Henri Silván

SAMKROBOTIN TOIMINTA JA SEN VIDEOKUVAUKSEN KEHITTÄMINEN

Automaatiotekniikan koulutusohjelma
2015

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

SAMKROBOTIN TOIMINTA JA SEN VIDEOKUVAUKSEN KEHITTÄMINEN

Silván, Henri
Satakunnan ammattikorkeakoulu
Automaatiotekniikan koulutusohjelma
Toukokuu 2015
Ohjaaja: Asmala, Hannu
Sivumäärä: 29
Liitteitä: 0

Asiasanat: robotti, Arduino, web-kamera, videokuvaus

Opinnäytetyön tarkoituksena oli perehtyä SAMKRobotin toimintaan ja kertoa laitteen toiminnasta. SAMKRobot on Satakunnan Ammattikorkeakoulussa suunniteltu ja rakennettu robotti, joka on paikallisessa verkossa kahdella ohjaavalla pyörällä liikkuva laite. Robottia ohjataan sen selainkäyttöliittymän avulla. Sen toimintaa tullaan kehittämään SAMK:ssa aina tarpeen tullen. Opinnäytetyössä kerrotaan, mistä komponenteista robotti muodostuu sekä annetaan tietoa robotin sisäisestä toiminnasta. Lisäksi havainnollistetaan tiedonsiirtoa sekä selaimelta robotin suuntaan että toisinpäin.

Opinnäytetyö toteutettiin konstruktiivisena kvalitatiivisena tutkimuksena. Aineisto opinnäytetyöhön saatiin havainnoimalla SAMKRobotin toimintaa ja suunnittelemalla robottiin uusia lisäosia.

Opinnäytetyön alussa esitellään robotissa käytetyistä tekniikoista tärkeimmät. Tekniikat käydään läpi yksitellen ja niistä annetaan yksinkertaisia havainnollistavia esimerkkejä. Tekniikoiden läpikäyminen on pyritty pitämään perusteiden tasolla kertoen vain oleellisimmat pohjatiedot, minkä avulla on hyvä lähteä perehtymään ja ymmärtämään laitteen syvällisempää toimintaa. Alussa myös kerrotaan tekniikoiden tehtävät ja niiden fyysinen sijainti robotissa.

Robotin toiminnasta kerrotaan tarkemmin myös kooditasolla. Koodeista tarkimmin esitellään Javascript, HTML, CSS ja Arduino -koodit. Pintapuolisesti käydään läpi Python-koodin tehtävä.

Opinnäytetyön tuloksena saatiin toteutettua videokuvaus Arduino Yuniin liitettävällä web-kameralla. Lopuksi videokuvaus otettiin käyttöön myös SAMKRobotissa.

THE FUNCTION OF THE SAMKROBOT AND THE DEVELOPMENT OF ITS VIDEO STREAMING

Silván, Henri
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Automation Technology
May 2015
Supervisor: Asmala, Hannu
Number of pages: 29
Appendices: 0

Keywords: robot, Arduino, webcam, video streaming

The purpose of this thesis was to familiarize the functionality of SAMKRobot and tell how the machine works. The SAMKRobot has been designed and built in Satakunta University of Applied Sciences and it works on local network and runs by two steering tires. The robot is controlled by the browser interface. It is a long time project and it will be developed in SAMK whenever necessary. This thesis describes which are the components of the robot and provides knowledge about how the machine works internally. In addition, the thesis also illustrates the data transmission from browser to the robot and the other way around.

This thesis was conducted as a constructive and qualitative research. The material of this thesis was obtained by observing the functionality of the SAMKRobot and designing new add-ons for the robot.

At the beginning of this thesis the used techniques in robot are introduced. These techniques will be gone through one by one and given simple illustrative examples will be given. Introduction of the techniques have been kept at the fundamental level, telling only the most essential information. It makes a good starting point to familiarize and understand how the machine works. There is also described the tasks of techniques and their physical location in the robot.

The function of the SAMKRobot is discussed in more detail on the code level. The most precise presentation will be given about Javascript, HTML, CSS and Arduino codes. The tasks of the Python code are gone through shortly.

As a result of this thesis, video streaming by webcam attached to the Arduino Yun is completed. Finally, the video streaming was also put to use in SAMKRobot.

SISÄLLYS

1 JOHDANTO

Viime aikoina esineiden internet (Internet of Things) ja teollinen internet on ollut valtavassa nosteessa eikä sen kasvulle ole näkyvissä loppua. Yhä enemmän erilaisia koneita ja sulautettuja järjestelmiä liitetään internetiin ja saadaan ne vaikuttamaan älykkäämmiltä. Aiemmin internet yhdisti vain ihmisiä, mutta nykyään koneet ovat liittyneet keskustelemaan ihmisten ja toisten koneiden kanssa. Tämä on mahdollistanut sen, että esimerkiksi teollisuudessa voidaan tarkkailla ja analysoida koneiden toimia internetin välityksellä. (Vanhalakka 2015, A26–A27)

Tämä opinnäytetyö on ajankohtainen, koska HTML5:n julkaisun myötä web-ympäristössä voidaan luoda entistä nopeampia, tehokkaampia ja rikkaampia internet-sovelluksia. Myös esineiden ja teollisen internetin kehitys tulee jatkumaan tulevaisuudessa, joten siinä on syytä pysyä mukana ja käyttää sen valtavaa potentiaalia laajoissa käyttökohteissa.

Opinnäytetyön kohteena on SAMKRobot. Kyseinen robotti ei ole yhteydessä globaaliin internetiin, vaan se toimii paikallisessa verkossa internet-ympäristössä hyödyntäen HTML5:n uusia mahdollisuuksia. Robotin liittäminen internetiin tavalla tai toisella olisi kuitenkin mahdollista. Esimerkiksi robotista saatavaa videokuvaa voitaisiin katsoa ympäri maailman pilvipalvelun kautta.

Tämän opinnäytetyön tavoitteena on perehtyä SAMKRobotin toimintaan. SAMKRobot on Satakunnan Ammattikorkeakoulussa suunniteltu ja rakennettu robotti. Laite toimii paikallisessa verkossa ja sitä voidaan ajaa selainkäyttöliittymällä, jolloin robotin kahta ohjaavaa pyörää saadaan liikutettua. Sen toimintaa tullaan kehittämään SAMK:ssa aina tarvittaessa. Työssä on tavoitteena kertoa, mistä komponenteista robotti koostuu ja kuinka sitä käytetään. On myös tavoitteena havainnoida robotin sisäistä ja ulkoista tiedonsiirtoa sekä rakentaa robotin selainkäyttöliittymään näkymään videokuvaa robottiin liitetystä web-kamerasta.

2 ROBOTISSA KÄYTETYT TEKNIIKAT

Tässä luvussa esitellään SAMKRobotissa käytössä olevia tekniikoita. Ohjelmointikielistä on syytä hallita erityisesti Javascriptiä ja Arduino-kieltä sekä laitepuolella Arduinon toimintaa, sillä näillä on erittäin laaja asema robotin toiminnan kannalta. Toki muillakin tekniikoilla on oma merkityksensä. CSS ja HTML toimivat selaimessa visuaalisella puolella, kun taas WebSocket toimii selaimen rajapintana. Robotissa käytetään myös Python-ohjelmointikieltä, mutta sitä ei tulla esittelemään tässä työssä, koska sen merkitys laitteen toiminnassa on vain tiedon siirtämisessä ja työtä on myös haluttu rajata.

Kokoelma tekniikoiden tehtävästä ja sijainnista robotissa:

- Arduino-alusta
    - koko robotin toiminnallisuus pohjautuu vahvasti siihen
    - käytetään robotin aivoina
- Arduino-kieli
    - käskytetään robotin liikettä ja toimintaa
    - toimii Arduino Yunin ATmega32U4-mikro-ohjaimessa
- Javascript
    - luodaan selain puolen toiminnallisuus
    - käytössä robotin selainkäyttöliittymässä
- HTML
    - luodaan selainkäyttöliittymän rakenne
- CSS
    - luodaan selainkäyttöliittymän ulkoinen ilme
- WebSocket
    - toimii selaimen rajapintana nopealle tiedonsiirrolle
    - käytössä tiedonsiirrossa tietokoneen tai matkapuhelimen selaimen ja robotin välillä
- Python
    - toimii robotin tiedonsiirrossa
    - käytössä Arduino AR9331 Linux -mikroprosessorissa

2.1 CSS

CSS-kielellä (Cascading Styling Sheet) vaikutetaan nettisivujen ulkoasuun helposti ja monipuolisesti tyyliohjeita käyttämällä. Kielellä määritellään, kuinka HTML-dokumentissa olevat elementit tulee esittää selaimessa. Kieli on W3C:n (World Wide Web Consortium) kehittämä ja ylläpitämä. Siitä on julkaistu jo useita versioita, joista ensimmäinen on julkaistu vuonna 1996. Uusin versio on CSS3 (1998–) ja sen kehitys jatkuu edelleen. Viimeisimmällä versiolla on jo erittäin hyvä selaintuki, mutta on syytä varmistua käytettyjen ominaisuuksien tuesta aina selainkohtaisesti. CSS:n kirjoittaminen on helppo aloittaa, koska koodin tuottamiseen riittää ihan tavallinen tekstieditori. Tosin on myös saatavilla laadukkaampia ohjelmia, joilla kirjoittaminen on helpompaa ja nopeampaa. (W3Schools www-sivut 2015)

CSS syntaksissa ilmoitetaan valitsin, jonka ominaisuuksille annetaan arvoja:

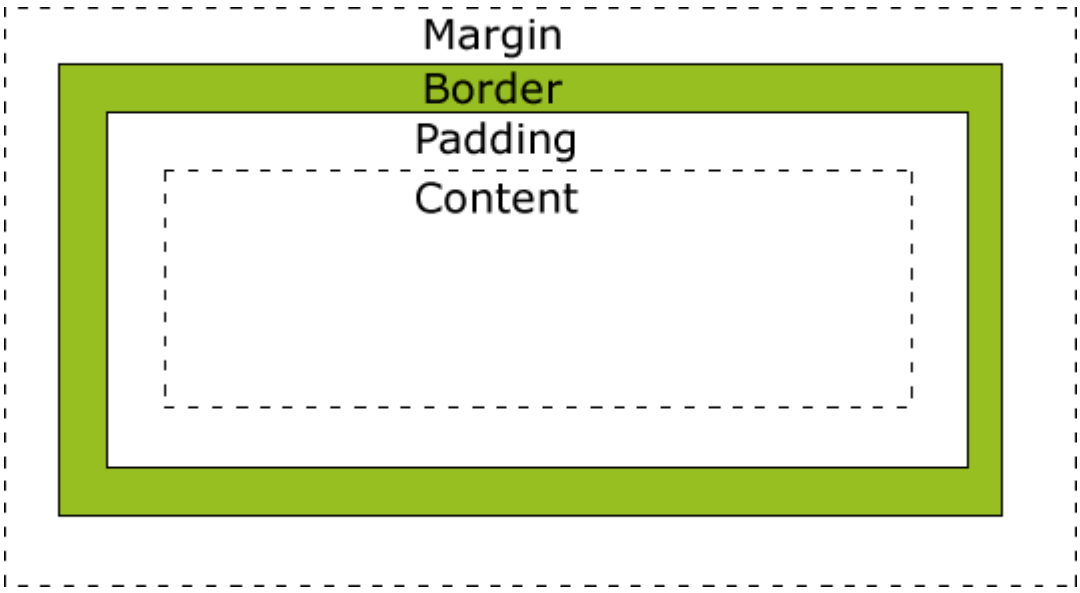
```
valitsin { ominaisuus: arvo; ominaisuus: arvo;}
```

Esimerkki koodissa, jossa muun muassa annetaan sivu sisältöalueelle (body), taustaväri ja tekstin väri sekä h1-tason otsikolle tekstin koko ja väri:

```
body { background-color: blue; color: red; }
h1 { font-size: 20px; color: green; }
.paraUL { text-decoration: underline; }
#imgRobot { border: 1px solid black; }
```

Valitsimia on monenlaisia, kuten esimerkki osoittaa. Ensinnäkin tyylejä voidaan määritellä elementtikohtaisesti, kuten bodylle. Toisaalta voidaan antaa tyylejä luokkakohtaisesti (class) kaikille elementeille, joilla on sama luokka-arvo, kuten .paraUL. Voidaan myös antaa tyyliohjeita yksilöidyn tunnisteen (id) avulla, jolloin tyylit vaikuttavat vain yhteen elementtiin, kuten #imgRobot. Näiden tapojen lisäksi on myös muita keinoja ilmoittaa tyylejä, kuten kerralla useammalle elementille tai tietylle sisäiselle elementille. Kielessä käytetään myös pseudo-luokkia, joista esimerkkinä voisi olla yleisesti käytetty :hover, jolla luodaan elementille vaihtuva näkymä käyttäjän osoittaessa hiirellä elementtiä. (W3Schools www-sivut 2015)

Tyylimääritykset voidaan kirjoittaa kolmeen paikkaan, joita ovat erillinen tiedosto, head-osio ja suoraan elementin aloitustagiin. Paikat ovat järjestyksessä vähiten merkitsevästä eniten merkitsevään. Sivustoa tehdessä tyyliohjeet voidaan tallentaa erilliseen tiedostoon ja linkittää HTML-tiedostoihin. Tällöin selvitään pienemmällä työllä, koska jokaiselle HTML-tiedostolle ei tarvitse kirjoittaa mahdollisia samoja tyylejä uudestaan. On myös hyvä pitää mielessä HTML-elementtien pohjautuminen kuvan 1 mukaiseen laatikkomalliin, jolloin elementin sisällön sijaintiin vaikuttavat marginaalit, reunaviivat ja täytteet. Lisäksi selaimilla on omat oletustyylinsä, jonka mukaan ne toimivat, jos tyylimääreitä ei muuteta. CSS-tiedoston päätteenä on yleisesti css. (W3Schools www-sivut 2015)



Kuva 1. Laatikkomalli (W3Schools www-sivut 2015).

2.2 HTML

HTML (HyperText Markup Language) on kuvauskieli, jolla nettisivuja saadaan rakennettua. HTML-dokumentissa kerrotaan, kuinka internetselaimen tulisi tulkita ja esittää dokumentin sisältö. Kieli on alkujaan Tim Berners-Leen keksimä vuonna 1991, jonka jälkeen kielestä on julkaistu useita versioita. HTML5 on kielen uusin standardi, jonka päivittäminen ja kehittäminen tulevat jatkumaan. Kaikki modernit selaimet tukevat hyvin uusinta versiota ja nykyisin HTML5:llä kannattaa toteuttaa uudet nettisivut. Tällöin sivut saadaan monipuolisemmiksi ja niiden koodaaminen

tulee olemaan helpompaa, koska saadaan kielen uusimmat päivitykset käyttöön. Sivuja tehdessä on myös hyvä ottaa huomioon vanhojen selainten käyttäjät ja toteuttaa nettisivut niin, että myös he näkevät sivut mahdollisimman samankaltaisina. (W3C www-sivut 2015)

HTML-dokumentti rakennetaan useilla HTML-elementeillä, jotka koostuvat yleisesti kahdesta tagista (tunniste): aloitus- ja lopetustagista. Tagien välissä kerrotaan elementin sisältö. Esimerkki rakenteesta alla:

```
<tunnisteen nimi>sisältö</tunnisteen nimi>
<p>Tämä on tekstikappaleen sisältöä.</p>
```

Esimerkki yksinkertaisesta nettisivusta kommentoituna:

```
<!DOCTYPE html> <!—Ilmoitetaan dokumentin kieleksi HTML5 -->
<html> <!-- Kuvaillaan HTML-dokumenttia -->
<head> <!--Tietoa selaimelle dokumentista -->
<title>Sivun otsikko</title> <!--Työkalupalkin otsikko -->
</head>
<body> <!-- Sivun näkyvä sisältö -->
<h1>Ensimmäinen otsikko</h1> <!-- Otsikko -->
<p>Tässä kappale tekstiä.</p> <!-- Tekstikappale -->
</body>
</html>
```

2.3 Javascript

Javascript on yleisesti HTML:n yhteydessä selainskriptien tekemiseen käytetty kieli, jolla saadaan lisättyä internetsivuille vuorovaikutteisuutta ja toiminnallisuutta. Skripti on ohjelmakoodi, joka suoritetaan HTML-dokumentin näyttämisen yhteydessä asiakkaan selaimessa, mutta sitä on myös mahdollista käyttää muun muassa palvelimessa toimivana. (Korpela 2014, 55.)

Netscape kehitti Javascriptin vuonna 1995 tuomaan verkkosivuille dynaamista toiminnallisuutta web-suunnittelijoiden tarpeesta. Kieli oli tarpeen kehittää, koska HTML:n katsottiin olevan tähän yksin riittämätön. Kieli on kehittynyt huomattavasti alkuajoistaan ja siitä on tullut tärkeä palanen selainohjelmoinnissa. (Negrino & Smith 2007, 1.)

Javascriptillä hallinnoidaan sivuston käyttäytymistä monin mahdollisin tavoin. Sillä voidaan muun muassa paikata selainten HTML5-tuen aukkoja sekä muokata HTML-sivun rakennetta, sisältöä ja ulkoasua. Kielellä on mahdollista tehdä suuriakin interaktiivisia nettisovelluksia, kuten pelejä. Sen osuus on merkittävä ns. HTML5-sovelluksissa. (Korpela 2014, 56.)

Selainohjelmointikielenä Javascriptille on toteutettu erittäin laaja tuki selaimissa. Koodi kirjoitetaan ensisijaisesti ulkoiseen js-tiedostoon, jonka viittaus lisätään HTML-dokumenttiin. Javascript-koodia on myös mahdollista kirjoittaa HTML-dokumenttiin script-elementillä, HTML-elementtien tapahtumamääritteiksi tai näennäisprotokollan avulla. (Korpela 2014, 56–57.)

Dokumenttioliomallin (DOM) avulla päästään käsiksi HTML-dokumentin elementteihin ja selaimen toiminnallisiin osiin, jolloin ne ovat javascriptin käytettävissä. DOM on tärkeä palanen hallinnoitaessa HTML-sivua javascriptillä, vaikka se ei suoranaisesti kieleen kuulukaan. (Korpela 2014, 56–57.)

2.4 Arduinon ja sen ohjelmointikieli

Arduino on erittäin suosittu avoin elektroniikan kehitysalusta. Laitteita on saatavana useita eri malleja, joista valitaan omaan käyttöön sopivin. Laitteen käyttöönotto ja ohjelmointi on tehty erityisen helpoksi ja internetistä saa tarvittaessa hyvin apua ongelmatilanteissa. Tämän vuoksi edullisena kehitysalustana Arduino soveltuu hyvin aloittelijoille ja prototyyppien rakentamiseen. Arduino yhdistetään tietokoneeseen kehitysalustasta riippuen joko langallisesti (esim. USB) tai langattomasti (esim. WLAN). Arduino-tuotemerkin alta löytyy myös joukko kehitysalustoihin liitettäviä

lisäosia. (Banzi 2011, 1–6; Karvinen & Karvinen 2010, 34–36; Arduinon www-sivut 2015.)

Arduino-ohjelmointikieli on Arduino-kehitysympäristössä (IDE) käytettävä kieli. Se on C++:aan pohjautuva oma Prosessing-kieli, jolla saadaan ohjelmoitua Arduino-kehitysalustojen toimintaa omien käyttötarpeiden mukaan. Kehitysympäristö toimii kaikilla yleisimmillä käyttöjärjestelmillä (Windows, Macintosh, Linux). Ympäristö tarkistaa koodin ja muuttaa sen konekielelle sekä lähettää koodin laitteen mikro-ohjaimelle. Tällöin laitteella voidaan esimerkiksi ohjata porttipinneihin kytkettäviä toimilaitteita tai lukea pinnien tilatietoja. Arduinon etuna etenkin aloittelijoille on sen perustuminen avoimeen lähdekoodiin ja laitteistoon. (Banzi 2011, 20; Karvinen & Karvinen 2010, 34–36.)

Yksinkertainen koodiesimerkki Arduinon pinnissä 13 olevan ledin ohjauksesta, jossa ledi vilkkuu ollen kaksi sekuntia päällä ja yhden sekunnin pimeänä. Laitteen ollessa päällä ohjelma pyörii ikuisella silmukalla:

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

void loop()
{
digitalWrite(13, HIGH);
delay(2000);
digitalWrite(13, LOW);
delay(1000);
}
```

2.5 WebSocket

WebSockets (Web-vastakkeet) on HTML5:n julkaisun mukana tullut uusi protokolla. Se mahdollistaa kaksisuuntaisen samanaikaisen (full-duplex) tiedonsiirron web-ympäristössä asiakkaan ja palvelimen välillä. Näin voidaan toteuttaa entistä nopeampia ja tehokkaampia nettisovelluksia. Asiakkaan eli selaimen ja palvelimen välille avataan jatkuva yhteys, jonka kautta siirretään vain tarvittava tieto. Tämä eroaa aiemmissa tekniikoissa, joissa on tarpeen liikuttaa paikoin huomattavastikin enemmän dataa. (Korpela 2014, 786–787.)

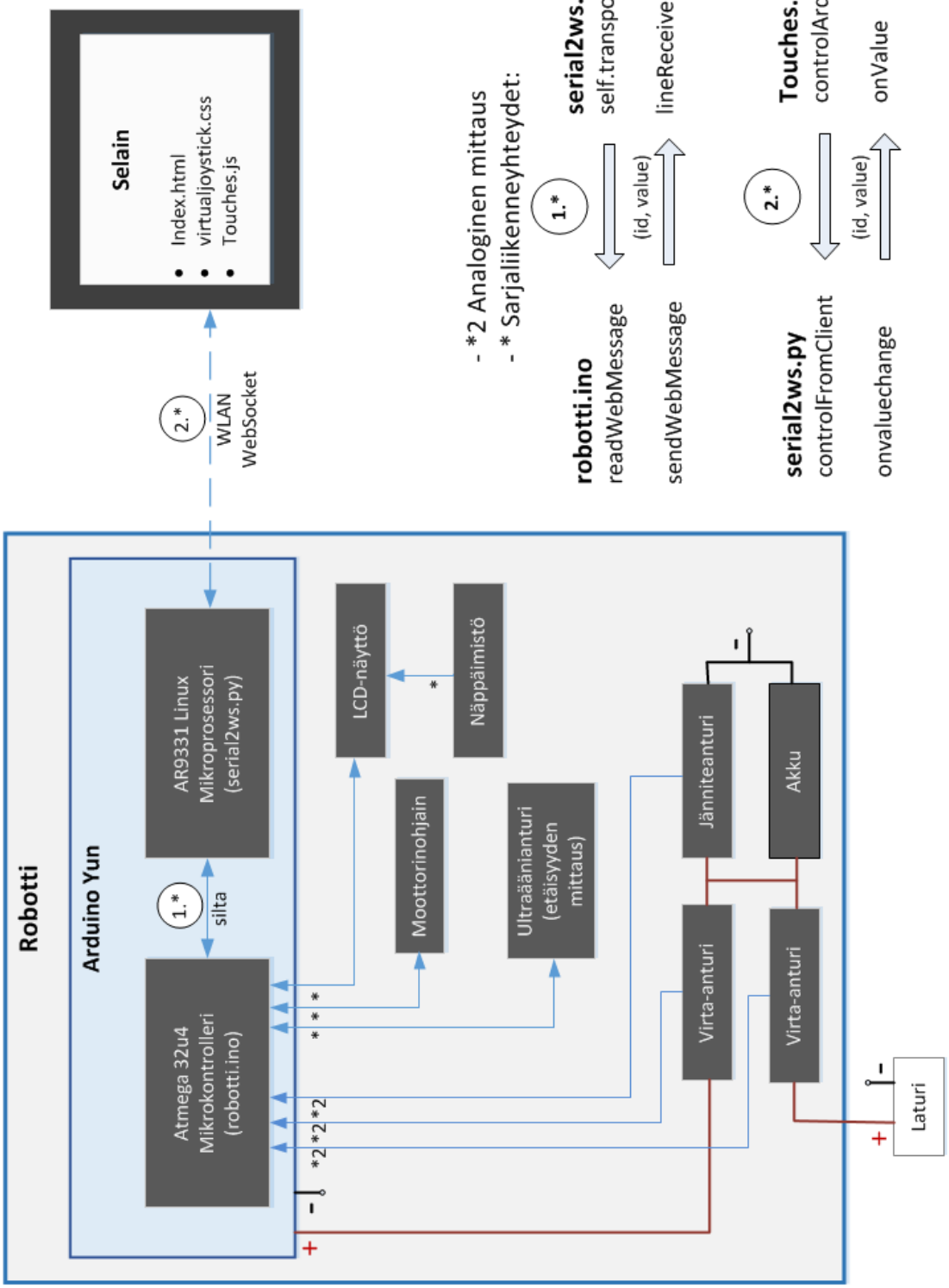
Vastaavan tapaisia tekniikoita ennen WebSocketia olivat Polling, Long Polling, Streaming, Postback ja AJAX. Nämä aiemmat tekniikat on rakennettu myös reaaliaikaista kaksisuuntaista kommunikointia varten, mutta eivät ole yhtä tehokkaita kuin uusin WebSocket. Aiempien tekniikoiden tiedonsiirto on kaksisuuntaista ja eriaikaista (half-duplex), jolloin sekä palvelin että asiakas joutuvat odottamaan toisen lopetusta. Myös http-otsikoiden lähettämisessä koituu välitettävän datan määrän kasvua ja ne lisäävät viivettä. Tällöin myös palvelin kuluttaa erityisen paljon resursseja kommunikointiin. (Pterneas 2013, 8–9.)

WebSocket API on ohjelmointirajapinta, joka sallii luoda yhteyden paikalliseen tai etäpalvelimeen, vastaanottaa viestejä, lähettää dataa ja lopuksi sulkea yhteyden. Käytettäessä WebSocketia kannattaa aluksi tarkistaa selaimen tuki siihen. Tuen puuttuessa käyttäjälle on hyvä antaa siitä palautetta. Jos taas tuki löytyy selaimesta, voidaan pyytää palvelinta avaamaan yhteys. Kun yhteys on muodostettu, tietoa välitetään sovelluskohtaisesti asiakkaan ja palvelimen välillä. Kommunikoinnin lopussa yhteys suljetaan, esimerkiksi käyttäjän painaessa sulje yhteys -painiketta. (Pterneas 2013, 17–18, 27.)

3 ROBOTIN TOIMINTA

SAMKRobot on kahdella ohjaavalla renkaalla langattomasti liikutettava robotti (kuva 2). Sen aivoina toimii Arduino Yun -kehitysalusta, jossa sijaitsee laitteen toiminnan kannalta tärkeät mikro-ohjaimet ATmega32U4 ja AR9331 Linux. Näistä ensin mainittu keskustelee sarjaliikenteen kautta AR9331 Linux:n lisäksi moottorinohjaimen, LCD-näytön, näppäimistön ja antureiden (ultraääni, 2*virta ja jännite) kanssa. Ultraäänianturilla mitataan etäisyyttä robotin edessä oleviin kappaleisiin ja jänniteanturilla tarkkaillaan akun käyttöjännitettä. Virta-antureilla sen sijaan mitataan sekä lataus- että käyttövirtaa. Moottorinohjainkorttiin on liitetty moottorit renkaineen, joilla laitetta saadaan liikutettua. AR9331 Linux -mikroprosessorin tehtävänä taas on hoitaa langatonta tiedonsiirtoa robottia ohjaavan laitteen kanssa, joka voi olla esimerkiksi tietokone tai matkapuhelin. Samalla AR9331 Linux myös keskustelee toiseen suuntaan ATmega32U4-mikro-ohjaimen kanssa. Seuraavalla sivulla oleva kuva 3 havainnollistaa robotin rakennetta ja yhteyksiä eli sitä, kuinka tieto liikkuu sekä robotin sisällä että robotin ja selaimen välillä.

Kuva 2. SAMKRobot.



Kuva 3. Robotin rakenne ja yhteydet.

Tiedonsiirto robotilta selaimelle:

1. robotti.ino, sendWebMessage
2. serial2ws.py, lineReceived
3. serial2ws.py, onvaluechange
4. Touches.js, onValue

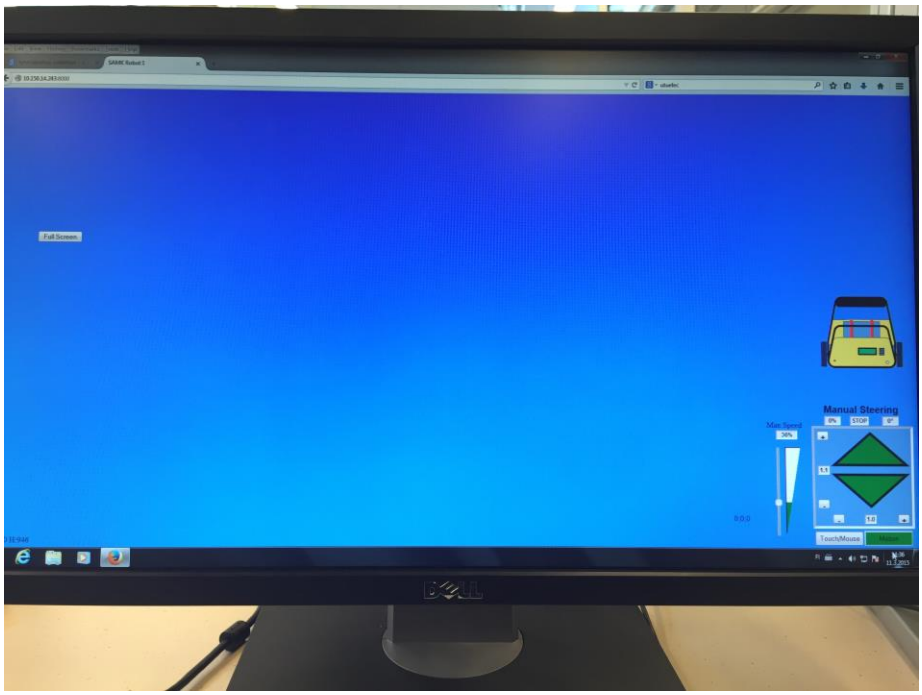
Tiedonsiirto selaimelta robotille:

1. Touches.js, controlArduino
2. serial2ws.py, controlFromClient
3. serial2ws.py, self.transport.write
4. robotti.ino, readWebMessage

Langattomassa tiedonsiirrossa käytetään WebSocket-protokollaa. Langaton yhteys luodaan kerran alussa, jonka jälkeen se on jatkuvasti auki. Koska siis yhteyden kättelyä ei tarvitse jatkuvasti suorittaa, saadaan siirrettävän datan määrä karsittua pieneksi ja tällöin myös tiedonsiirron viive tulee olemaan vähäinen. Tämä on oleellista hallittaessa tämän kaltaista laitetta.

Käyttäjä ohjaa robottia langattomasti wlan-yhteyden kautta. Haluttaessa ohjata robottia langattomasti, tulee käyttäjän ensin käynnistää robotti kääntämällä virta-avain on-asentoon. Tällöin alkaa robotin alustaminen toimintakuntoon ja robotin näyttöön tulee teksti ”Starting…”. Hetken kuluttua, kun LCD-näyttö päivittyy, robotti on valmis ohjattavaksi.

Käyttäjä avaa selaimen ja kirjoittaa sen osoitekenttään sekä robotin IP-osoitteen että avattavan portin. Tällöin luodaan langaton yhteys tietokoneen ja robotin välille. Kun yhteys on auki, ladataan robotilta nettisivu index.html (kuva 4) ja myös muut siihen liitetyt tiedostot. Kaiken ollessa kunnossa, robotti on valmiina ohjattavaksi. Käyttäjä voi valita robotille haluamansa nopeuden liukusäätimellä ja sen jälkeen ohjata robottia vierityspallon avulla.



Kuva 4. Robotin internetsivu.

3.1 Javascript-koodi

Javascript-koodilla luodaan WebSocket-asiakas selaimen päähän ja päivitetään käyttöliittymää. Tämän koodin kautta tieto siirtyy selaimelta robotille ja toisinpäin. Koodin alussa luodaan funktiot selaimen näkymän päivitykseen ja muuttamiseen kokonäytölle. Full Screen -painiketta painamalla selainkäyttöliittymän näkymää saadaan muutettua koko näytölle. Painikkeelle on laitettu HTML-koodissa onclick-tapahtumaan funktion fullscreen()-kutsu. Tämä Javascriptin fullscreen()-funktio suorittaa varsinaisen näkymän muutoksen. Funktio hakee ensin HTML-koodista elementin, jonka id on base. Selaimesta riippuen tarkastellaan, minkälainen requestFullscreen-ominaisuus selaimella on ja suoritetaan funktio, joka muuttaa näkymän. Sen jälkeen määritellään funktio requestAnimFram(), jota apuna käyttäen saadaan piirrettyä canvas-elementti uudestaan. Tämä toteutetaan tasaisin väliajoin, joka tässä tapauksessa on 30 kertaa sekunnissa.

Koodissa käytettävät muuttujat on tarpeen esitellä ja alustaa ennen niiden käyttöä. Dokumentilla on myös EventListener, jolla tarkkaillaan, milloin sivuston rakenne on latautunut. Tällöin kutsutaan funktiota init(), jonka kautta alustetaan selaimen näky-

mä käyttövalmiiksi. Selaimen ikkunan koon muuttaminen tai näkymän kääntyminen aiheuttaa canvasin resetoinnin.

Robotilta tulevien tietojen päivittäminen robotin nettisivuille hoidetaan funktiolla onValue(args), joka saa parametrina id ja value arvopareja. Tämä data käsitellään ja sisällytetään näkyväksi HTML:ään. Datan käsittelyssä value jaetaan puolipisteen kohdalta, taulukoidaan, parsitaan kokonaisluvuksi ja sijoitetaan arvot oikeisiin muuttujiin.

Tietojen lähettämiseen robotille käytetään funktiota controlArduino(id, data), jolle kutsuttaessa välitetään parametreina id ja data. Nämä arvot siirtyvät WebSocket-yhteyttä apuna käyttäen AR9331 Linux -mikroprosessorille, joka vastaanottaa id:n ja datan serial2ws.py python-koodin funktiolla controlFromClient.

Funktiolla updateEventCnt() suoritetaan kutsu tietojen lähetystä varten robotille, kun kutsutaan funktiota controlArduino() ja välitetään sille parametreina sekä id että data. Lähetyksen ehtona on, että vähintään yksi ohjauksen tiedoista on muuttunut. Jos muutos on tapahtunut, ohjauksen nykyiset tiedot sijoitetaan muuttujiin, joiden arvoina halutaan pitää ohjauksen edelliset tiedot. Näin tarkkaillaan ohjaustietojen lähettämisen tarpeellisuutta.

Nettisivun sisällön latauduttua luodaan WebSocket-yhteys WAMP:lla, joka tulee sanoista Web Application Messaging Protocol. Autobahn-asiakkaille luodaan yhteys WAMP reitittimelle Crossbar.io ja avataan yhteys. Luodaan myös istunto, jossa julkaistaan onValue-funktion avulla selaimelle päin tuleva dataa, jos sitä on saatavilla. Lisäksi kokeillaan, onko tarpeellista lähettää tietoja robotille päin updateEventCnt() -funktion avulla. Kyseistä funktiota kutsutaan kymmenen kertaa sekunnissa. Sen jälkeen määritellään init()-funktio, jolla luodaan canvakset funktiolla setupCanvas(), lisätään canvasille osoitinta tarkkailevat tapahtuman kuuntelijat ja tehdään kokoelma touches. Kutsutaan myös funktiolla requestAnimFrame funtiota draw(), joka piirtää canvasin sisällön näkyviin.

Canvas resetoidaan funtiolla resetCanvas(e), jota kutsutaan muun muassa muutettaessa selaimen ikkunan kokoa. Tällöin canvas tyhjentyy. Canvas-alustalle haetaan uu-

si koko ikkunan koosta. Lisäksi varmistetaan scrollaus vasemmasta yläkulmasta ja nollataan robotin ohjauksen tiedot.

Piirto canvaseille toteutetaan funktiota draw() apuna käyttäen. Tässä funktiossa ensin tyhjennetään halutut osat canvaseista ja piirretään robotin ohjaukseen käytettävä vierityspallo takaisin näkyviin. Tämän lisäksi lasketaan robotin liikkeen perusteella, minkälainen nuoli piirretään robotin kuvan ylä- tai alapuolelle. Nuoli osoittaa ohjataanko robottia eteen vai taakse ja sen viivan paksuus muuttuu robotin nopeuden mukaan. Nuoli on suora, jos robotti kulkee suoraan, mutta kaareutuu robotin kääntyessä. Funktiossa draw() myös päivitetään sekä ohjauksen arvoja ilmaisevat span-elementit että nopeuden valinnan liukusäädin ja piirretään vielä viiva vierityspalloon hiiren liikkeiden mukaan. Tämän funktion piirto suoritetaan tasaisin väliajoin requestAnimFrame-funktion avulla.

Funktiolla createPointerObject luodaan nimensä mukaisesti osoitinobjekti. Tarkastetaan, onko käytössä esimerkiksi hiiri ja piirretään laskettuun ohjauksen aloituskohtaan ympyrä. Funktio palauttaa tarkastelemiaan tietoja.

Funktiolla onPointerDown(), onPointerMove() ja onPointerUp() tarkkaillaan osoittimen käyttäytymistä. Näistä ensimmäinen lisää kohdan kokoelmaan touches funktiota createPointerObject apuna käyttäen, kun osoitin on ala-asennossa. Funktiolla onPointerMove tarkkaillaan kokoelmaan touches lisättyjen tapahtumien osoittimien liikkeitä. Lasketaan kohta, jossa osoitin sijaitsee robotin ohjauksen canvasissa ja myös ohjauksen asetusarvot. Näin saadaan ohjauksen nopeuden, suunnan ja kulmanasetusarvot. Funktiolla onPointerUp tuhotaan ohjaustapahtuman tiedot ja nollataan tarvittavat muuttujat, joita ovat ohjauksen asetusarvot ja aloituskohta.

Funktiolla setupCanvas() poimitaan sitä kutsuttaessa canvasit HTML-tiedostosta id:n perusteella ja tehdään niistä myöskin 2d kontekseja. Tämä funktio hakee myös ohjaustietoja ilmaisevien span-elementit id:n perusteella Javascriptin käyttöön. Näiden lisäksi canvasSurf saa koon ikkunan koosta ja kutsutaan funktiota setSpeedLimit().

Kyseisen funktion setSpeedLimit() kautta hoidetaan nopeuden valitsevan liukusäätimen alustaminen ja päivitys funktiolle välitetyllä arvolla. Tämä funktio kutsuu toista

funktiota speedLimitGraph(), joka suorittaa siis varsinaiset näkyvät muutokset liukusäätimeen.

3.2 HTML-koodi

HTML-koodilla luodaan nettisivun pohjarakenne. Aluksi ilmoitetaan dokumentissa käytettävän HTML-kielen olevan HTML5 ja sivulla käytettävä tekstikieli, joka tässä tapauksessa on englannin kieli (en). Head-osiossa kerrotaan tietoja, mitä HTML-dokumentti pitää sisällään. Sivun merkistöksi valitaan utf-8 ja kerrotaan sisältöalueen korkeus ja leveys sekä skaalautuvuus. Sivulle määritellään title-elementillä otsikko SAMK Robot 1, joka tulee näkymään selaimen työkalupalkissa. Head-osiossa tehdään myös linkit tarvittaviin Javascript- ja tyylitiedostoihin.

Body-osiossa luodaan sivun varsinainen sisältö, joka laitetaan div-elementin sisälle. Div saa id:n arvoksi base ja luokan container. Tämä div yksilöidään id:llä, jota käytetään sivun näkymän muuttamisessa koko näytölle. Luokkaa container taas käytetään, kun annettaan tyylimäärityksiä CSS-tiedostossa. Sisältöalueen kahdelle alimmalle tasolle luodaan canvas-elementit ja määritetään niille koot. Pohjatasolla on canvasSurface ja ylemmällä tasolla canvasJoystick. Sivulle liitetään kuva robotista img-elementillä, jossa kerrotaan kuvan id, vaihtoehtoinen teksti, kuvan koko sekä käytettävä kuva ja sen osoite.

Nettisivu sisältää myös useita span-elementtejä, joille määritellään luokat, id:t ja tekstin keskitys sekä z-index-tason arvoksi kaksi. Painikkeen Full Screen tapahtumalla onclick kutsutaan funktio fullscreen(), joka taas määritellään Touches.js-tiedostossa. Painikkeen sijainti on absoluuttinen, etäisyys sivun vasempaan yläkulmaan nähden on 300 pikseliä ylös ja 100 pikseliä vasemmalle. Määritellään myös painikkeelle ja liukusäätimelle sama z-index-taso kuin spaneille. Sijoitetaan liukusäädin absoluuttisesti sivun oikeasta alakulmasta sekä annetaan sille koko, läpinäkyvä taustaväri ja suunta. Säätimen arvoa voi muuttaa nollasta sataan askeleen ollessa yksi, näin säädetään robotin nopeutta. Dokumentin latauksessa robotin nopeuden alkuarvoksi laitetaan 20. Kun arvo säätimessä muuttuu, toteutuu onchange-tapahtuma ja haluttu nopeuden arvo saadaan välitettyä eteenpäin.

3.3 CSS-koodi

Robotin nettisivun tyylit luodaan paljolti erillisten tyylitiedostojen avulla. Nämä tiedostot linkitetään HTML-tiedostoon, esimerkiksi CSS-tiedostossa virtualjoystick.css sijaitsee index.html-tiedostoon liitettyjä tyylimäärityksiä. Tyylitiedoston tähdellä merkityt tyylit vaikuttavat koko dokumenttiin. Tässä kohtaa on annettu webkit:lle ohjeita. HTML-elementteihin voi viitata suoraan niiden nimillä, esimerkiksi kirjoitetaan vain body, jonka jälkeen tyylit kirjoitetaan aaltosulkujen sisälle. Näin CSS-tiedostossa boby-elementille on laitettu taustaväri valkoiseksi ja marginaaleiksi nolla pikseliä.

Tyylejä luodaan myös yksilöidysti id:n avulla sekä ryhmittäin luokan (class) avulla. Esimerkiksi canvaseille canvasSurface ja canvasJoystick sekä kuvalle imgRobot annetaan tyylit id:n avulla, jolloin tyylitiedostossa nimien eteen lisätään #-merkki. Robotin kuvaan viitataan siis #imgRobot, jossa kuvalle kerrotaan muun muassa sen tarkka absoluuttinen sijainti. Toisaalta tyylit voidaan antaa kerralla isommalle joukolle elementtejä luokkien avulla, jolloin samat tyylimääritykset vaikuttavat kaikkiin niihin elementteihin, joiden class-attribuutin arvot ovat samat. Tyylitiedostossa luokan nimeen viitataan lisäämällä piste luokan nimen eteen. Luokkia hyväksikäyttäen tyylejä annetaan muun muassa sisältöalueelle container ja robotin ohjaustietoja esittäville span-elementeille, jotka on luokitettu valueDisp nimellä. Näille luokille tyylit ilmoitetaan siis tyylitiedostossa kerralla ryhmitetysti .container ja .valueDisp. Näille ohjaustietoja esittäville valueDisp:eille on aseteltu muun muassa koko, taustaväri vaalean harmaaksi ja reunaviivojen tyylit sekä tekstin sijoitus oikealle ja fontti Arialiksi.

Nettisivun ulkoasuun on vaikutettu myös muita tapoja hyödyntäen, esimerkiksi robotin nopeuden määrittämiseen käytetyn liukusäätimen ulkonäkömäärityksiä on annettu ilmoittamalla input[type=range]. Tällöin viitataan input-elementteihin, joiden type-attribuutin arvona on range. Näin on annettu elementille muun muassa korkeus ja leveys.

3.4 Python-koodi

Python-koodilla on oleellinen tehtävä langattoman tiedonsiirron kannalta. Se siis hoitaa AR9331 Linux:ssa tiedon välitystä langattomasti selaimen kanssa ja tämän lisäksi se keskustelee langallisesti sarjaliikenteellä myös toiseen suuntaa ATmega32U4:n kanssa. Kun yhteyttä selaimella avataan robotin suuntaan, python-koodissa luodaan asiakas langatonta tiedonsiirtoa varten. Jotta laitteen käyttöliittymä saadaan robotilta käyttäjän hallintaan, tarvitsee käyttöliittymän tiedostot saada selaimelle näkyviin. Näiden alkutoimenpiteiden jälkeen, kun robotti on ohjattavissa, AR9331 Linux:ssa oleva python-koodi keskittyy lähinnä tiedon välitykseen.

3.5 Arduino-koodi

Arduino Yun -alustan robotti.ino-tiedostossa on ohjelmakoodi ATmega32U4-mikro-ohjaimelle. Tiedoston alkuun lisätään kirjasto SoftwareSerial.h sarjaliikennettä varten ja määritellään pinnit, joita sarjaliikenteessä tullaan käyttämään. Moottorinohjainkortti MD49 ja LCD-näyttö LCD05 saavat omat pinninsä sekä datan lähetykseen että vastaanottoon. Tämän lisäksi moottorinohjainta ja näyttöä varten lisätään määreitä (define), jotta molempia laitteita olisi helpompi hallita niiden ohjauksen koodia kirjoitettaessa sekä myöhemmin luettaessa. Myös koodissa yleisesti tarvittavat muuttujat on syytä määritellä ennen varsinaista toiminnallista koodia. Muuttujille annetaan tyyppi, nimi ja monesti myös jokin alkuarvo, joka yleensä asettuu joko nollaksi tai epätodeksi (false).

Mikro-ohjaimen ATmega32U4 käynnistyttyä suoritetaan aina koodin funktio setup() kertaalleen. Tässä osassa ilmoitetaan, mikä mikro-ohjain on käytössä ja valitaan sarjaportit Arduino-kortin mukaan. Sen lisäksi on tarpeen avata tiedonsiirtonopeudella 9600 bps olevat yhteydet näytölle ja moottorinohjaimelle. Näytölle asetetaan myös kontrasti sekä tyhjennetään näyttö ja ajetaan se kotiasemaan, johon kirjoitetaan teksti ”Starting…” laitteen käynnistymistä ilmoittamaan.
Luodaan tiedonsiirron kannalta tärkeät funktiot, joita myöhemmin tullaan kutsumaan funktiossa loop(). Funktiolla readWebMessage() vastaanotetaan AR9331 Linux:lta saapuva data, jos sellaista on saatavilla ja luetaan data talteen merkki kerrallaan. Vas-

taavasti tiedon lähetys AR9331 Linux:lle hoituu funktiolla sendWebMessage(), joka saa parametreina merkkijonomuuttujat id:n ja data:n. Näiden muuttujien arvot tämä funktio sitten lähettää hallitusti AR9331 Linux:lle lähtevään porttiin. Koodissa määritellään myös funktio getValue(), jolla saadaan pilkottua datasta robotin ohjauksen asetusarvot, joita on liikkeen nopeudella, suunnalla ja kulmalla.

Seuraavaksi kutsutaan jo aiemmin mainittua funktiota loop(), jota suoritetaan alusta loppuun ikuisella silmukalla. Sen kutsua toteutetaan aina niin kauan, kun Arduino Yunin ATmega32U4-mikro-ohjain on päällä. Tähän osioon kirjoitetaan varsinainen toiminnallinen koodi robotti.ino-tiedostossa. Ohjelma tarkkailee siis jatkuvasti käyttäjän selaimelta tulevia ohjauskäskyjä ja robotin näppäimistön painalluksia sekä hallitsee moottorinohjainta ja LCD-näyttöä. Sen lisäksi koodissa lähetetään tietoa myös robotilta selaimelle päin.

Funktiolla readMD49Data() keskustellaan moottorinohjaimen kanssa sen jälkeen, kun sen toiminta on ensin alustettu funktiolla writeMD49Setup(). MD49-moottorinohjaimelta kysellään erilaisia tietoja, joita ovat muun muassa molempien moottorien nopeudet. Tämän jälkeen moottorinohjain on valmis vastaanottamaan ohjauskäskyjä. Lasketaan robotille uusi nopeus ja kulma, jolloin robotin renkaat voivat pyöriä erilaisilla nopeuksilla toisiinsa nähden. Uudet robotin liikkeen tiedot lähetetään myös selaimelle, jotta käyttäjä saa tietoa robotin liikkeestä ja sitä on helpompi hallita.

4 ROBOTIN KEHITTÄMINEN

4.1 Videokuvauksen toteutuksen perustaa

Arduinolla suoritettavan videokuvauksen toteutuksessa lähdettiin liikkeelle pohjatietojen etsinnällä internetistä. Englanninkielisistä lähteistä saatiin käsitystä siitä, mitä videon kuvaaminen ja kuvan välittäminen selaimelle käyttäjän nähtäväksi voisi vaatia. Suomenkielistä materiaalia aiheeseen liittyen ei juurikaan löytänyt.

Alussa jo kävi selväksi, että aihe tulee olemaan uutta, koska Arduino Yun -kehitysalustakin on vielä melko uusi tuttavuus, joten mistään vastaavankaltaisesta työstä ei ollut aiempaa kokemusta. Tämän lisäksi Atheros AR9331 on Linux-mikroprosessori, jonka käyttöä on tarpeen opetella. Sitä tarvitaan videokuvauksen toteutukseen, koska web-kamera tullaan liittämään siihen USB-portin välityksellä. Komentorivin käyttökin oli lähtökohtaisesti melko uutta. Sitä tullaan käyttämään tietokoneella luotaessa yhteyttä Arduino Yunin suuntaan ja operoitaessa Atheros AR9331 -mikroprosessoria.

Kuvauslaitteiston rakentamisessa haluttiin lähteä liikkeelle omalla kalustolla. Aluksi hankittiin Arduino Yuniin USB-porttiin liitettävä web-kamera sekä microSD-muistikorti mahdollisesti vaadittavaa lisämuistia varten. Kameran vaatimuksena oli, että sen tulisi olla UVC (USB video device class) yhteensopiva, jota monet web-kamerat ovat. Tällä kertaa kameraksi valikoitui edullisuutensa vuoksi Locitech HD Webcam C270. Muistikortiksi valikoitui tähän tarkoitukseen hyvin runsaan muistiominaisuuden omaava 4GB:n microSD-kortti. Näillä komponenteilla lähdettiin rakentamaan videokuvausta ensin omalle Arduinolle ja myöhemmin kuvauksen voisi mahdollisesti toteuttaa robotin käyttöliittymään.

4.2 Työn toteutus

Työtä lähdettiin toteuttamaan Arduino Yunilla, jossa oli jo valmiiksi asennettuna uusin versio OpenWrt-Yunista. Arduino laitettiin myös samaan paikalliseen langatto-

maan verkkoon tietokoneen kanssa, jolloin sen kanssa on helppo toimia ja olla siihen yhteydessä.

Yhteys tietokoneelta Arduinolle muodostettiin SSH (Secure Shell) -yhteydellä, jolla laitteiden välille saadaan salattu ja turvallinen verkkoyhteys. Windows-käyttöjärjestelmällä yhteys voidaan luoda esimerkiksi PuTTY-ohjelmalla, kun puolestaan Macintosh-käyttöjärjestelmässä yhteys voidaan muodostaa Pääte-ohjelmalla. Alla olevassa esimerkissä osoitetaan, kuinka Macbookin Pääte-ohjelmalla yhteyden saa muodostettua melko yksinkertaisesti. Luotaessa yhteyttä on mahdollista käyttää joko Arduinon IP-osoitetta tai Arduinon nimeä, esimerkiksi MyYun.local. Tämän jälkeen syötetään Yunin salasana, jolloin kaiken ollessa kunnossa laitteiden välille on luotu salattu SSH-yhteys.

```
ssh root@192.168.240.1
```

Käydessä ilmi Arduino Yunin muistin rajallisuus videokuvaustarkoitukseen, muistia haluttiin lisätä ulkoisella 4GB:n microSD-muistikortilla, jolloin tilaa olisi varmasti tarpeeksi myöhempää mahdollista tarvetta varten. Muistin lisääminen suoritettiin YunDiskSpaceExpander-sketsin avulla, jolloin muistikortin tila jaetaan halutussa suhteessa Linuxin tiedostojärjestelmälle ja muulle tiedon säilytykselle microSD-kortissa. Taulukossa 1 ilmenee, kuinka muistin laajennuksen jälkeen käytössä on nyt hyvin tilaa ja miten se on jaettu.

Taulukko 1. Muistikortin tilan jako Arduinossa.

```
Filesystem                Size      Used Available Use% Mounted on
rootfs                    1.7G     62.6M      1.6G   4% /
/dev/root                 7.5M      7.5M         0 100% /rom
tmpfs                    29.8M    144.0K     29.7M   0% /tmp
tmpfs                   512.0K         0    512.0K   0% /dev
/dev/sda2                 1.7G     62.6M      1.6G   4% /overlay
overlayfs:/overlay        1.7G     62.6M      1.6G   4% /
/dev/sda1                 1.9G    972.0K      1.9G   0% /mnt/sda1
root@Arduino:~#
```

Videokuvaustarkoitukseen tarvitaan opkg (Open PacKaGe Management) -pakettien asentamista. Kuvaaminen haluttiin suorittaa käyttämällä mjpg-streamer:ia. Kyseinen opkg-paketti ei ole Linux-mikroprosessorissa oletuksena asennettuna. Näin ollen se ja myös muut tarvittavat paketit asennettiin, josta seuraavassa annetaan malliesimerkki:

```
opkg update
opkg install nano
```

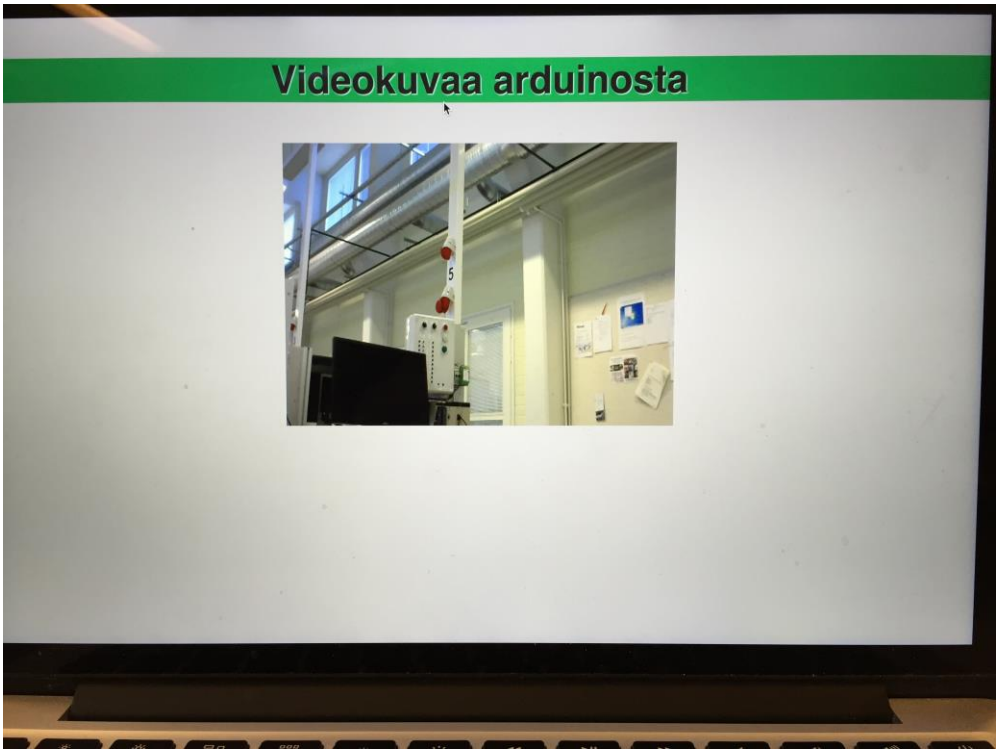
Videokuvaaminen käynnistettiin ja kuvaa näytettiin selaimella. Mjpg-streamer ei ole oletuksena käynnissä, joten sen käyttö sallittiin ja se käynnistettiin. Kamera lähtee kuvaamaan esimerkiksi seuraavalla tavalla:

```
mjpg_streamer -i "input_uvc.so -d /dev/video0 -r 320x240 -f 25" -o
"output_http.so -p 8080 -w /www/webcam"
```

Kuvaa haluttiin esittää itse tehdyllä internetsivulla. Tämä toteutettiin luomalla HTML- ja CSS-tiedostot internetsivua varten. Tiedostot sijoitettiin samaan kansioon, missä myös videokuva on saatavilla. Näin ollen Arduinoon liitetyllä web-kameralla saatiin videokuvaa omatekoiselle internetsivulle.

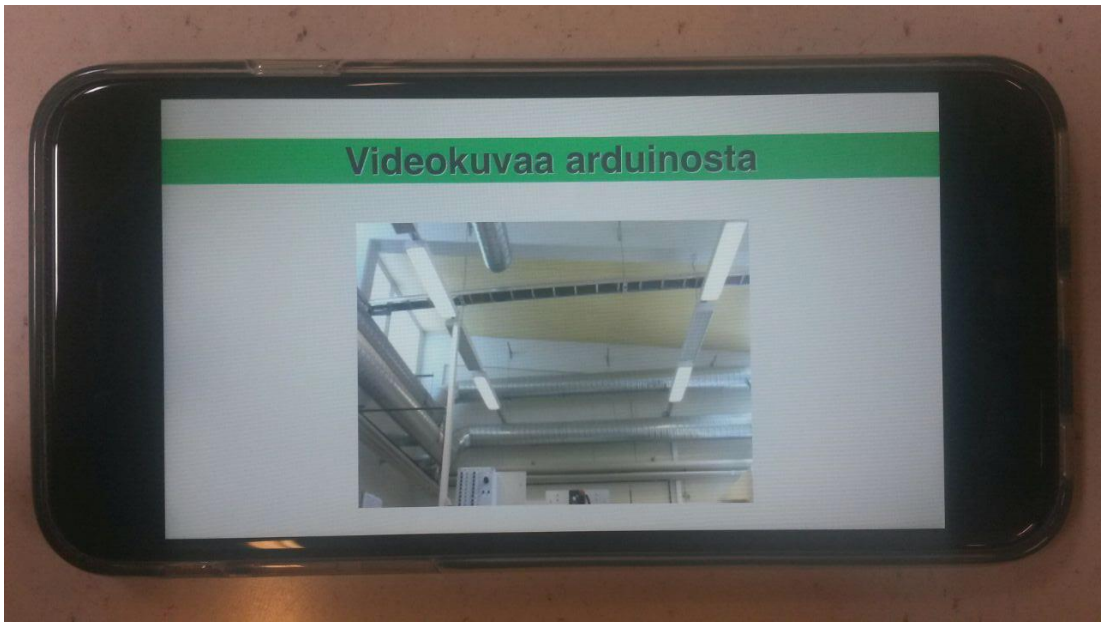
4.3 Videokuvaa internetsivulla

Valmiina tuotteena syntyi internetsivu videokuvan esittämiseen. Web-kameran kuvatessa käyttäjä siirtyy selaimella kuvauksen nettisivulle (kuva 5) ja pääsee näkemään videokuvaa langattomasti Arduinosta. Tuloksena saatiin internetsivulle kohtalaisen hyvänlaatuista videokuvaa, joka on lähes reaaliajassa käyttäjän nähtävillä. Lopuksi videokuva liitettiin myös SAMKRobotin internetsivulle.



Kuva 5. Videokuvauksen internetsivu tietokoneen selaimessa.

Internetsivusta luotiin monikäyttöinen. Sivun toteutuksessa haluttiin huomioida sen toimiminen hyvin eri laitteilla. Kuvasta 6 havaitaan internetsivun näkymä matkapuhelimen selaimella katseltaessa.



Kuva 6. Videokuvauksen internetsivu puhelimen selaimessa.

5 POHDINTA

Opinnäytetyön tavoitteena oli selvittää yksityiskohtaisesti SAMKRobotin toimintaa ja luoda robottiin uusia toimintoja. Työssä oli tavoitteena saada mahdollisimman kattava selvitys robotin toiminnasta, komponenteista ja ohjelmointikoodeista. Opinnäytetyön tavoitteena oli myös luoda robottiin jokin uusi toiminto, joka selkiytyi vasta työn edetessä. Robottiin liitettävä videokuvaustoiminto vaikutti erittäin mielenkiintoiselta, joten web-kameralla kuvaaminen robotin näkökulmasta muodostui hyväksi tavoitteeksi kehittää robottia. Tarkoituksena oli saada videokuvaa robotilta langattomasti selaimelle mahdollisimman reaaliaikaisesti, jolloin laitetta voisi ohjata videokuvan välityksellä.

Opinnäytetyölle asetetut tavoitteet täyttyivät varsin hyvin. SAMKRobotin toiminnasta saatiin luotua monipuolinen selvitys kooditasolle asti. Myös kuvamateriaalin streamaus onnistui käyttäjän selaimelle. Videokuvauksen toiminta ja laatu yllätti opinnäytetyön tekijän positiivisesti, vaikka laadussa pieniä puutteita onkin havaittavissa.

Kuvanlaatu videokuvaksessa ei ole aivan täydellistä, mutta kuitenkin tarkoitukseen ihan riittävää. Myös pientä viivettä ja häiriöherkkyyttä on kuvassa havaittavissa. Paikallisen verkon laatukin näyttäisi vaikuttavan myös kuvanlaatuun. Näihin asioihin voidaan kuitenkin hieman vaikuttaa säätämällä kuvan resoluutiota ja fps (frames per second) -arvoa sekä pitämällä muutenkin Arduino Yunin Linux-mikroprosessorin turha työmäärä mahdollisimman kevyenä. Vaikka Arduino Yun on pienenä tietokoneena loistava laite SAMKRobotin kaltaisissa projekteissa, ei se kuitenkaan vastaa suorituskyvyltään nykypäivän tietokoneita. Videokuvaustarkoituksessa voisi kokeilla esimerkiksi Raspberry Pi 2:ta, joka mahdollisesti tuottaisi Arduino Yunia tehokkaampana laitteena parempilaatuista kuvaa selaimelle. Tällöin videokuvauksessa kannattaisi kokeilla myös laadukkaampaa web-kameraakin. Kuvan siirtämiseen kehitysalustalta tietokoneelle kannattaisi kokeilla WebSocket-protokollaa. Tällöin dataa liikkuisi vähemmän laitteiden välillä ja kuva siirtyisi entistä reaaliaikaisempana, mikä on hyvin tärkeää, jos kuvaa halutaan käyttää robotin ohjaustarkoituksessa.

Opinnäytetyö oli monipuolinen ja kehitti tekijänsä osaamista monissa asioissa. Työn alussa pohjatietoina olivat HTML:n ja CSS:n perusteet sekä hieman käsitystä Javascriptistä. Työn edetessä opinnäytetyön tekijä opiskeli näitä kieliä erilaisten nettikurssien avulla useita kymmeniä tunteja. Vaikka työssä ei varsinaisesti pyritty esittelemään Python-koodia, niin sekin osaaminen lisääntyi opinnäytetyön loppuvaiheilla. Arduino-kehitysalustat ja niiden käyttö olivat alkuun vieraita. Näiden käytöstä päästiin kuitenkin varsin hyvin perille opinnäytetyön ja sen ohella tehtyjen muiden projektien avulla. Myös HTML5 ja WebSocket selkiytyivät tämän opinnäytetyön myötä tekijälleen.

SAMKRobotin kehittämisessä vain mielikuvitus on rajana. Internetistä löytyy paljon englanninkielistä materiaalia ja oppaita Arduino-kehitysalustoihin liittyen. Mielenkiintoisia robotin kehitysehdotuksia voisivat olla esimerkiksi videokuvan tai valokuvien tallentaminen pilvipalveluihin, jolloin ne olisivat nähtävissä ympäri maailman sekä erilaisten paikannusantureiden lisääminen robottiin, jolloin se tunnistaisi paremmin oman ympäristönsä ja sijaintinsa.

LÄHTEET

Arduinon www-sivut. 2015. Viitattu 30.3.2015. http://www.arduino.cc

Banzi, M. 2011. Arduino – Perusteista hallintaan. Norderstedt. O'Reilly.

Karvinen, K. & Karvinen, T. 2010. Sulautetut. Porvoo.WS Bookwell Oy.

Korpela, J. 2014. HTML5-käsikirja. Saarijärven Offset Oy.

Negrino, T. & Smith, D. 2007. Javascript – Tehokas hallinta. Jyväskylä. Gummeruksen kirjapaino Oy.

Pterneas, V. 2013. Getting Started with HTML5 WebSocket Programmng. Birmingham: Packt Publishing. Viitattu 17.4.2015. http://site.ebrary.com.lillukka.samk.fi/lib/SAMK/reader.action?docID=10754108&ppg=1

Vanhalakka, V. 2015. Koneiden vuoro valloittaa internet. Aamulehti 10.5.2015, A26–A27.

W3C www-sivut. 2015. Viitattu 4.4.2015. http://www.w3.org /

W3Schools www-sivut. 2015. Viitattu 20.3.2015. http://www.w3schools.com/