Lisäksi täytyy varmistaa E-blocks-korttien kestävyys, koska meillä on useita aloittavia ryhmiä.

Yhtenä tämän työn tavoitteena onkin löytää menetelmiä ja opetustapoja, joiden avulla ensimmäisen vuoden opiskelija saadaan paremmin ymmärtämään mikrokontrollerin ohjelmoinnin ideaa ja sulautettujen järjestelmien periaatteita. Flowcoden avulla saadaan nopeasti näkyvää aikaiseksi lyhyen perehtymisen jälkeen. Tavoitteena on perehtyä ensin itse flowcodella ohjelmointiin ja sen jälkeen laatia oppimateriaali, jonka avulla opiskelija ymmärtää, mikä on sulautettu järjestelmä ja mikrokontrolleri. Oppimateriaalin pyrin laatimaan siten, että opiskelija pääsee sisälle flowcodella ohjelmoinnin perusteisiin opettajan ohjauksella tai peräti itsenäisesti ja saavuttaa tutkinnon osan tavoitteen tältä osin. Lisäksi testaan materiaalia ja harjoituksia oman ryhmäni opiskelijoilla.

2 YLEISTÄ SULAUTETUISTA JÄRJESTELMISTÄ

Sulautettu järjestelmä on tiettyyn kohteeseen suunniteltu laite, jota ohjaa laitteen sisällä oleva mikrotietokone. Sulautetun järjestelmän elektroniikkalaitteisto ja ohjelmisto, joka on ladattu mikrokontrollerin muistiin, eivät ole yleiskäyttöisiä vaan juuri kyseiseen tarkoitukseen suunniteltuja. Mikrokontrolleri (mikro-ohjain) voi olla vain kerran ohjelmoitavissa, jolloin ohjelma on ladattu kiinteään muistiin. Nykyisin kuitenkin ovat yleistyneet mikrokontrollerit, joita voidaan ohjelmoida uudelleen. (2, s. 7.)

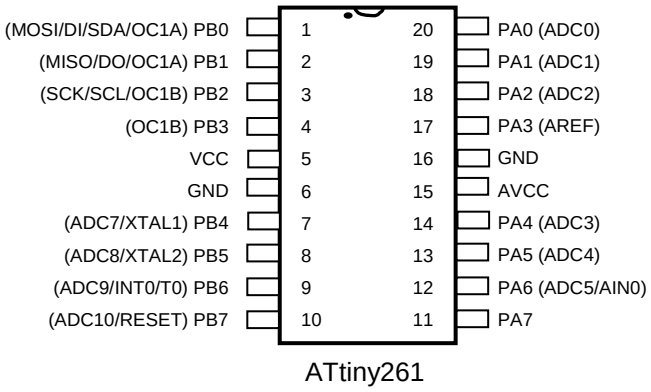
Esimerkkejä sulautetuista järjestelmistä on joka puolella ympäristöämme. Kaupassa punnitessamme hedelmiä käytämme sulautettua järjestelmää. Hedelmän valintanäppäimistö antaa mikrokontrollerille digitaalista tietoa, vaaka puolestaan analogista tietoa. Tässä tapauksessa mikrokontrollerissa on oltava analogia-digitaalimuunnin (A/D-muunnin), sillä kontrolleri ymmärtää vain digitaalista tietoa. Vaaka ja näppäimistö on kytketty kontrollerin tuloihin, ja kontrolleri käsittelee tietoa ohjelman vaatimalla tavalla. LCD-näyttö, joka kertoo hedelmien painon ja hinnan, on puolestaan kytketty mikrokontrollerin lähtöihin. (2, s. 8.)

Myös kotoa löytyy sulautettuja järjestelmiä. Television kaukosäädin on esimerkki paristolla toimivasta hyvin vähän tehoa vaativasta järjestelmästä. Kyseessä on yksinkertainen laite, jonka ohjaamiseen riittää yksi mikrokontrolleri. Kontrolleri on lepotilassa aina silloin, kun kaukosäädintä ei käytetä, ja näin saadaan minimoitua tehonkulutus. Kontrolleri ”herää”, kun jotain näppäintä painetaan, tutkii, mitä on painettu, ja lähettää näppäintä vastaavan koodin infrapunaledille. Pääsääntöisesti mikrokontrollereiden käyttöjännite on 5 V, mutta paristokäyttöisillä se on laajempi ja voikin vaihdella esimerkiksi 2,5 V:sta aina 6 V:iin saakka.(2, s. 9.)

Koska sulautettu järjestelmä on juuri tiettyyn tarkoitukseen suunniteltu, ovat siihen sisällytetyn mikrokontrollerin ominaisuudetkin kohteen mukaiset. Mikrokontrollereita ei siitä syystä suunnitella yleiskäyttöisiksi vaan aina tietyt ominaisuudet sisältäviksi. Vaikka jokaisessa mikrokontrollerissa on prosessori, muistia ja I/O-portteja, ei kaikissa ole esimerkiksi A/D-muunninta. Myös muistin, I/O-liitäntöjen ja muiden ominaisuuksien määrä vaihtelee tarpeen mukaan.

Edellä olevasta johtuen mikrokontrollereita on runsaasti erilaisia. Eri valmistajilla on omat tuoteperheensä. Esimerkkinä voidaan mainita muun muassa Intelin 51-perheen ohjaimet, Microchipin PIC-mikro-ohjaimet ja Atmelin AVR-mikro-ohjaimet. Kaiken kaikkiaan pelkästään Atmelilla on 438 erilaista mikrokontrolleria, joista AVR-tuoteperheeseen kuuluu noin 250 kontrolleria (3).

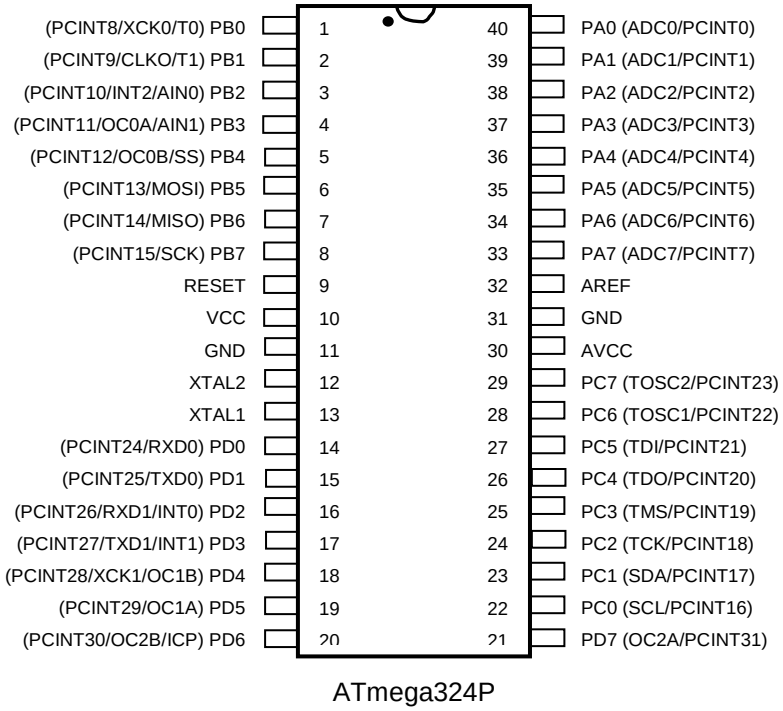
Atmelin runsaasta AVR-mikrokontrolleritarjonnasta löytyy eri kotelotyyppejä ja erilaisia ominaisuuksia. Yhteistä kaikille AVR-mikrokontrollerimalleille on samanlainen muisti- ja rekisterirakenne. Muistin ja muiden ominaisuuksien määrä riippuu mallista. Esimerkkinä vaikkapa 20-nastainen ATtiny216, jonka nastajärjestys on kuvassa 1. Attiny216:ssa on 16 I/O-linjaa. Se sisältää A/D-muuntimen, ja esimerkiksi flash-muistia sillä on 2 kilotavua. 20-nastaisena sitä on näppärä käyttää opetuksessa silloin, kun ei vaadita esimerkiksi enempää I/O-linjoja (4, s. 5).



KUVA 1. Attiny216:n nastajärjestys

Toisaalta Atmelilta löytyy myös 40-nastainen ATmega324P, jossa on neljä täyttä kahdeksan bittistä I/O-porttia eli 32 I/O-linjaa. Nastajärjestys on kuvattu seuraavan sivun kuvassa 2. Muita ominaisuuksia ja muistia löytyy tietysti edellistä mikrokontrolleria enemmän. Flash-muistia 324P:llä on jo 32 kilotavua. Eräs etu verrattuna ATtiny261:een on sekin, että vaikka siihen kytkisi ulkoisen kiteen, jäisi siihen silti neljä täyttä 8-bittistä I/O-porttia muuhun käyttöön (5, s. 5). ATtiny261:llä ulkoinen kide vie yhden I/O-linjan. Koska ATmega324P-mikrokontrollerista löytyy täydet neljä I/O-porttia, se sopii hyvin E-blocks- laitteiston kanssa käytettäväksi. Kuitenkaan 40-nastaisena sitä ei kannata käyttää

silloin, kun tehdään opiskelun yhteydessä pieniä harjoituslaitteita, joissa tarvitaan vain muutama I/O-linja käyttöön.



KUVA 2. Atmega324P:n nastajärjestys

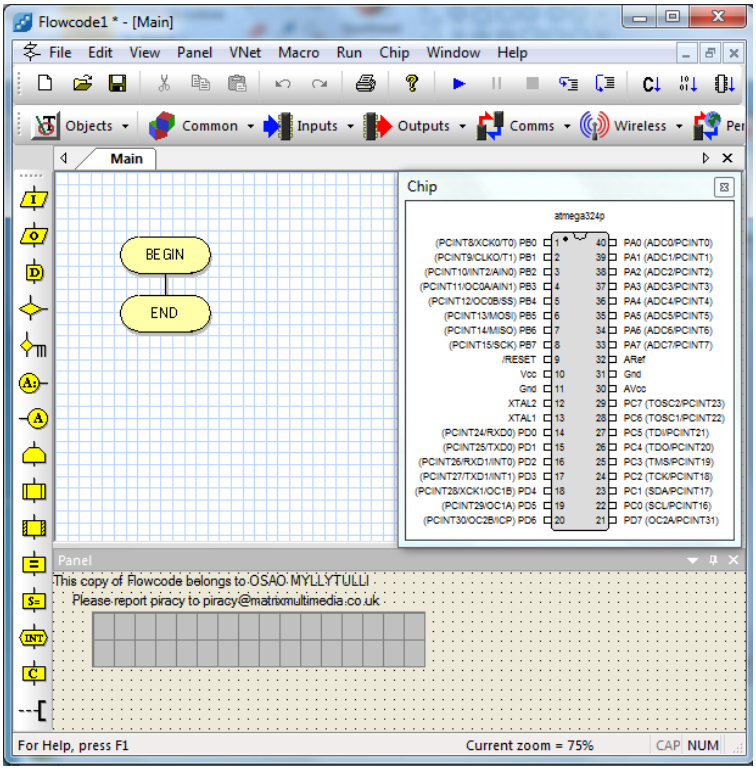
3 FLOWCODE

Mikrokontrollerit eivät ymmärrä suomea eivätkä edes englantia. Ne kykenevät käsittelemään ainoastaan ohjelmaa, joka muodostuu numeroista: ykkösistä ja nollista. Meidän taas on erittäin vaikea luoda ohjelmaa, joka koostuu pelkästään binäärikoodista tai heksadesimaalikoodista. Siksi tarvitsemme kääntäjää, joka kääntää osaamamme ohjelmointikielen mikrokontrollerin ymmärtämään muotoon. Ohjelmointikieliä on monentasoisia: symbolinen konekieli (assemby) on alemman tason, C-kieli on jo korkeamman tason ja flowcode on korkean tason ohjelmointikieli.

Flowcode on Matrix Multimedian erittäin kehittynyt korkean tason graafinen ohjelmointikieli mikrokontrollereille. Flowcoden etuna on se, että sen avulla pääsee nopeasti ja helposti kiinni mikrokontrollereiden ohjelmointiin. Lisäksi siitä löytyy ohjelmaversiot niin PIC- kuin Atmelin AVR tai ARM- mikrokontrollereiden ohjelmointiin. Laadittu ohjelma on myös helposti ladattavissa mikrokontrollerille.

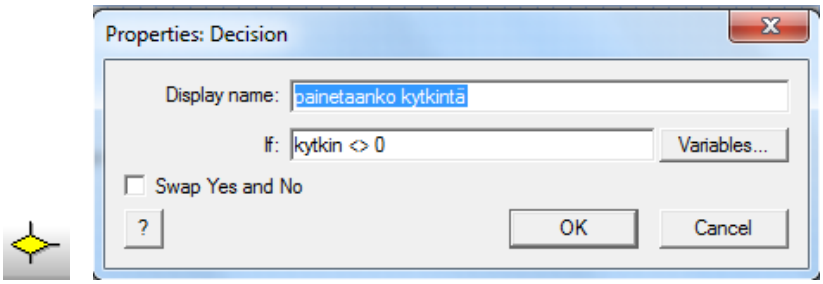
3.1 Perusnäkymä ja vakiokuvakkeet

Flowcoden helppous perustuu vakiokuvakkeisiin, jotka näkyvät seuraavalla sivulla olevan kuvan 3 vasemmassa reunassa. Vakiokuvakkeet eli käskyt yksinkertaisesti raahataan ja tiputetaan vuokaavio-ohjelmaan (6, s. 2). Vuokaavio-ohjelma rakennetaan vakiokuvakkeista kuvassa 3 näkyvien alku- ja loppukuvakkeiden väliin, ja se toteutetaan tietysti ylhäältä alaspäin. Eli ohjelmaa ei tarvitse kirjoittaa rivi riviltä, kuten esimerkiksi C-kielessä. Kuvassa näkyy myös chip-ikkunassa valittu mikrokontrolleri, joka tässä tapauksessa on ATmega324P. Näkymästä on helppo tarkistaa tarpeen niin vaatiessa mikrokontrollerin nastajärjestys ja nastojen mahdolliset erityistoiminnot ohjelman laatimisen aikana. Simuloinnin aikana siinä näkyy mikrokontrollerin käytössä olevien nastojen tilat.



KUVA 3. Flowcoden työtila

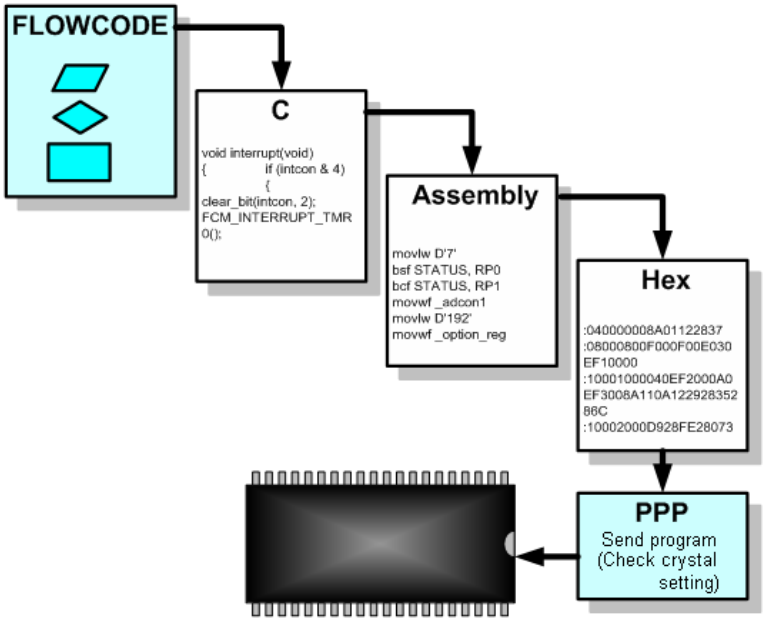
Kuvakkeiden käyttö on helppoa, sillä tuplaklikkaamalla kuvaketta päästään ominaisuudet-ikkunaan, jossa määritellään käskyyn liittyviä toimintoja. Esimerkki desicion eli päätös kuvakkeesta ja siihen liittyvästä ominaisuudet- ikkunasta on kuvassa 4. Päätöskuvakkeella valitaan kahden eri reitin välillä. Kyseessä on IF- eli jos-lauseke, jossa testataan, onko annettu ehto tosi. Alla olevassa kuvassa 4 ehtona on ”jos kytkin (muuttuja) on eri suuri kuin 0”. Erilaisia ehtolauseita voidaan muodostaa käyttämällä matemaattisia tai loogisia operaattoreita.



KUVA 4. Päätöskuvake ja sen ominaisuudet-ikkuna

3.2 Käännösprosessi ja komponentit

Kun vuokaavio-ohjelma on valmis ja se halutaan ladata mikrokontrollerille, prosessoi flowcode ohjelman useaan kertaan ennen mikrokontrollerille lähettämistä. Kuvassa 5 on kuvattu ohjelman käännösprosessi. Flowcode kääntää vuokaavio-ohjelman ensin C-kielelle, siitä assembleriksi ja lopulta konekielelle (heksadesimaaliluvuiksi). Konekieltä mikrokontrolleri ymmärtää. Konekieli eli heksadesimaaliluvut on se koodi, joka kontrollerille lähetetään ja poltetaan kontrollerin ohjelmamuistiin. (7.)



KUVA 5. Flowcoden käännösprosessi (7)

Eräs syy flowcodella ohjelmoinnin helppouteen on se, että siihen on ohjelmoitu usealle eri komponentille valmiiksi aliohjelmia eli makroja, joiden avulla komponentteja on helppo ohjata. Valmiit makrot saa käyttöön lisäämällä komponentin esimerkiksi input- tai output-valikosta. Nämä valikot näkyvät edellä olleessa kuvassa 3. Toiseksi, jos valitaan esimerkiksi output-valikosta LCD-näyttö ja liitetään se porttiin C ja tämän jälkeen itse ohjelmaan lisätään tietysti output-kuvake, saamme LCD-näytön ohjaukseen valmiit makrot. Lisäksi flowcode oh-

jelmoi mikrokontrollerin I/O-portin (tässä tapauksessa C) sen mukaan, onko valittu portille input- vai output-kuvake.

Flowcoden ja E-blocks-korttien avulla on siten helppo rakentaa ja simuloida erilaisia elektronisia järjestelmiä. E-blocks-kortteja on runsaasti alkaen yksinkertaisesta led-kortista aina ohjelmointikortteihin saakka. Se tukee myös LCD-näytön lisäksi monia eri komponentteja esimerkiksi ledejä, kytkimiä ja 7-segmenttinäyttöä. (6, s. 2.)

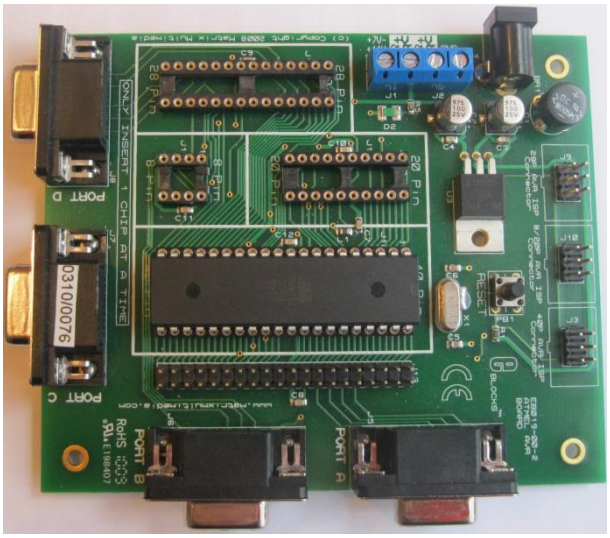
Flowcoden helppous, valmiit aliohjelmat ja moninaiset E-blocks-kortit luovat yhdessä kokonaisuuden, joka soveltuu erinomaisesti opetukseen. Se on joustava järjestelmä, jossa on helppo vaihtaa kortti toiseen. Sillä on nopea rakentaa erilaisia kokonaisuuksia ja Matrix Multimedialla on useita valmiita opetusratkaisuja tarjolla. E-blocks-kortit ovat kestäviä, ja korttien suojaksi on tarjolla valmis alusta ja mittojen mukaisia suojalevyjä. (8, s. 6 – 13.)

4 E-BLOCKS-KORTIT

E-blocks-kortit ovat piirilevyjä, joiden avulla voidaan rakentaa elektronisia järjestelmiä. Ne ovat yhteen liitettäviä elektroniikkalohkoja. E-blocks-kortit jaetaan kahteen ryhmään: upstream-kortteihin ja downstream-kortteihin. Upstream -kortit hallitsevat tiedonvälitystä, ja ne pystyvät määräämään väylässä tapahtuvan tiedonsiirron suunnan. Esimerkiksi AVR-mikrokontrollerin ohjelmointikortti on upstream-kortti. Downstream-kortit ovat upstream-kortin ohjauksessa, mutta tieto voi siirtyä molempiin suuntiin. Downstream-kortteja on hyvin monia erilaisia, esimerkkinä led-kortti, kytkinkortti ja LDC-näyttökortti. (8, s. 7.) Seuraavassa esitellään tarkemmin ne E-blocks-kortit, jotka ovat tällä hetkellä tiimini ja opiskelijoidemme käytössä.

4.1 AVR-ohjelmointikortti

Kuvassa 6 olevalla AVR-ohjelmointikortilla voidaan ohjelmoida 8, 20, 28 ja 40 nastaisia Atmelin AVR-mikrokontrollereita. Kortti sisältää neljä 8-bittistä I/O-porttia ja se on yhteensopiva flowcoden kanssa.



KUVA 6. AVR-ohjelmointikortti

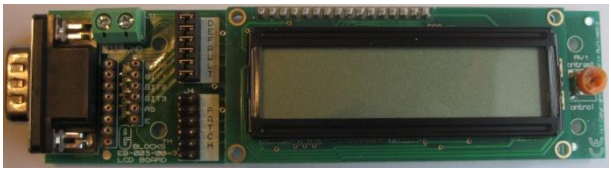
AVR-ohjelmointikortin käyttöjänniteliittimeen voidaan kytkeä 14 V:n reguloimaton jännite, koska kortilla on tasasuuntaaja ja 5 V:n regulaattori. D9-liittimet (PORT A – D) kytkeytyvät mikrokontrollerin 8-bittisiin I/O-portteihin. Niihin voidaan kytkeä muita E-blocks-kortteja kuten led-kortti. Erikokoisiin mikropiirin kantoihin asetetaan haluttu AVR-mikrokontrolleri ohjelmointia varten. Kortilla on liittimet AVRISP:n naarasliitintä varten ohjelman siirtämiseksi PC:ltä mikrokontrollerille. Ruuviliitimestä voidaan kytkeä käyttöjännite muille E-blocks-korteille esimerkiksi kytkinkortille. Painokytkin on kontrollerin resetointipainike. Kun kontrollerin resetoi, se aloittaa siihen ladatun ohjelman alusta. Lisäksi AVR-ohjelmointikortilla on irrotettava kide, joka ohjaa kontrollerin kelloa. (9, s. 3 – 4.)

Jotta AVR-ohjelmointikorttia voidaan käyttää, tarvitaan Atmelin kuvassa 7 näkyvä AVRISP-laite. Sillä voidaan ohjelmoida sekä Flashiä että EEPROMia. Se ei tarvitse erillistä virtalähdettä vaan laite saa käyttöjännitteen tietokoneelta USB-väylän kautta.

KUVA 7. AVRISP-laite

4.2 Downstream-kortit

LDC-näyttökortin näyttö (kuva 8) on kaksirivinen ja alfanumeerinen. Suomalaisia ääkkösiä ei voi tulostaa näytölle noin vain. Jos haluaa saada ne näkyviin, on perehdyttävä näyttökortin dokumenttiin ja tutustuttava siitä löytyvään merkkiluetteloon. Näyttö tarvitsee esimerkiksi AVR-ohjelmointikortilta tuodun käyttöjännitteen. Jännite kytketään ruuviliittimeen.(10, s. 5.)



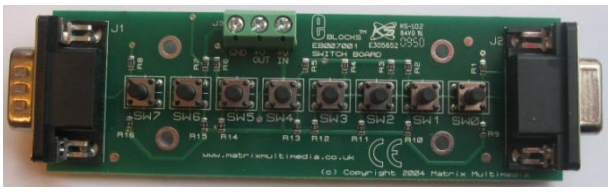
KUVA 8. LCD-näyttökortti

Näyttö on aina ohjelman aluksi alustettava ja kursori asetettava haluttuun kohtaan. Vasen yläkulma on 0,0. LCD-näytön käsittely tapahtuu flowcodessa komponettimakron avulla. LCD-kortilla on oma mikrokontrollerinsa, joka huolehtii LCD-näytön toiminnoista, joten flowcode-ohjelmoijan tarvitsee vain lähettää näyttökortille dataa tai käskyjä. Flowcode huolehtii muun muassa LCD:n ajuriasioista.(11.)

Led-kortissa, joka on kuvassa 9, on 8 lediä (D0-D7). Ledit on kytketty 560 Ω -etuvastuksen kautta D9-liittimen nastoihin. D0-ledi on kytketty nastaan 1, D1-ledi nastaan 2 ja niin edelleen. Ohjelmoitaessa D0:a vastaa vähiten merkitsevä bitti ja D7:ä eniten merkitsevä bitti. D9-liittimen nasta 9 on kytketty maihin. Kortti ei tarvitse erillistä jännitesyöttöä. (12, s.3 – 4.)

KUVA 9. Led-kortti

Kuvan 10 kytkinkortissa on 8 painokytkintä (SW0-SW7), jotka on kytketty käyttöjännitteestä 390 Ω -etuvastuksen kautta D9-liittimen nastoihin. SW0-kytkin on kytketty nastaan 1, SW1-kytkin nastaan 2 ja niin edelleen. Ohjelmoitaessa SW0-kytkintä vastaa vähiten merkitsevä bitti ja SW7-kytkintä eniten merkitsevä bitti. D9-liittimen nasta 9 on kytketty maihin. Kortti tarvitsee +5 V:n jännitteen muodostamaan loogisen tason 1. Jännite kytketään ruuviliittimeen esimerkiksi AVR-ohjelmointikortilta. (13, s. 3 – 4.)



KUVA 10. Kytkinkortti

5 OPPIMINEN JA OPETUKSEN SUUNNITTELU

Oppimista on tutkittu ja pohdittu paljon. Oppimiskäsityksiä on historian varrella kehittynyt ja kehitetty useita. Humanistinen oppimiskäsitys painottaa oppijan kokemusta oppimistapahtumasta ja oppijan omaa toimintaa. Behavioristit uskovat muuttumattomaan tietoon, joka voidaan siirtää sellaisenaan opettajalta oppijalle, ja konstruktivistinen käsitys näkee tiedon olevan subjektiivista riippuen yksilön aiemmista kokemuksista ja tietopohjasta. Oppimiskäsitysten lisäksi on löydetty ja jaoteltu erilaisia oppimistyylejä. Meistä jokaisella on oma yksilöllinen tyylimme oppia. Omaan oppimistyyliin vaikuttaa niin perinnölliset tekijät kuin vuosien varrella opitut opiskelutavatkin.

5.1 Oppimiskäsityksiä

Perinteinen opetus on perustunut behavioristiseen käsitykseen ihmisen oppimisesta. Tieto opetetaan pieninä palasina, ja riittävän suuresta määrästä tietopalasia muodostuu vähitellen kokonaisuus. Tavoitteena on saada oppijalta oikea vastaus opettajan kysymykseen Se, mitä tapahtuu oppijan pään sisällä, ei kiinnosta vaan oppija on pelkästään passiivinen tiedon vastaanottaja. Opetuksen tehtävänä on tarjota sellaisia virikkeitä, jotka ovat tavoitteiden mukaisia. (14, s. 148 – 151.)

Opetus ja koulutus suunnitellaan yksityiskohtaisesti. Opettajan tehtävänä on etukäteen miettiä, millaisiin palasiin opetettava kokonaisuus pienitään, ja siirtää palaset opiskelijalle. Opiskelijan tehtävänä on passiivisesti vastaanottaa tietoa. Lyhyesti sanottuna opettaja kaataa tietoa opiskelijalle, ja opiskelija vastaanottaa pureskelematta ja suoltaa sitten tarpeen vaatiessa esimerkiksi kokeessa saman tiedon muuttumattomana paperille. Oppiminen on behavioristisen näkemyksen mukaan faktojen pänttäämistä ja ulkoa opettelua. (14, s. 148 – 151.)

Humanistinen käsitys painottaa yksilön kasvua ja kehitystä, ja siihen se myös pyrkii. Kasvu on itsesäätöistä, eikä sitä voida ohjata ulkoapäin. Oppiminen on kokemuksellista ja luova prosessi, johon opettaja antaa puitteet mutta ei puutu itse tapahtumaan. Humanistit vaativat oppijalta paljon, koska luovuus, itseohjautuvuus ja itse itsensä kasvatus ovat vaikeita asioita. On paljon helpompaa olla

behavioristien passiivinen tiedon vastaanottaja. Opetusta humanistit eivät suunnittele kovin pitkälle vaan pohtivat oppimisprosessia. Opettaja tarjoaa mahdollisuudet ja puitteet oppijan kasvulle. (15, s. 28 – 31.)

Konstruktivistisessa oppimiskäsityksessä tieto nähdään toisenlaisena kuin perinteisessä käsityksessä. Tieto on muuttuvaa riippuen siitä, millainen aiempi tietopohja oppijalla on. Oppimiskäsitys lähtee siitä, että oppija ei ole tiedon passiivinen vastaanottaja vaan aktiivinen toimija. Aktiivisuus alkaa jo informaation valikoimisena. Hän ei ota kaikkea vastaan vaan yksilöllisesti valikoi ne havainnot ja tiedot, jotka läpäisevät huomion. (16, s. 37 – 43.)

Oppiminen on monivivahteinen tapahtuma, johon vaikuttaa oppijan aikaisemmat tiedot ja taidot eli se tietorakenne, jonka hän on jo muodostanut. Lisäksi oppijan älykkyys, persoonallisuus ja tausta ovat oppimisen kannalta merkityksellisiä. Oppimisympäristö, johon sisällytetään opettaja, opetus- ja arviointimenetelmät, oppiaine ja jopa opetussuunnitelma yhdessä oppijan henkilökohtaisten tekijöiden kanssa, muodostaa perustan oppimiselle. Oppijan havainnot ja tulkinnat johtavat oppimisprosessiin, jonka tuloksena on toivottavasti omien tavoitteiden ja uusien taitojen saavuttaminen. (16, s. 37 – 43.)

Keskeistä on merkitysten rakentaminen, jolloin asia täytyy ymmärtää. Oppijan omien aivojen käyttö ajatteluun ja merkitysten ja yhteyksien hakemiseen on olennaista. Ei voi oppia, jollei ymmärrä, mistä on kyse. Pelkkä ulkoa luku ei johda ymmärtämiseen, koska silloin muistissa on vain erillisiä tiedon palasia. Kun tieto saadaan nivottua aikaisemmin opittuun, on mahdollista ymmärtää ja siten todella oppia asia. Tietoa ei voi vain kopioida sellaisenaan muistiin ja tarpeen vaatiessa tulostaa aivoista, vaan uuden oppiminen vaatii tiedon konstruointia jo olemassa olevaan tietoon/tietorakenteeseen. (16, s. 37 – 43.)

5.2 Opetuksen suunnittelu

Miten eri oppimiskäsitykset vaikuttavat opetuksen suunnitteluun ja itse opetukseen? Behavioristit kirjoittivat opetussuunnitelman hyvin yksityiskohtaisesti painottaen tavoitteita ja tarkkoja sisältöjä. Opetus koostuu perinteisesti pienistä tietopalasista. Nämä faktat sitten opettaja siirtää esimerkiksi luennolla oppijoille.

Kun tietopalasia on oppijan päässä riittävästi, niistä oletetaan muodostuvan kokonaisuuksia.

Konstruktivistien opetussuunnitelmaan on merkitty olennaiset tavoitteet sekä ideat, joiden avulla saavutetaan opintokokonaisuuden osaaminen. Opettaminen vaatii opettajalta enemmän kuin perinteinen opetus. Hänen on kyettävä järjestämään opiskelijoille sellaiset puitteet ja tehtävät, jotka aktivoivat opiskelijoita. Tarkoituksena ei ole tiedon ulkoa opettelu vaan asian ymmärtäminen ja siten oppiminen. Tämä onnistuu, kun opettaja saa aikaan keskustelua ja herättää kysymyksiä, jotta opiskelijat voisivat rakentaa omia sisäisiä mallejaan asiasta. (14, s. 162 – 165.)

Konstruktivistisessa opetuksessa pyritään opiskelijoiden oppimisvalmiuksien parantamiseen ja oppimaan oppimiseen. Ajattelutaidon kehittäminen ja vastuun ottaminen omasta opiskelusta on tärkeää. Opettaja auttaa opiskelijaa tunnistamaan oman oppimistapansa ja kehittämään itselleen parhaiten sopivan tavan opiskella. Opettaja ja opiskelija toimivat yhteistyössä saavuttaakseen tavoitteen. (14, s. 162 – 165.)

Konstruktivistiseen oppimiskäsitykseen ja opetukseen tutustuessa vaikuttaa se erinomaiselta tavalta opettaa. Ensimmäinen ajatus on, että behavioristinen opetus täytyy heittää kokonaan romukoppaan ja unohtaa. Tarkemmin ajatellen huomaa, että perinteistäkin opetusta tarvitaan. Toisinaan asia on niin uusi, ettei sitä voi konstruoida mihinkään aiemmin opittuun. Tarvitaan ensin erillisten tiedonpalasten opettelua. Kokemalla oppimisessakin on hyvät puolensa, joita ei voi ohittaa. Olenkin tullut siihen tulokseen, että näitä eri oppimiskäsityksiin pohjautuvia opetusmenetelmiä on hyvä käyttää kaikkia sopivassa suhteessa. Mikä se oikea jako eri tapojen välillä on, riippuu ihan opiskelijoista, opetettavasta aiheessa, opettajasta ja tavoitteista.

Matrix Multimedian flowcode-ohjelmointikieli ja siihen kiinteästi liittyvät E-blocks -kortit mahdollistavat hyvinkin konstruktivistisen oppimisen. Mikäli oppijalla ei ole mitään käsitystä ohjelmoinnista tai mikrokontrollerista etukäteen, joutuu opettaja antamaan alkuun perinteikkääseen behavioristiseen tyyliin hiukan informaatiota sekä ohjelmoinnista että mikrokontrollerista. Opettaja voi myös oh-

jata oppijan kyseisen tiedon lähteelle, jolloin oppija voi heti alusta asti ryhtyä itsenäisesti keräämään tietoa ja rakentamaan itselleen käsitystä ohjelmoinnista ja mikrokontrollerista.

Koska vuokaavio-ohjelmoinnissa pääsee kohtalaisen nopeasti alkuun, se edesauttaa oppijan motivaatiota ja aiheeseen sisälle pääsemistä. Flowcodella on helppo simuloida ohjelman toimintaa ja simuloida esimerkiksi käsky kerrallaan. Näin eri vaihtoehtojen kokeileminen on yksinkertaista. Ei tarvitse esimerkiksi pelätä virheitä, koska ne löytyvät jo simulointivaiheessa. Oppija voi siten lisätä tietoaan ohjelmoinnista ja oppia lisää. Ohjelmointia oppii parhaiten ohjelmoimalla, ja flowcodella se onnistuu. Toisaalta visuaalisena ohjelmointikielenä se antaa ihan erilaisen näkymän ohjelmaan kuin vaikka C-kieli. Ohjelman eri vaiheiden ja käskyjen ymmärtäminen, joka on todellista oppimista, on ehkä vuokaavio-ohjelmassa yksinkertaisempaa kuin muissa ohjelmointikielissä.

6 TYÖN TOTEUTUS

Minulla on 12 vuoden kokemus opettamisesta ammatillisella toisella asteella. Lisäksi olen suorittanut ammatillisen opettajan pedagogiset opinnot ja ammatillisen erityisopettajan opinnot. Flowcode-ohjelmoinnin perusteiden oppimateriaalia ja harjoituksia laatiessani olen hyödyntänyt käsitystäni oppimisesta sekä kokemustani opettamisesta ammatillisella toisella asteella. Lisäksi opiskelijan tuntemukseni ja näkemykseni tämän hetken tieto- ja tietoliikennealan ensimmäisen vuoden opiskelijoiden tieto- ja taitotasosta sekä motivaatiosta ovat ohjanneet työtäni.

Aloittaessani tätä opinnäytetyötä en itse tiennyt flowcodesta kuin nimen. En myöskään ole itse opiskellut saati opettanut sulautettuja järjestelmiä. Jouduinkin tutustumaan aiheeseen alusta asti. Tutustumisen aloitin flowcodeen perehtymisellä ja kävin kahden päivän flowcoden peruskurssin. Kurssi auttoi minua saamaan vuokaavio-ohjelmoinnin juonesta kiinni ja pääsinkin sen avulla mukavasti alkuun.

Koska työn tavoitteena oli tuottaa materiaalia ja harjoituksia ensimmäisen vuoden opiskelijoiden ja tiimini käyttöön, jouduin ratkaisemaan materiaalin sisällön ja rakenteen siitä lähtökohdasta, ettei opiskelijoilla ole etukäteistietoa ohjelmoinnista tai sulautetuista järjestelmistä. Toisaalta tämän päivän opiskelijoilla on kokemusta tietokoneiden käytöstä, ja heille on opetettu esimerkiksi office-ohjelmien käyttöä. Heille ei siis tarvitse opettaa, miten ohjelmia avataan, minne tallennetaan tai miten tallennetaan. Lisäksi osalla opiskelijoista on aina alaan liittyvää harrastuneisuutta tai he ovat erityisen motivoituneita ja kykenevät etenemään keskiverto-opiskelijaa nopeammin ja pidemmälle. Myös heidät pyrin huomioimaan harjoituksia laatiessani.

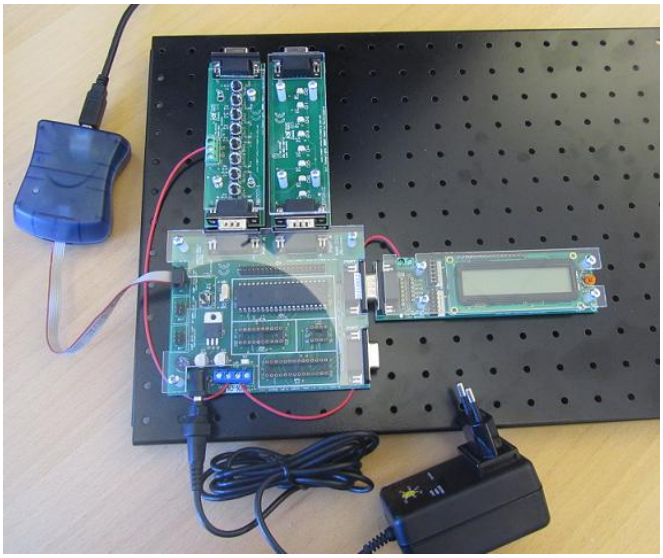
Vaikka materiaalin ja harjoitusten laatiminen olikin työn pääosassa, jouduin myös miettimään E-blocks-korttien säilytystä. Meillä on aloittavia opiskelijoita 5 ryhmää ja tarkoituksena on, että ensimmäiset harjoitukset, jotka toteutetaan E-blocks-korteilla, tekevät kaikki opiskelijat. Korttien kestävyys on siitä syystä varmistettava. Lisäksi jouduin miettimään, miten pitemmälle ehtivät opiskelijat saavat ladattua ohjelmansa mikrokontrollerille.

6.1 Flowcode-lisenssit ja E-blocks-laitteisto

Oppilaitosympäristössä on aina taloudelliset resurssit rajoittavana tekijänä. Tiimini kävikin keskustelua koulutusjohtajan kanssa flowcode-ohjelmalisenssien ja E-blocks-korttien määrästä. Valmiiksi meillä oli kaksi flowcode- ohjelmalisenssiä ja kaksi AVR-ohjelmointi-, led- ja kytkinkorttia ja kaksi LCD-näyttökorttia. Saimme luvan ja hankimme 50 ohjelmalisenssiä sekä kuusi kappaletta edellä mainittuja E-blocks-kortteja lisää.

Tässä vaiheessa mietin ja keskustelin myös tiimissäni, mikä mikrokontrolleri kannattaa valita harjoituksiin. Muutamalla opettajalla oli jo valmiita harjoituksia ATtiny2313-kontrollerille. Sitä emme kuitenkaan valinneet käytettäväksi ohjelmointiharjoituksiin, koska AVR-ohjelmointikortilla ei voi sitä ohjelmoida. Nastajärjestys poikkeaa muista 20-nastaisista AVR-mikrokontrollereista. AVR-ohjelmointikortin mukana tulee ATmega324P, jonka nastajärjestys on kuvattu kuvassa 2. Siinä on 32 I/O-linjaa eli neljä täyttä I/O-porttia. Siitä syystä se on järkevä valinta E-blocks-laitteistoon käytettäväksi.

Koska korttien kestävyys täytyi myös varmistaa, hankin E-blocks-korteille tarkoitetun metallisen alustan ja mittojen mukaiset suojalevyt korttien päälle. Kuusi korttisarjaa kytkin kiinteästi siten, että portissa A on kytkinkortti, portissa B ledkortti ja portissa C on LCD-näyttökortti. Portti D jäi näin ollen vapaaksi odottamaan mahdollista lisäkorttia. Kiinteästi kytketty E-blocks-laitteisto näkyy kuvassa 11. Kaksi AVR-ohjelmointikorttia jäi vapaaksi käytettäväksi esimerkiksi eri mikrokontrollerien lataamiseen.



KUVA 11. Laitteisto

Koska osa opiskelijoista ehtii pidemmälle ja heille on suunniteltu harjoituksia, joissa rakennetaan verolevylle elektroniikkaa mikrokontrollerin ympärille, tarvitsemme enemmän kuin kaksi mikrokontrollerin latauskorttia. Liitteessä 1 on latauskortin piirikaavio ja kuva valmiista latauskortista. Kortissa on 40-jalkainen ZIF-kanta. Kortti on suunniteltu siten, että sillä voidaan ladata 28-nastainen ATmega168 tai vastaavalla jalkajärjestyksellä oleva mikrokontrolleri, kun kontrolleri sijoitetaan alkaen nastasta yksi. Lisäksi sillä voidaan ladata 20-nastainen Attiny2313-mikrokontrolleri, kun se sijoitetaan alkaen nastasta 40. Tämä mahdollistaa eri mikrokontrollerien käytön harjoituksissa. Jatkossa voimme helposti rakentaa lisää latauskortteja esimerkiksi niin, että kukin opiskelija tekee verolevylle oman latauskorttinsa. Se on helppo tehdä tarvitsemalleen mikrokontrollerille.

6.2 Oppimateriaalin teoriaosuus

Pohtiessani oppimateriaalini teoriaosuuden laajuutta, jouduin ottamaan huomioon sekä opetussuunnitelman tavoitteet että ajan rajallisuuden. Pääpaino ensimmäisen vuoden opiskelijoiden opetuksessa on opettaa tarpeelliset elektroniikan ja ICT:n perustiedot ja taidot, jotta heillä on edellytykset syventää osaamistaan seuraavina vuosina. Varsinainen sulautettujen järjestelmien opetus on toisen vuoden tutkinnon osa. Ensimmäisenä vuonna on tältä osin tavoitteena saada käsitys mikrokontrollerista, sen ohjelmoinnista ja siitä, mikä sulautettu järjes-

telmä on. Erityisesti opetussuunnitelman tavoitteen mikrokontrollerin kehitysympäristön käytöstä ja ohjelman latauksesta kontrollerille täytyy toteutua. Keskustelimme tiimissäni, kuinka paljon aikaa on mahdollista tähän varata, ja totesimme, että voidaan käyttää noin yksi opintoviikko. Tämä tarkoittaa lähiopetustunteina noin 30 tuntia.

Oppimateriaalissani on lyhyesti kerrattu lukujärjestelmät siltä osin, kuin ne on tarpeellista osata ohjelmoitaessa flowcodella. Lähinnä muistutan mieleen binäärijärjestelmän ja hekadesimaalijärjestelmän ja kuinka binääriluku muutetaan heksaluvuksi. Tästä on apua I/O-porttien ohjauksessa, sillä niiden ohjaamiseen on järkevää käyttää joko heksadesimaalilukuja tai binäärilukuja. Varsinaisesti niitä ei opeteta, koska lukujärjestelmät opiskellaan Elektroniikka 2 -osajaksolla heti jakson alussa. Mikrokontrollerin ohjelmointia opiskellaan vasta myöhemmin Elektroniikka 2 -osajaksolla.

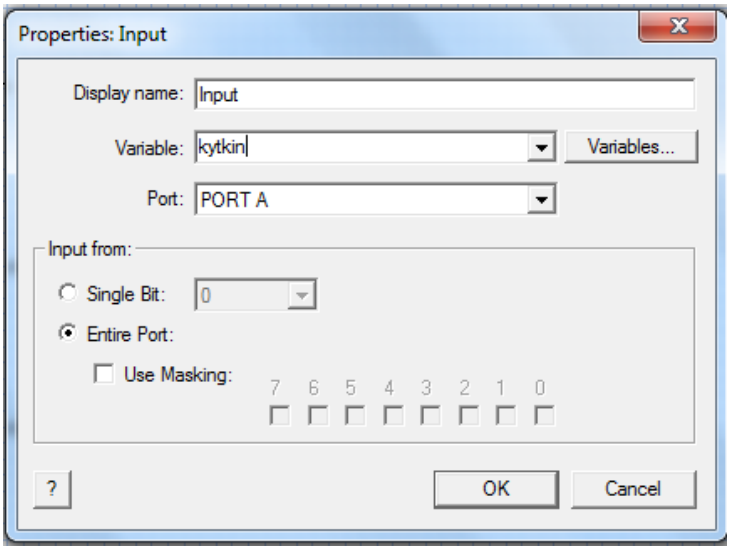
Koska tämän päivän tieto- ja tietoliikenne alan opiskelijat (ainakin meillä) eivät jaksa keskittyä lukemaan pitkää tekstiä, olen lukujärjestelmäkertauksen mahduttanut yhdelle sivulle. Samoin olen sivun verran kertonut lukujärjestelmien jälkeen, mikä on sulautettu järjestelmä. Sulautetusta järjestelmästä kerron samaan tapaan kuin kappaleessa 2. Myös mikrokontrollerista kerron sivun verran. Esimerkkinä käytin kuvassa 1 olevaa ATtiny261-mikrokontrolleria. Materiaalissa on kuva nastajärjestyksestä ja hiukan selvitetty liitännöistä ja muisteista. Esimerkiksi rekisterit jätän suosiolla seuraavalle vuodelle.

Vuokaavio-ohjelmoinnista kerrotaan tietysti materiaalissa hiukan enemmän. Kaikki vakiokuvakkeet eli käskyt esitellään lyhyesti ja osassa annetaan myös esimerkki ominaisuudet ikkunasta.

Esimerkki vuokaaviokuvakkeen esittelystä

Input- ja output-käskyt ovat samankaltaiset. Input (tulo) tarkastaa kontrollerin jonkin portin tai yksittäisen nastan (bitin) tilan. Input tallentaa tilan muuttujaan (variable), joka on määritelty. Output (lähtö) asettaa kontrollerin portin/nastat muuttujan määräämään tilaan. Outputille voidaan antaa lukuarvo

myös suoraan ilman muuttujaa, input tarvitsee aina muuttujan. Kuvassa 12 on tulokäskyn ominaisuudet ikkuna.



Properties: Input eli Ominaisuudet: tulo –ikkuna

Tässä muuttuja nimeltä **kytkin**. I/O portti A. Lukee koko portin 8 bitin tilan.

Voisi myös valita yksittäisen bitin tai käyttää maskia, jolla valitaan halutut bitit.

KUVA 12. Tulokäskyn ominaisuudet ikkuna

6.3 Sanalliset tiedonhakutehtävät

Konstruktivistinen oppimiskäsitys pitää sisällään oppimaan oppimisen ja oppijan oman toiminnan oppimisprosessissa. Siitä syystä kaikkia asioita ei ole materiaalissa behavioristisesti valmiiksi annettuna. Toisaalta alamme opiskelijat tänä päivänä eivät jaksa lukea tai kuunnella pitkiä teoriaselostuksia. Näistä syistä materiaalissa on tiedonhakutehtäviä, joiden avulla pyritään aktivoimaan opiskelijaa itsenäiseen tiedonhakuun ja oman ajattelun kehittämiseen. Tiedonhakutehtävät tietysti myös lisäävät opiskelijan tietoa aiheesta eli mikrokontrollerista ja ohjelmoinnista. Tiedonhakutehtävät mahdollistavat myös, että motivoituneet ja aiheesta kiinnostuneet opiskelijat voivat laajentaa tietämystään esimerkiksi sulautetusta järjestelmästä ja opiskella asiaa enemmänkin. He mahdollisesti pääsevät jo vähän pidemmälle ohjelmoinnissa ja syvemmälle mikrokontrollerin toimintaan, kuin mikä on ensimmäisen vuoden perusvaatimus.

Mikrokontrolleriin liittyviä selvitettäviä asioita ovat esimerkit:

1. Selvitä esimerkiksi internettiä apuna käyttäen, mistä sanoista tulevat lyhenteet ja millaisia muisteja ovat: EEPROM, SRAM, FLASH-muisti.
2. Selvitä käsitteet kello-oskillaattori, reset, osoiteväylä, dataväylä, ohjausväylä.

Ohjelmointiin liittyvät esimerkit:

1. Selvitä, mitä tarkoittaa muuttuja (variable) ohjelmoinnissa.
2. Selvitä, mitä alla olevat muuttujatyypit tarkoittavat / miten eroavat toisistaan: BYTE, INT, STRING ja FLOATING POINT.

Tässä on vain pari sanallista tehtävää. Oppimateriaalissa on lisää tehtäviä, ja aiheen opettajan on helppo laajentaa opetusta tai tehtävänantoa tarpeen niin vaatiessa ja ajan antaessa myöten.

6.4 Oppimateriaalin harjoitukset

Ohjelmoimaan oppii ohjelmoimalla eli ihan itse tekemällä. Siksi tein oppimateriaaliin 10 ohjelmointiharjoitusta valmiiksi. Ensimmäiset kuusi harjoitusta ovat puhtaasti ohjelmointiharjoituksia, jotka voidaan testata tietysti simuloimalla mutta myös E-blocks-laitteiston avulla. Ohjelmointiharjoitukset alkavat hyvin helposta ledivilkusta, joka on yksityiskohtaisesti neuvottu kuvien avulla. Harjoitus 1 on liitteessä 1. Samasta harjoituksesta on jatkotehtävä 1B, jossa joutuu itse miettimään, miten eri ledit saa vilkkumaan erilaisella kombinaatiolla.

Ohjelmointiharjoitukset olen suunnitellut siten, että ensimmäisessä on mukana ainoastaan led-kortti ja kuvakkeista output ja viive. Ohjelmointiharjoituksessa 2 on mukana muuttuja ja muuttujan käsittelyä (lasketaan, montako kertaa) sekä päätöskuvake. Tässä harjoituksessa on osittain ohje mukana. Seuraavaan tulee mukaan input ja kytkinkortti, ja opiskelijan pitäisi jo käyttää tähän asti opittua sekä osata jo hiukan soveltaa aiemmin oppimaansa. Lisäksi hänen pitäisi pikkuhiljaa yhdistää uutta tietoa aiempaan. Harjoituksessa 5 tutustutaan LCD-näytön ohjaukseen ja siihen liittyviin makroihin, ja harjoituksessa 6 itse kokeillaan LCD-näytön ohjausta valmiiden makrojen avulla.

Ensimmäisen kuuden harjoituksen jälkeen alkavat harjoitukset, joissa pitää jo vähän kytkeä elektroniikkaa mikrokontrollerin ympärille. Ensin on koekytkentäalustalle hyvin helppoja tehtäviä, joissa ei ehkä niinkään tule uutta ohjelmointiasiaa vaan tutustumista mikrokontrollerin datasheettiin ja kytkennän miettimistä. Viimeinen valmiiksi mietitty (sellaiselle opiskelijalle, joka niin pitkälle ehtii) on niin sanottu Munakello-harjoitus. Harjoituksen kytkentäkuva ja ohje ovat liitteessä 3. Aiheen opettaja voi antaa mielikuvituksensa lentää ja opetettavan ryhmän mukaan lisätä joko haastavampia tai mahdollisesti helpompia harjoituksia, jos aikaa on.

6.5 Materiaalin kokeilu

Aloitin materiaalin testauksen ryhmäni kanssa. Ryhmässäni on yksi hiukan vanhempi opiskelija ja loput suoraan peruskoulusta tulleita. Kokeilin materiaalia siten, että annoin opiskelijoiden perehtyä omatoimisesti teoriaosuuteen ja harjoituksiin. Ohjelmointiharjoitusten aikana neuvoin ja havainnoin opiskelijoiden toimintaa. Heti alusta asti oli nähtävissä, että nuoret eivät oikein jaksa keskittyä teoriaan, vaan he aloittivat lähes suoraan sanallista tiedonhakutehtävistä, jotka he tekivätkin tunnollisesti. Koska he eivät jaksa lukea teoriaa, pitää se käydä nuorten kanssa aika opettajajohtoisesti läpi.

Ohjelmointiharjoituksissa oli myös nähtävissä nuorten halu päästä helpolla. Ensimmäinen tehtävä onnistui lähes kaikilla omatoimisesti, koska se oli niin perusteellisesti ohjeistettu. Seuraavissa harjoituksissa osa kaipasikin apua. Aikaa nuorten kanssa kului sen verran, ettemme ehtineet viittä ensimmäistä harjoitusta pidemmälle. Joka tapauksessa jokainen tutustui mikrokontrollerin ohjelmointiin ja latasi ohjelman kontrollerille. Eli tältä osin opetussuunnitelman tavoite täyttyi.

Seuratessani oman ryhmäni toimintaa täydensin materiaalini flowcode -esittelyosuutta. Oppimateriaalin toisen version otti käyttöön aikuisryhmän opettaja. Koska ryhmä koostui huomattavasti omatoimisemmista opiskelijoista, saivat he tehtyä harjoitukset nopeassa tahdissa. He pääsivät jo ihan oikeasti kiinni flowcodella ohjelmointiin ja tekivät myös muun muassa Munakello-harjoituksen.

Tämä johtui osittain siitä, että he olivat nopeampia, motivoituneempia ja heillä oli myös enemmän aikaa.

Kysyin omalta ryhmältäni kommentteja materiaalista. Vastauksia sain laidasta laitaan. Eräs kirjoitti ”*Hyvää materiaalia oli ja aika selvää sillee että sitä osas tehä*”. Hänellä riitti todennäköisesti kiinnostusta ja motivaatiota miettiä itsekin, miten ohjelmoinnin toteuttaa. Toinen taas kirjoitti ”*Heikot ohjeet joihinkin tehtäviin*”. Hän olisi kaivannut tarkempia ohjeita, kun minä taas pyrin siihen, että opiskelija alun ohjauksen jälkeen ratkaisisi ohjelmointitehtävät itse ja näin oppisi uutta.

7 YHTEENVETO

Työn tavoitteena oli saada luotua materiaali, jonka avulla opiskelijamme tutustuvat vuokaavio-ohjelmointiin, mikrokontrolleriin ja ymmärtäisivät, mikä on sulautettu järjestelmä. Lisäksi teoriapakettia ja harjoituksia oli tarkoitus kokeilla ryhmäni opiskelijoiden kanssa. Kun tavoitteiden saavuttamista tarkastelee tällä tasolla, voin sanoa saavuttaneeni osan tavoitteista.

Materiaalin valmistaminen vei yllättävän paljon aikaa. Teorian rajaaminen ja sen pohtiminen, kuinka syvällisesti tai pinnallisesti minkäkin asian selittää kirjallisessa materiaalissa, oli todella vaikeaa. Aikaa on vain vähän, joten mikrokontrollerin sielunelämän tutkimiseen ei voi mennä. Toisaalta, voiko kontrolleria ohjelmoida ja sen toimintaa ymmärtää, jos ei tunne sen sielunelämää? Tässä kohtaa palasin opetussuunnitelman perusteiden pariin ja totesin, ettei siellä vaadita mikrokontrollerin toiminnan ymmärtämistä. Siitä syystä rajasin teorian todella minimiin. Jatko-opinnoissa sulautettujen järjestelmien tutkinnon osassa opiskelijat pääsevät perehtymään aiheeseen syvällisemmin.

Flowcoden asentaminen tietokoneille sujui hyvin, mutta oikean AVRISP-ajurin löytyminen vei aikaa ja hermoja. Jostain syystä flowcoden mukana tulleessa asennuslevyssä eikä Atmelin ja Matrix Multimedian sivuilta löytyvä ajuri toiminut oppilaitoksemme kannettavissa eikä pöytätietokoneissa. Myöskään ATK-tuesta ei tämän selvittämiseen löytynyt aikaa. Onneksi meillä on aikuisopiskelijaryhmä, josta löytyi aiheeseen kiinnostunut opiskelija etsimään oikeaa ajuria. Hän löysi Mighty Ohmin sivulta sopivan ajurin, ja pääsin jatkamaan työtäni eteenpäin. Mikäli ajuria ei olisi ajoissa löytynyt, olisimme joutuneet tyytymään ainoastaan simuloimaan ohjelmia.

Olen tyytyväinen materiaaliini pääosin. Kollegani, joka myös käytti materiaalia, oli siihen tyytyväinen. Hänen kanssaan olin jo pitkin matkaa keskustellut harjoituksista ja siitä, millaisia harjoituksia materiaaliin liitetään. Hänen ryhmänsä, joka ehti tehdä myös Munakello-harjoituksen, ymmärtää sulautetuista järjestelmistä ja mikrokontrollerista huomattavasti enemmän kuin oma ryhmäni. Tämä kuvastaa tietysti opiskelijan oman toiminnan merkitystä ja sitä, kuinka tärkeää on miettiä opetus aina ryhmän mukaan. Materiaalini toimii pienellä ohjauksella

itsenäiseen opiskeluun kykenevällä opiskelijalla. Nuorella, joka vaatii ulkoapäin tulevaa motivointia ja ohjausta, se toimii oppimisen tukena, kunhan opettaja ohjaa riittävän paljon.

Ensi lukuvuonna pyrimme siihen, että kaikki luokat tutustuvat flowcode-ohjelmointiin ja tekevät ainakin materiaalin ensimmäiset harjoitukset. Materiaalia ja harjoituksia kehitetään tietysti eteenpäin. Uusia harjoituksia tulee tiimin eri opettajilta ja niitä voi siihen lisätä. Kun nyt tekemääni oppimateriaalia käytetään eri opiskelijaryhmien kanssa, huomataan, mitä tietoa tarvitaan lisää. Koska materiaali on sähköisessä muodossa, onnistuu sen täydentäminen ja kehittäminen helposti.

Vaikka olen oppimateriaalini pääosin tyytyväinen, eivät kaikki tavoitteet toteutuneet oman ryhmäni kanssa. Ymmärrys sulautetusta järjestelmästä ja mikrokontrollerista jäi heikoksi. Jatkossa tähän täytyy käyttää enemmän aikaa ja se onnistuukin ensi lukuvuonna, koska aiheen käsittely voidaan aloittaa aiemmin. Lisäksi sulautetun järjestelmän ja mikrokontrollerin ymmärtämiseksi on opettajan käytettävä aikaa ja kerrottava aiheesta enemmän, kuin materiaalissa on. Kaikkea on turha liittääkään oppimateriaaliin. Jatkoa ajatellen kannattaisi laatia esimerkiksi powerpoint-esitys, josta olisi helppo ottaa aina pätkä sopivassa kohdassa.

LÄHTEET

1. Opetushallitus. Tieto- ja tietoliikennetekniikan perustutkinto 2009. 2009. Vaasa: Oy Fram Ab.

2. Koskinen, J. 2004. Mikrotietokonetekniikka. Sulautetut järjestelmät. Keuruu: Otavan Kirjapaino oy.

3. Atmel. Home > Products > Microcontrollers > Atmel AVR 8- and 32-bit. Saatavissa: http://www.atmel.com/products/microcontrollers/avr/default.aspx. Hakupäivä: 26.4.2012.

4. Atmel. 2010. 8-bit Microcontroller with 2/4/8K Bytes In-System Programmable Flash. ATtiny261/V, ATtiny461/V, ATtiny861/V. Summary. Saatavissa: http://www.atmel.com/Images/2588S.pdf. Hakupäivä 20.3.2012.

5. Atmel. 2010. 8-bit Microcontroller with 16K/32K/64K Bytes In-System Programmable Flash. ATmega164P/V, ATmega324P/V, ATmega644P/V. Saatavissa: http://www.atmel.com/Images/doc8011.pdf. Hakupäivä 20.2.2012.

6. Matrix. Flowcode. No coding, no limits. Saatavissa: http://www.matrixmultimedia.com/datasheets/TEFLC-60-4.pdf. Hakupäivä 25.1.2012.

7. Matrix. 2011. Introduction to Microcontroller Programming. Välilehti The flowcode process. Saatavissa: http://www.matrixmultimedia.com/courses/itm/index.php?n=Programming. TheFlowcodeProcess. Hakupäivä 12.1.2012.

8. Matrix. 2012. Computer science and electronics teaching resources. Saatavissa: http://www.matrixmultimedia.com/resources/files/misc/New-E-blocks-catalogue-small-file-size.pdf. Hakupäivä 3.3.2012.

9. Atmel AVR® multiprogrammer system EB194-00-2. 2008. Matrix Multimedia. Saatavissa:

http://www.matrixmultimedia.com/resources/files/datasheets/EB194-30-2.pdf. Hakupäivä 12.1.2012.

10. Matrix Multimedia. 2005. E-Blocks. LCD board datasheet EB005-00-3. Saatavissa: http://www.matrixmultimedia.com/resources/files/datasheets/EB005-30-3.pdf. Hakupäivä 12.1.2012.

11. Matrix. 2011. Introduction to Microcontroller Programming. Välilehti LCD Board EB005. Saatavissa: http://www.matrixmultimedia.com/courses/itm/index.php?n=EblocksBoards.LCDBoardEB005. Hakupäivä 12.1.2012.

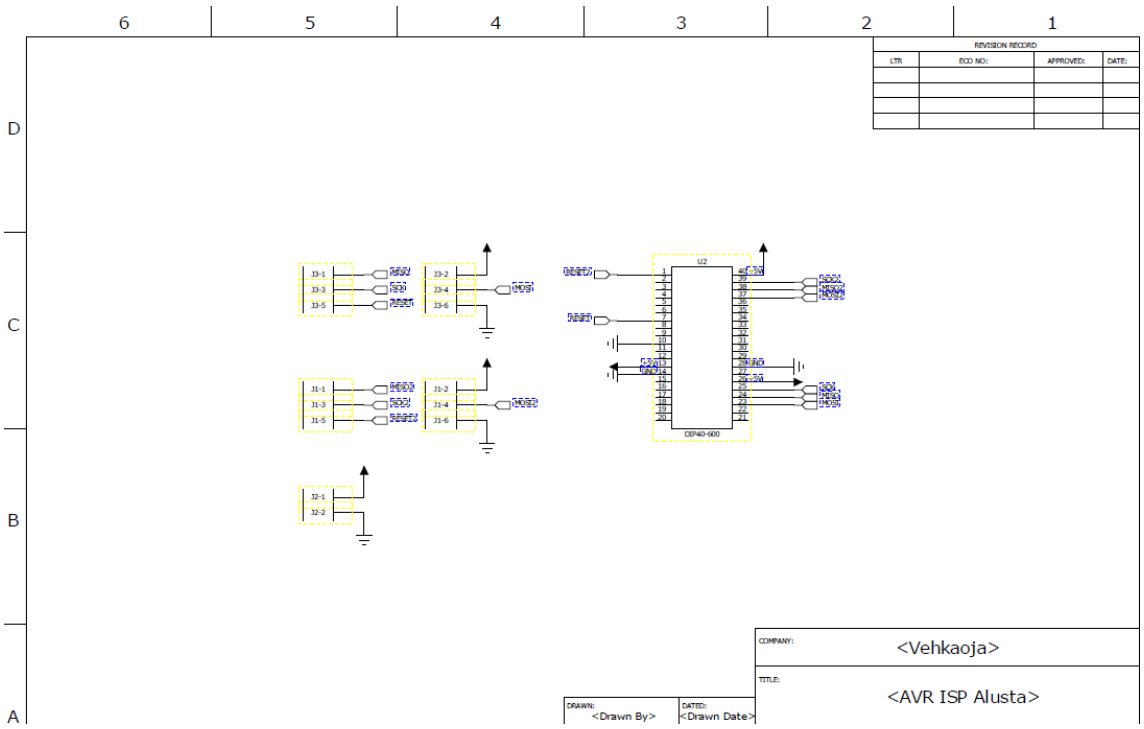
12. Matrix Multimedia. 2005. E-Blocks. LED board datasheet EB004-00-2. Saatavissa: http://www.matrixmultimedia.com/resources/files/datasheets/EB004-30-2.pdf. Hakupäivä 12.1.2012.

13. Matrix Multimedia. 2005. E-Blocks. Switch board datasheet EB007-00-1. Saatavissa: http://www.matrixmultimedia.com/resources/files/datasheets/EB007-30-1.pdf. Hakupäivä 12.1.2012.

14. Rauste-Von Wright, Maijaliisa - Von Wright, Johan - Soini, Tiina 2003. Oppiminen ja koulutus. Juva: WS Bookwell Oy.

15. Kauppila, Reijo A. 2007. Ihmisen tapa oppia. Juva: WS Bookwell Oy.

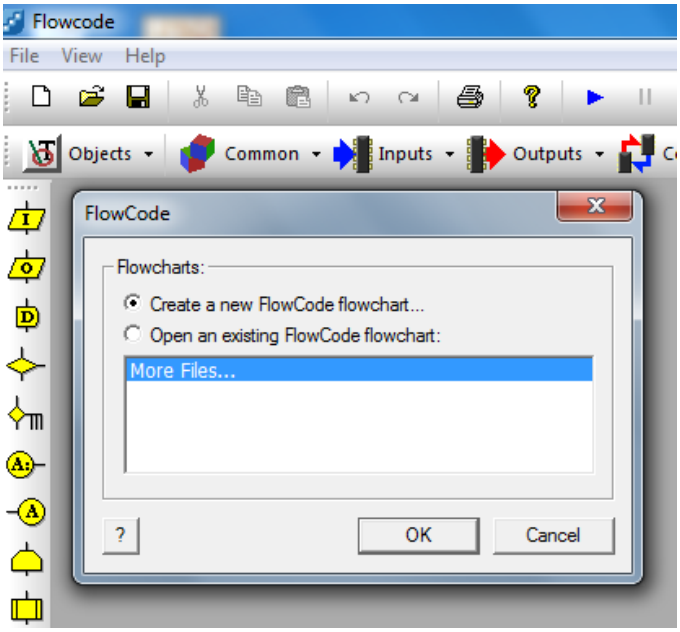
16. Tynjälä, Päivi 1999. Oppiminen tiedon rakentamisena. Konstruktivistisen oppimiskäsityksen perusteita. Tampere: Tammer-Paino Oy.

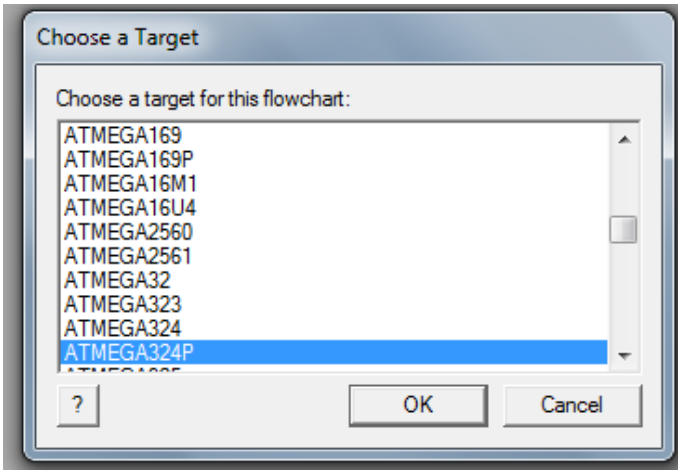

6
5
4
3
2
1
D
C
B
A
REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
U2
DIP40-600
COMPANY:
<Vehkaoja>
TITLE:
<AVR ISP Alusta>
DRAWN:
<Drawn By>
DATED:
<Drawn Date>

Harjoitus 1

Ohjelma sytyttää vuorotellen ledikortin 4 ensimmäistä ja 4 viimeistä lediä.

Käynnistä flowcode ja aloita uusi vuokaavio.



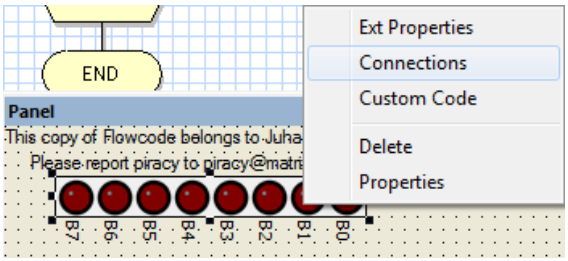


Sinulle avautuu mikrokontrollerin valintaikkuna.

Valitse ATMEGA324P.

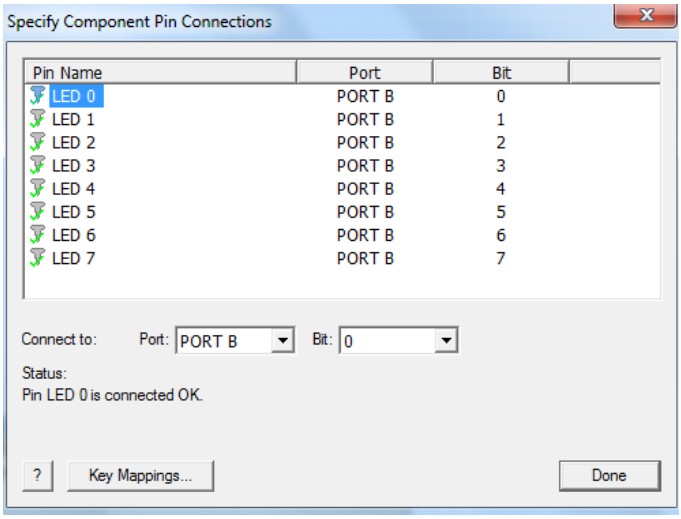
Käytämme sitä myös seuraavissa harjoituksissa.

Koska tässä harjoituksessa käytämme ledikorttia, kannattaa se valita jo heti alussa. Hae ledikortti **Outputs-> ledarray** Ledien pitäisi tulla näkyviin työpöydän alareunassa olevaan paneeliin.



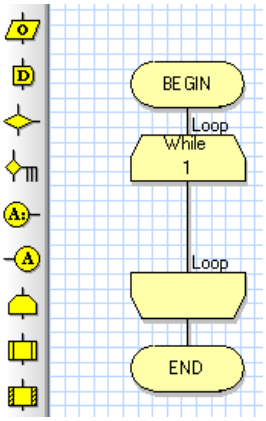
Ledirivin päällä klikkaa oikea hiiren näppäintä ja valitse **connections**.

HUOM! Ext Properties kautta voit vaikkapa vaihtaa ledien väriä ja kokoa.



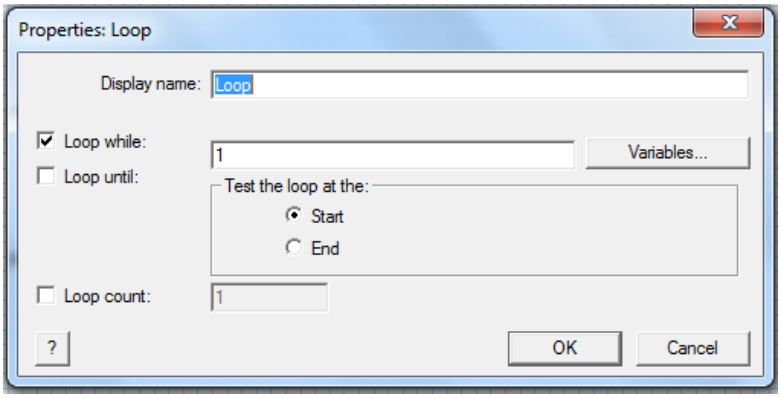
Ledikortti on fyysisestikin kytketty porttiin B, joten varmista myös ohjelmassa, että ledit on kytketty porttiin B.

Varmista, että kaikki 8 lediä on kytketty portin B bitteihin.



Raahaa silmukkakuvake (loop) alun (begin) ja lopun (end) väliin. Jos emme käyttäisi silmukkaa, ledin välähtäisivät vain kerran.

Tuplaklikkaa loop –kuvaketta niin sinulle aukeaa **properies: loop** (Ominaisuudet: silmukka) ikkuna. Tässä tapauksessa sitä ei tarvitse muuttaa.

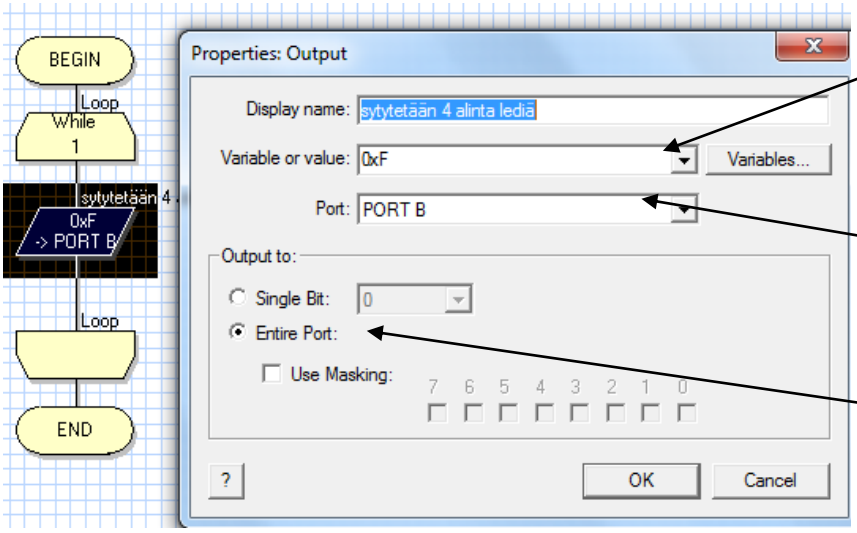


Loop while **1** tarkoittaa ikuista silmukkaa.

Tekstikenttään voisi myös kirjoittaa ehdon.

Loop count valinnalla voi määrätä montako kertaa silmukka toistuu.

Raahaa output kuvake silmukan väliin. Tuplaklikkaa sitä, niin sinulle aukeaa properties: Output (ominaisuudet: lähtö) -ikkuna. Display name kannattaa olla kuvaileva kuten alla olevassa kuvassa.



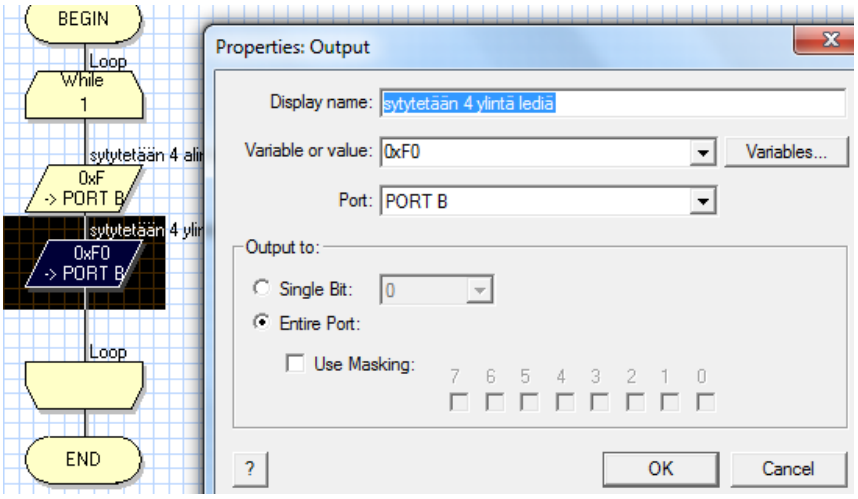
Anna arvoksi **0xF** eli heksaluku 15

Ledikortti on portissa B.

Ja käytetään kaikkia portin B kahdeksaa bittiä.

Ohjelmalle täytyy kertoa mitä lukujärjestelmää käytetään. Flowcodessa voidaan käyttää heksalukua **0x**, binääriluku **0b** tai desimaalilukua.

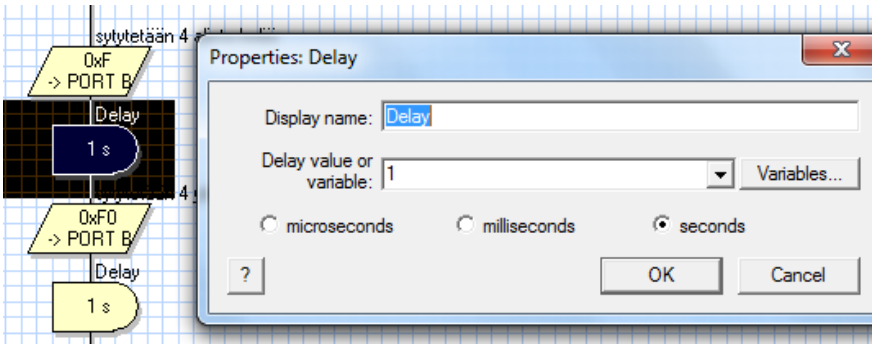
Jos haluat antaa äskeisen arvon binäärilukuna, niin silloin kirjoita **0b00001111**.
Jos taas haluat antaa sen kymmenjärjestelmän lukuna, niin silloin ei tule mitään etumäärettä vaan voit suoraan **value** kohtaan kirjoittaa **15.**



Lisää toinen output kuvake ja siihen anna arvoksi **0xF0** eli sytytetään neljä ylintä bittiä.

Tässä vaiheessa tallenna ohjelma nimellä harjoitus1 ja kokeile miten simulointi onnistuu.

Kuten huomasit ledit vilkkuvat niin nopeasti, että sitä on jo vaikea havaita. Siitä syystä kannattaa lisätä **viiveet** ledien sytyttämisen jälkeen. Raahaa viivekuvake (delay) output kuvakkeen jälkeen ja tuplaklikkaa sitä. Sinulle avautuu viiveen käsittelyikkuna. Valitse viiveen kestoksi 1sekunti.



Tallenna ja simuloi. Simuloinnin jälkeen lataa tekemäsi ohjelma mikrokontrollerille ja näet käytännössä toimiiko ohjelma.

Harjoitus 1.B

Muuta ohjelmaa siten, että ledit D0, D2, D4 ja D6 vilkkuvat vuorotellen ledien D1, D3, D5 ja D7 kanssa. Onnistuu varmasti helpoimmin käyttämällä binäärilukuja heksalukujen sijaan. Tallenna myös muutettu ohjelma harjoitus1B ja simuloi. **Lataa** tekemäsi ohjelma mikrokontrollerille ja kokeile käytännössä toimiiko ohjelma.

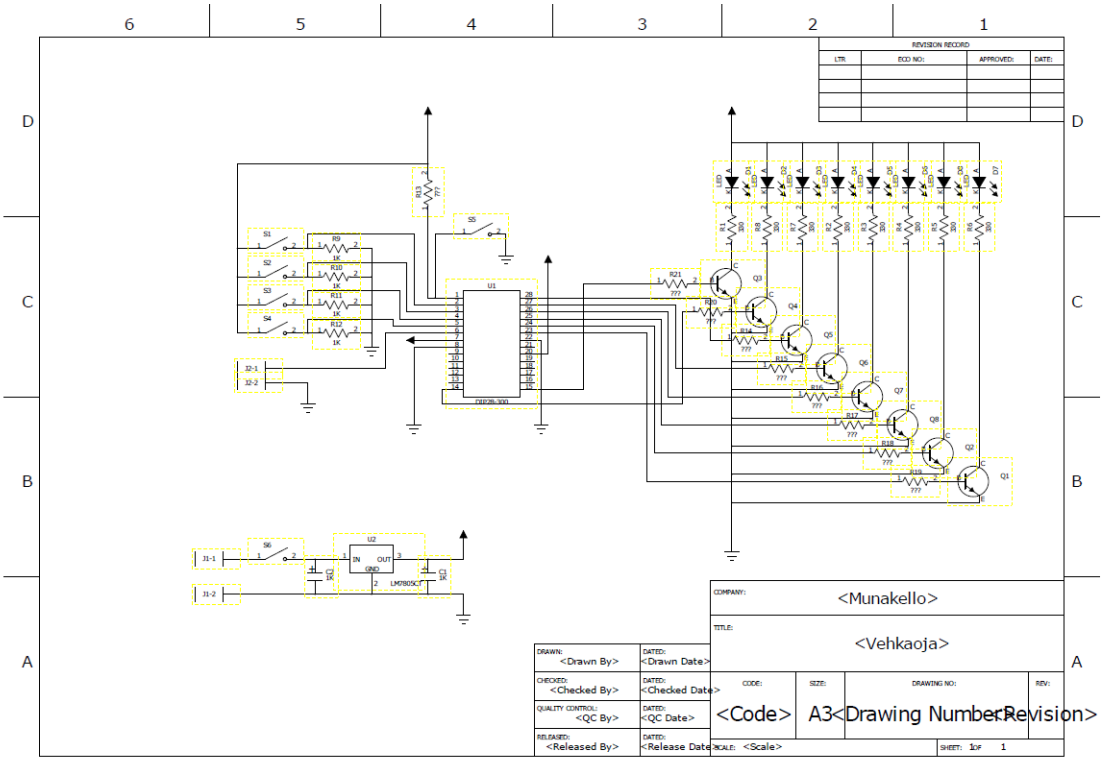
Harjoitus 10 ”Munakello”

Tutustu mikrokontrolleri ATmega168 datasheettiin ja nastajärjestykseen. Rakenna munakello kytkentäkuvan mukaisesti verolevylle. Munakellon osat on lueteltu osaluettelossa. On/off kytkin S6 toimii virtakytkimenä. Painokytkimet S1-S4 asettavat ajan ja painokytkin 5 resetoi ohjelman (kytketty nastaan 1). Summeri on kytketty nastaan 6.

Tee flowcodella ohjelma, joka toimii seuraavasti: Jos painat kytkintä 1 syttyvät kaikki ledit ja munakellon aika käynnistyy esim 3 min. Aika kuluu ja ledit sammuvat yksitellen. Kun kolme minuuttia on kulunut summeri huutaa ja kaikki ledit ovat sammuneet. Toinen vaihtoehto on, että ledit syttyvät vähitellen ja 3 minuutin jälkeen kaikki ledit palavat ja summeri huutaa.

Kytkimestä 2 ajaksi määräytyy esim. 4 minuuttia. Kytkimestä 3 viisi minuuttia ja kytkimestä 4 vaikkapa 6 minuuttia.

Osa	Tyyppi	Arvo	Määrä
Vastus		1 Kohm	5 kpl
Vastus		330 Ohm	8 kpl
Kondensaattori		1 Kf	2 kpl
Ledit			8 kpl
Transistori	BC 547		8 kpl
kytkimet	paino		5 kpl
kytkimet	on/off		1 kpl
Regulaattori	LM 7805		1 kpl
Piiri	Atmega 168-20PI		1 kpl
Summeri			1 kpl


6
5
4
3
2
1
D
C
B
A
REVISION RECORD
LTR
ECO NO:
APPROVED:
DATE:
<Munakello>
<Vehkaoja>
DRAWN: <Drawn By>
DATED: <Drawn Date>
CHECKED: <Checked By>
DATED: <Checked Date>
QUALITY CONTROL: <QC By>
DATED: <QC Date>
RELEASED: <Released By>
DATED: <Release Date>
CODE: <Code>
SIZE: A3
DRAWING NO: <Drawing Number>
REV: <Revision>
SCALE: <Scale>
SHEET: 1 OF 1