Metropolia Ammattikorkeakoulu
Tietotekniikan koulutusohjelma

Markku Aalto

Videokuvan kaappaus valvomo-sovelluksessa

Insinöörityö 4.5.2010

Ohjaava opettaja: Hannu Laine

Metropolia Ammattikorkeakoulu **Insinöörityön tiivistelmä**

Tekijä Otsikko Sivumäärä Aika	Markku Aalto Videokuvan kaappaus valvomo-sovelluksessa 71 sivua 4.5.2010
Koulutusohjelma	tietotekniikka
Tutkinto	insinööri (AMK)
Ohjaava opettaja	opettaja Hannu Laine

Insinöörityön tavoitteena oli tutustua kuvankäsittelyyn tietokoneessa, Windows-käyttöjärjestelmän tarjoamiin kuvankäsittelymahdollisuuksiin sekä toteuttaa kuvankaappaussovellus teollisuuden prosessien ohjaukseen käytettävällä valvomosovellusalustalla Windowsin API-rajapinnan AVICap-luokkaa apuna käyttäen.

Valvomosovellusten tehtävänä on ohjata ja valvoa teollisuuden tuotanto-, kappaletavaran käsittely-, vedenkäsittely- tai vastaavaa prosessia sekä tallettaa ja analysoida prosessista kerättyä tietoa. Valvonta tapahtuu pääsääntöisesti prosessiin asennettujen mittalaitteiden avulla. Kameravalvontaa voidaan käyttää tilanteissa, joihin ei sopivaa mittalaitetta ole saatavilla. Kameravalvonnan liittäminen valvomosovellukseen tarjoaa mahdollisuuden ohjata kuvankaappausta ja käsitellä talletettua kuvaa prosessin tapahtumien perusteella. Kuvaa voidaan tallentaa esimerkiksi jonkin toimi- tai mittalaitteen aiheuttaman hälytyksen laukaisemana.

Insinöörityössä käytettiin VijeoCitect-nimistä sovellusalustaa, joka tarjoaa työkalut kokonaisen valvomosovelluksen suunnitteluun ja toteutukseen. Sovellusalusta sisältää kaksi ohjelmointikieltä, Cicode ja CitectVBA, joilla kuvankaappaus oli tarkoitus toteuttaa. Cicode on sovellusalustan oma ohjelmointikieli ja CitectVBA VBA-yhteensopiva ohjelmointikieli.

Sovelluksen tehtävänä oli käynnistää ja pysäyttää kuvankaappaus tietyistä tapahtumista. Kuvankaappaus käsitti videoikkunan muodostuksen, tietokoneeseen kytketyn kameran kuvan liittämisen ikkunaan sekä kuvankaappaustiedoston kopioinnin haluttuun tiedostoon. Testitilanteessa kuvankaappauksen käynnistystä ja pysäytystä ohjaavia tapahtumia simuloitiin testaajan toimesta.

Insinöörityön tuloksena saatiin aikaan sovellus, joka kykeni edellä mainittuihin tehtäviin. Muutamissa testitapauksissa tosin havaittiin järjestelmän pysähtymistä, eli järjestelmä lopetti lyhyeksi ajaksi käyttäjän komentoihin vastaamisen. Sovellusta on vielä siis testattava ja kehitettävä ennen kuin sitä voidaan loppuasiakkaan valvomosovelluksessa käyttää.

Insinöörityön avulla tekijälle tuli kuitenkin runsaasti lisää tietoa kuvankäsittelystä, kuvankaappausprosessista Windows-käyttöjärjestelmässä ja siitä, mitä ominaisuuksia sovellukselta vaaditaan ollakseen toimiva kuvankaappaussovellus. Näiltä osin insinöörityön tavoitteet toteutuivat.

Hakusanat	kuvankaappaus, avicap, valvomosovellus, api, vba

Helsinki Metropolia University of Applied Sciences Abstract

Author	Markku Aalto
Title	Video capture in a process control application
Number of Pages	71 pages
Date	4 May 2010
Degree Programme	Information Technology
Degree	Bachelor of Engineering
Supervisor	Hannu Laine, Lecturer

The purpose of this thesis was to study how computers manipulate image and video data, what kind of possibilities the Windows operating system offers for image and video processing and to carry out a video capture application program on the process control platform using the Windows AVICap class.

Process control application programs are intended to control and monitor manufacturing, product handling, water treatment or similar processes. In most cases the control of a process is carried out by measurement instruments. In situations with no proper instrument available, video control might be a solution. Integration of video capture and a process control application program enables direct control of video capture devices by a process control system.

The application program of this thesis was designed to start and stop video capturing by certain events of the system. In this context video capture means creation of a video capture window, connection between camera and window, and copying of the video capture file to a desired file. In test situations start and stop events were controlled by a test person.

The goals of this thesis were only partly met as regards the application program. Most functions worked properly, but in some test cases the computer did not operate correctly. The application program is not ready for use until it has been further tested and developed.

The thesis process widened the author's knowledge about image and video processing by computer and the Windows operating system. In this regard the goals of the thesis were reached.

Keywords	video capture, avicap, application program, api, vba

Sisällys

Lyhenteet

API	*Application Programming Interface;* ohjelmointirajapinta tai käyttöliittymä, jolla ohjelmat voivat vaihtaa tietoja keskenään.
DVD	*Digital Video Disc;* Optinen datan tallennusväline.
DVI	*Digital Video Interface;* Videosignaalin liitäntätapa.
GSM	*Global System for Mobile communications;* maailmanlaajuinen matkapuhelinjärjestelmä.
GUI	*Graphical User Interface;* graafinen käyttöliittymä.
HDMI	*High Definition Multimedia Interface;*Videosignaalin liitäntätapa.
HDTV	*High Definition Television;* Teräväpiirtotelevisio.
LCD	*Liquid Crystal Display;* Nestekidenäyttö.
PLC	*Programmable Logic Controller;* ohjelmoitava laite, jota käytetään automatisoitujen prosessien ohjaukseen ja valvontaan.
RS	*Recommended Standard;* sarjamuotoisessa tietoliikenteessä käytettävien liitinstandardien tunnus.
SCADA	*Supervisory Control And Data Acquisition;* tietokoneohjelmisto, jolla ohjataan ja valvotaan automatisoitua prosessia.
VBA	*Visual Basic for Applications;* Windows käyttöjärjestelmässä toimivissa sovelluksissa käytettävä ohjelmointikieli.
VGA	*Video Graphics Array;* Tietokoneen näyttöstandardi.
WLAN	*Wireless Local Area Network;* langaton lähiverkko.

1 Johdanto

Tämän opinnäytetyön tarkoituksena on tutustua kuvan ja äänen tallettamiseen ja käsittelyyn tietokoneessa, tutkia miten Windows-käyttöjärjestelmässä voidaan toteuttaa videokuvan kaappaussovellus, sekä toteuttaa kaappaussovellus valvomosovellus-alustalla.

Valvomosovellusten tehtävänä on ohjata ja valvoa teollisuuden tuotanto-, kappaletavaran käsittely, vedenkäsittely- tai vastaavaa prosessia, sekä tallettaa ja analysoida prosessista kerättyä tietoa. Valvonta tapahtuu pääsääntöisesti prosessiin asennettujen mittalaitteiden avulla. Monissa prosesseissa on kuitenkin osia tai laitteita, joihin ei löydy sopivaa mittalaitetta tai anturia, jolloin prosessin osan valvonta jää visuaalisten, usein prosessin käytöstä vastaavien henkilöiden havaintojen varaan. Henkilöiden käyttäminen jonkin prosessin osan valvontaan on kuitenkin kallista tai se voi olla hankalaa. Videokameroiden avulla valvontaa saadaan keskitettyä ja henkilöresursseja varattua tärkeämpiin tehtäviin.

Videokuvan kaappaus valvomosovelluksessa on valittu opinnäytetyön aiheeksi siksi, että kuvan kaappauksen liittämisellä valvomosovellukseen kuvan käsittely saadaan liitettyä samaan järjestelmään, jolla prosessia valvotaan ja ohjataan. Tällä taas saavutetaan se etu, että kameran ohjaaminen prosessin tapahtumien perusteella saattaa olla tällä tavalla helpompaa kuin erilliseen laitteistoon liitettävä kamera.

Toinen peruste on se, että sovellusalustoissa ei ole ollut valmiita ratkaisuja kuvankaappauksen toteuttamiseen. Sovellusalustoissa on kuitenkin mahdollisuuksia, niiden ohjelmointityökaluja ja käyttöjärjestelmän ominaisuuksia hyväksikäyttäen, toteuttaa prosessin ohjaukseen integroituja kuvan käsittely- ja tallennustoimintoja. Valvomosovelluksissa tosin on ominaisuuksia, joiden avulla voidaan käynnistää muita tietokoneeseen asennettuja ohjelmia, esimerkiksi ActiveX-komponentteja, joiden tehtävänä on ohjata tietokoneeseen liitettyä kameraa. Kolmannen osapuolen tekemien ohjelmien käytön haittapuolena on se, että niiden toiminta ei välttämättä vastaa

prosessin tarpeita, eikä niiden muokkaaminen prosessin tarpeiden mukaan ole välttämättä mahdollista.

Kuvankaappauksen lisäksi kuvan käsittelyssä ja analysoinnissa löytyy paljon kehittämismahdollisuuksia. Esimerkkinä kuvan analysoinnista voidaan mainita jäteveden käsittelylaitos, jossa kuvan perusteella voidaan seurata vaahdon muodostumista veden pinnalle.

Veden pinnalle muodostuva vaahto aiheuttaa usein ongelmia ja sen poistamiseen joudutaan käyttämään kemikaaleja. Vaahdon havaitseminen mittalaitteilla on kuitenkin vaikeaa, joten altaisiin syötetään vaahdonestokemikaaleja usein turhaan.
Veden pinnan korkeutta voidaan mitata veden aiheuttamaa hydrostaattista painetta mittaamalla. Veden pinnalle muodostuva vaahto ei kuitenkaan muuta painetta, joten paineen mittaus ei sovi tähän tarkoitukseen. Veden pinnan korkeutta mitataan myös ultraäänellä. Ultraääni kuitenkin läpäisee vaahdon, jolloin vaahto ei näy myöskään ultraäänimittauksessa. Vaahto kuitenkin on usein eriväristä kuin vesi, joten sen havaitseminen tietokoneeseen talletettua kuvaa analysoimalla saattaisi olla mahdollista. Tällä tavalla vaahdonestokemikaaleja voitaisiin syöttää altaisiin vain silloin, kun vaahtoa muodostuu veden pinnalle. Näin säästettäisiin veden käsittelykustannuksia sekä päästettäisiin luontoon sitä kuormittavia kemikaaleja vähemmän.

Insinöörityö on tehty työnantajalleni SoftBase Oy:lle. SoftBase Oy on Espossa sijaitseva, vuodesta 1985 asti toiminut teollisuusautomaation suunnitteluun ja toteutukseen keskittynyt suunnittelutoimisto.

2 Valvomosovellukset

Valvomosovelluksia kutsutaan myöskin SCADA-järjestelmiksi. SCADA (Supervisory Control And Data Acquisition) -nimitystä tosin käytetään Suomesssa harvemmin. Valvomosovelluksella tarkoitetaan tietokoneessa toimivaa sovellusta, jonka tehtävänä on toimia erilaisissa teollisuus- tai muissa prosesseissa prosessin valvonta-, ohjaus- ja tiedon hallintajärjestelmänä. Valvomosovellukset toimivat yleisimmin Windows-käyttöjärjestelmässä, ja ne toteutetaan räätälöidysti prosessin tarpeiden mukaan jonkin sovellusalustan tarjoamilla työkaluilla.

Sovellusalustat sisältävät useita etukäteen tehtyjä komponentteja, joita voidaan käyttää hyödyksi sovelluskohtaisesti tarpeen mukaan. Sovellusalustat sisältävät yleensä piirtotyökalut graafisen rajapinnan toteuttamiseen, valmiit komponentit hälytysten ja muun talletettavan tiedon käsittelyyn sekä kommunikointiajurit ulkopuolisiin järjestelmiin. Valvomosovellukset eivät yleensä suoraan ole kosketuksissa itse prosessiin, vaan sovelluksen ja prosessin välissä on jokin erillinen ohjelmoitava laite, kuten ohjelmoitava logiikka, jonka välityksellä siirretään tietoa valvomosovelluksen ja prosessin välillä.

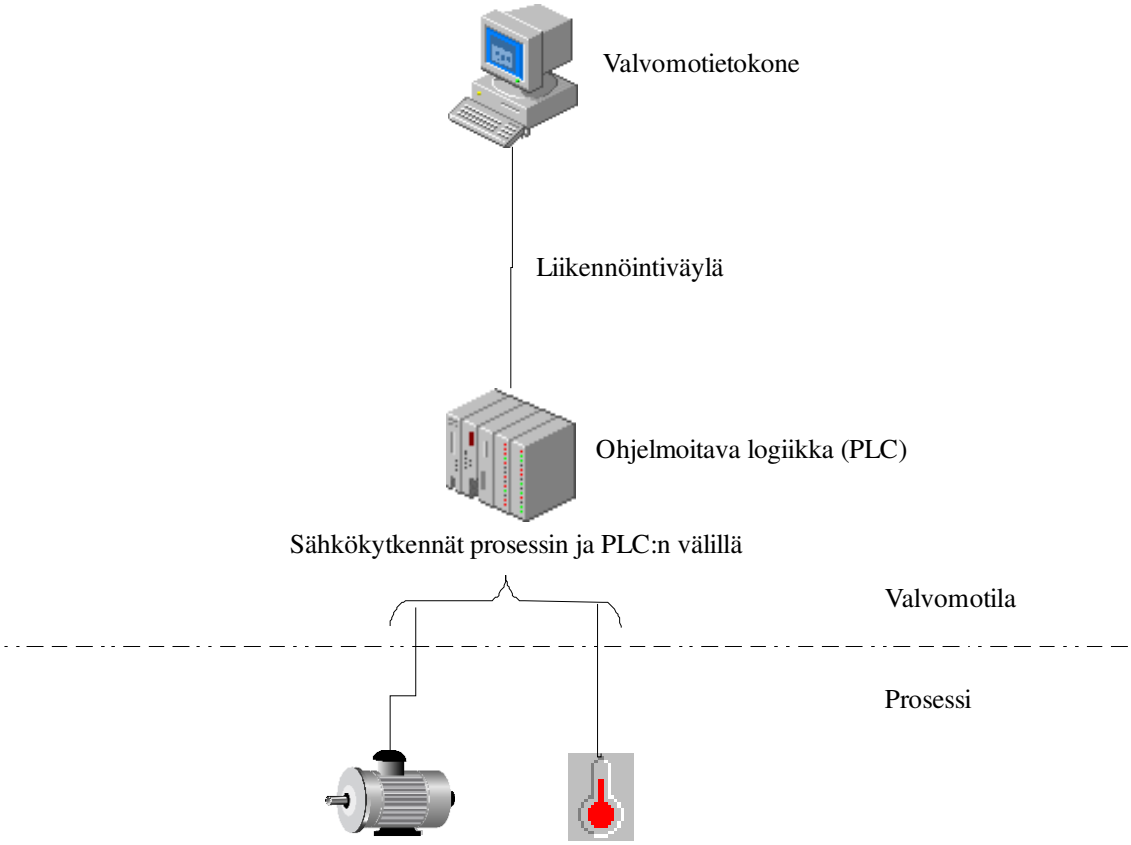
Tässä opinnäytetyössä käytettävä sovellusalusta on VijeoCitect, versio 7.0. Käyttöjärjestelmänä toimii Windows XP Pro -käyttöjärjestelmä. Sama sovellusalusta toimii myös Vista käyttöjärjestelmässä.

2.1 Valvomosovellusten yleiset ominaisuudet

Valvomosovellusten pääasiallisina tehtävinä ovat kommunikointi prosessin kanssa, graafisena rajapintana toimiminen käyttäjien ja prosessin välillä sekä prosesseista saatavien hälytys-, historia- ja tapahtumatietojen kerääminen, käsittely ja talletus.

Graafinen rajapinta (Graphical User Interface, GUI) koostuu ikkunoista, joiden sisältämiä komponentteja käytetään tietojen välittämiseen prosessin ja käyttäjien välillä. Näitä komponentteja ovat muun muassa mittaustietojen esittämiseen käytettävät numerokentät, historiatietojen esittämiseen käytettävät trendinäytöt, eri laitteiden tilojen esittämiseen käytettävät tekstikentät ja graafiset symbolit sekä käyttäjien käyttämät tekstin syöttökentät ja painikkeet.

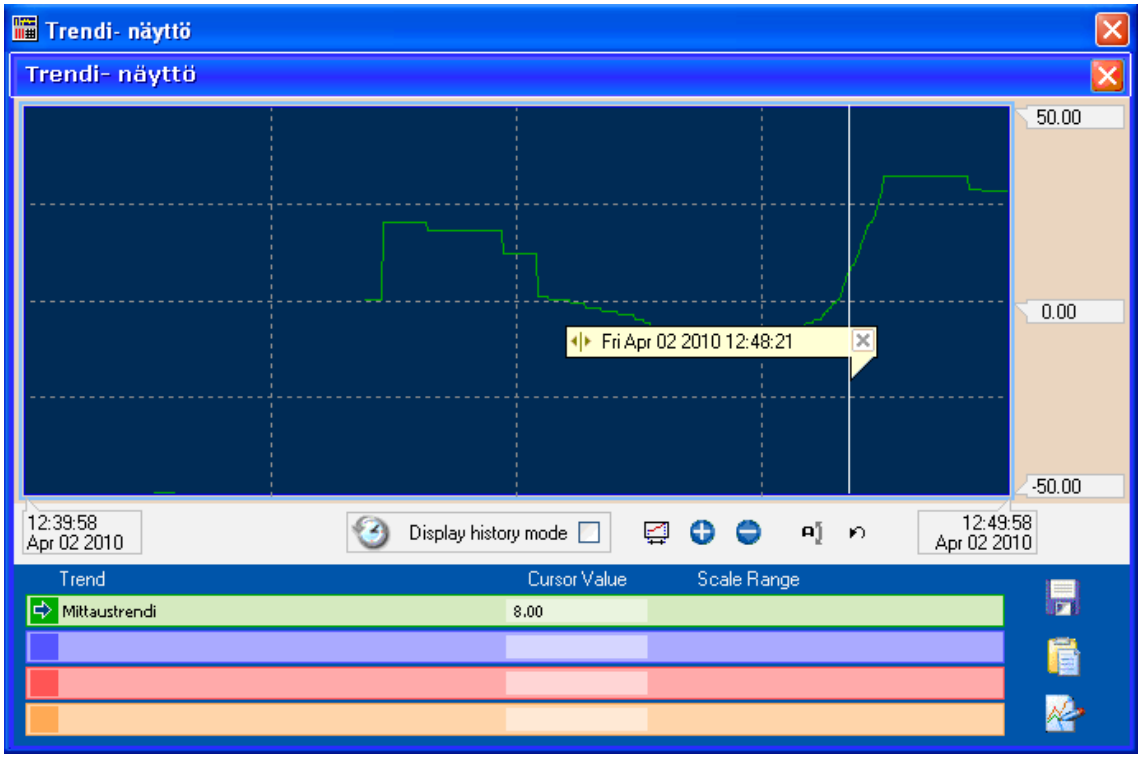
Valvomosovellukset tai tietokoneet, joissa sovellukset toimivat, eivät yleensä suoraan kommunikoi itse prosessin tai sen mitta- ja toimilaitteiden kanssa. Valvomosovellus on tällöin liitetty johonkin prosessia ohjaavaan laitteeseen, kuten ohjelmoitavaan logiikkaan (Programmable Logic Controller, PLC). Valvomosovellus kommunikoi jonkin liikennöintiprotokollan avulla ja jonkin liikennöintiväylän välityksellä prosessin ohjauslaitteen kanssa. Tällaisia kommunikointiprotokollia ovat muun muassa Modbus, Profibus ja CANopen. Liikennöintiväylinä käytetään yleensä sarjaliikenneväyliä, kuten RS-485/ 422 tai RS-232, sekä ethernet-väylää tai langatonta tiedonsiirtoa, kuten GSM- tai WLAN-verkkoa. Kuvassa 1 esitetään tyypillinen yhden valvomotietokoneen, PLC:n ja prosessin välisen yhteyden periaate. Valvomotietokoneita ja prosessia ohjaavia laitteita voi olla samassa prosessissa toisiinsa liitettynä useita ja näin suurissa prosesseissa usein onkin.



Kuva 1. Valvomotietokoneen ja prosessin välinen yhteys.

Valvomosovellusten tehtävänä on myös kerätä prosessista tietoa ja tallettaa sitä myöhempää tarkastelua varten. Tietoa voidaan tallettaa eri muodoissa aina tekstitiedostoista tietokantoihin asti. Jotkin sovellusalustat sisältävät myös tietokantapalvelimen.

Prosessiin kuuluvien mittausten historiatiedon tallennus on tärkeää valvomo-sovelluksissa. Mittausten historiatietoa halutaan usein jälkikäteen tarkastella esimerkiksi analysoitaessa syitä prosessissa esiintyviin hälytyksiin tai muihin ongelmiin. Mittaustiedot esitetään usein graafisessa muodossa trendinäyttönä. Kuvassa 2 on erääseen prosessiin liitettyjen mittausten trendinäyttö.



Kuva 2. Trendinäyttö.

Esimerkkinäyttöön on liitetty yksi mittaus, jonka arvo voi vaihdella välillä -50 – 50. Ikkunassa on alue, johon mittausarvo piirretään kuvaajalla. Sen lisäksi siinä on painikkeet aika- ja arvoakseleiden skaalausta varten. Piirtoalueeseen saadaan myös kohdistin, jonka avulla voidaan seurata mittausarvoa tarkasti jollain tietyllä ajan hetkellä.

Valvomosovelluksissa on myös toimintoja prosessissa tapahtuvien hälytysten käsittelyyn. Uudet hälytykset esitetään sovelluksessa reaaliaikaisena niiden tultua voimaan, niin graafisesti kuin tekstinäkin. Hälytysten käsittelyssä on kuittausominaisuus, jolla käyttäjä ”kertoo” sovellukselle havainneensa hälytyksen. Hälytykset usein myös kategorioidaan niiden kiireellisyyden perusteella. Hälytykset on voitava myös tallettaa myöhempää tarkastelua varten. Hälytykset voidaan tallettaa esimerkiksi tietokantaan kuten mittaustiedot. Hälytyksistä talletetaan usein sen nimen lisäksi, kategoria, hälytyksen voimaantulo-, poistumis- ja kuittausaika sekä hälytyksen kuitanneen käyttäjän tiedot.

2.2 VijeoCitect

VijeoCitect-sovellusalustalla on kaksi päätehtävää: tarjota työkalut sovellusten kehittämiseen ja mahdollistaa sovelluksen ajaminen run time -tilassa. Run time -tilalla tarkoitetaan tilaa, jossa sovellusta käytetään loppukäyttäjän prosessin valvontaan ja ohjaukseen. Alustalla voi olla luotuna useita sovelluksia, mutta run time -tilassa voidaan ajaa vain yhtä sovellusta kerrallaan.

VijeoCitectin run time -kokoonpano koostuu palvelimista ja asiakkaista. Jokainen sovellus vaatii toimiakseen vähintään yhden palvelimen ja yhden asiakkaan. Palvelimen tehtävänä on kommunikoida prosessin kanssa ja asiakkaan tehtävänä on toimia käyttöliittymänä. Suurissa sovelluksissa palvelimia ja asiakkaita voi olla useita. Palvelin ja asiakas voivat olla samassa tietokoneessa.

Run time -tilan ajaminen vaatii sovellusalustan lisäksi erillisen lisenssin, jolla määritellään projektissa käytettävien muuttujien maksimimäärä. Tähän määrään lasketaan mukaan muuttujat, joiden kautta välitetään tietoa jonkin ulkopuolisen laitteen tai järjestelmän, esimerkiksi ohjelmoitavan logiikan tai tietokannan, ja VijeoCitect-sovelluksen välillä. Jokainen palvelin ja asiakas vaativat toimiakseen oman lisenssin. Lisenssit ovat maksullisia, ja niiden hinta määräytyy tarvittavan muuttujamäärän mukaan.

Sovellusten luonti ja kehitystyö sekä run time -tilassa ajaminen vaativat siis VijeoCitect-sovellusalustan. Sovellus voidaan kuitenkin luoda ja kehitystyö toteuttaa eri koneissa, kunhan molemmissa tietokoneissa on VijeoCitect asennettuna. Kannattaako sovelluksen kehitystyö tehdä samassa koneessa, kuin sitä ajetaan, riippuu eri tekijöistä. Graafisen käyttöliittymän suunnittelu on usein järkevää tehdä loppukäyttäjän koneella, koska käyttöliittymään vaikuttavat monet tietokoneesta riippuvat tekijät, kuten näyttö ja näytön ohjain. Sovelluskehittäjällä ei välttämättä ole mahdollisuutta hankkia samanlaista laitteistoa kuin loppukäyttäjällä on käytössään, joten sovelluksen käyttöliitymän toteutus loppukäyttäjän tietokoneella saattaa säästää aikaa.

2.2.1 VijeoCitectin työkalut

Graphics Builder -työkalulla luodaan sovelluksen graafinen rajapinta. Graphics Builder sisältää piirtotyökalut graafisten objektien luontia varten.

Toinen VijeoCitectin sisältämä työkalu on Project Editor, joka nimensä mukaisesti toimii projektin editointityökaluna. Tällä työkalulla toteutetaan graafista rajapintaa lukuun ottamatta kaikki projektiin liittyvä työ.

Citect Explorer on kaikkien tietokoneessa olevien VijeoCitect-projektien hallintatyökalu. CitectExplorerilla muun muassa luodaan projektit, määritellään run time -tilassa käynnistettävä projekti sekä määritellään valvomotietokoneen yleistä käyttäytymistä.

VijeoCitectin neljäs työkalu on nimeltään Cicode editor, jolla sovellussuunnittelija voi kirjoittaa omaa ohjelmakoodia sovellusta varten. VijeoCitect sisältää satoja valmiita sisäänrakennettuja funktioita, joita sovelluksessa voidaan kutsua. Sovellussuunnittelija voi sen lisäksi kirjoittaa vapaasti omia funktiotaan sovellusta varten. Myös näitä funktioita voidaan kutsua samalla lailla kuin valmiita sisäänrakennettuja funktioita.

Ohjelmakoodia voidaan kirjoittaa joko VijeoCitectia varten kehitetyllä Cicode-kielellä tai yleisemmin tunnetulla VBA-kielellä (Visual Basic for Applications). VijeoCitectissa VBA-kieltä kutsutaan nimellä CitectVBA. Sovelluksessa voidaan käyttää molempia kieliä yhtäaikaa tai erikseen. Myös toisella kielellä tehtyjä funktioita voidaan kutsua toisella kielellä tehdyssä ohjelmakoodissa. Kielet eroavat toisistaan monella tavalla, joten miten tahansa kieliä ei voi sekoittaa eikä eri kielillä tehtyä koodia voida sijoittaa samaan tiedostoon.

2.2.2 Käyttöliittymä

VijeoCitect-sovelluksen käyttöliittymä on graafinen ja se toteutetaan Graphics Builder-työkalulla. Käyttöliittymän perustana toimivat sivut, joilla pääsääntöisesti esitetään prosessia tai sen osaa, hälytyslistaa ja muuttujista talletettuja historiatietoja. Prosessin koosta ja laajuudesta riippuen sovellus voi sisältää yhden tai useamman sivun.

Sivujen sisältö koostuu graafisista objekteista. Näitä objekteja ovat muun muassa viivat ja erilaiset kuviot, kuten ympyrät ja monikulmiot, painonapit, teksti- ja numerokentät, symbolit, trendinäytöt sekä ActiveX-komponentit.

Objektit voidaan jakaa edellä mainitun tyyppijaon lisäksi niiden toiminnallisuuden perusteella staattisiin ja dynaamisiin objekteihin. Staattisella objektilla tarkoitetaan objektia, jonka ulkoasu pysyy samanlaisena prosessin tilasta riippumatta ja joka ei reagoi mihinkään käskyihin tai tapahtumiin. Dynaaminen objekti sisältää joko johonkin tapahtumaan, esimerkiksi hiiren painallukseen, liittyvän toiminnon tai sen ulkoasu muuttuu prosessin mukaan.

Objekteille voidaan määritellä eri ominaisuuksia niiden tyypistä riippuen. Yleisesti ottaen kaikkien objektien ominaisuuksiin kuuluvat värit ja 3D-efektit. 3D-efekteillä objekteihin voidaan luoda varjostuksia ja korostuksia.

Objekteille voidaan määritellä ilmentymiä (eng. expression), joilla objekteihin saadaan dynamiikkaa. Tällöin objektiin liitetään jokin sovellukseen määritelty muuttuja, jonka tilan perusteella objektin ominaisuuksia voidaan muuttaa. Yleisesti erityyppisten objektien ominaisuuksia voidaan muuttaa seuraavasti:

- Objektin näkyvyyttä voidaan muuttaa piilottamalla objekti.
- Objektin paikkaa sivulla voidaan muuttaa siirtämällä sitä vaaka- ja/tai pystysuunnassa, esimerkkinä porttiventtiilin portin paikka.
- Objektia voidaan kääntää halutun kiintopisteen suhteen, esimerkkinä

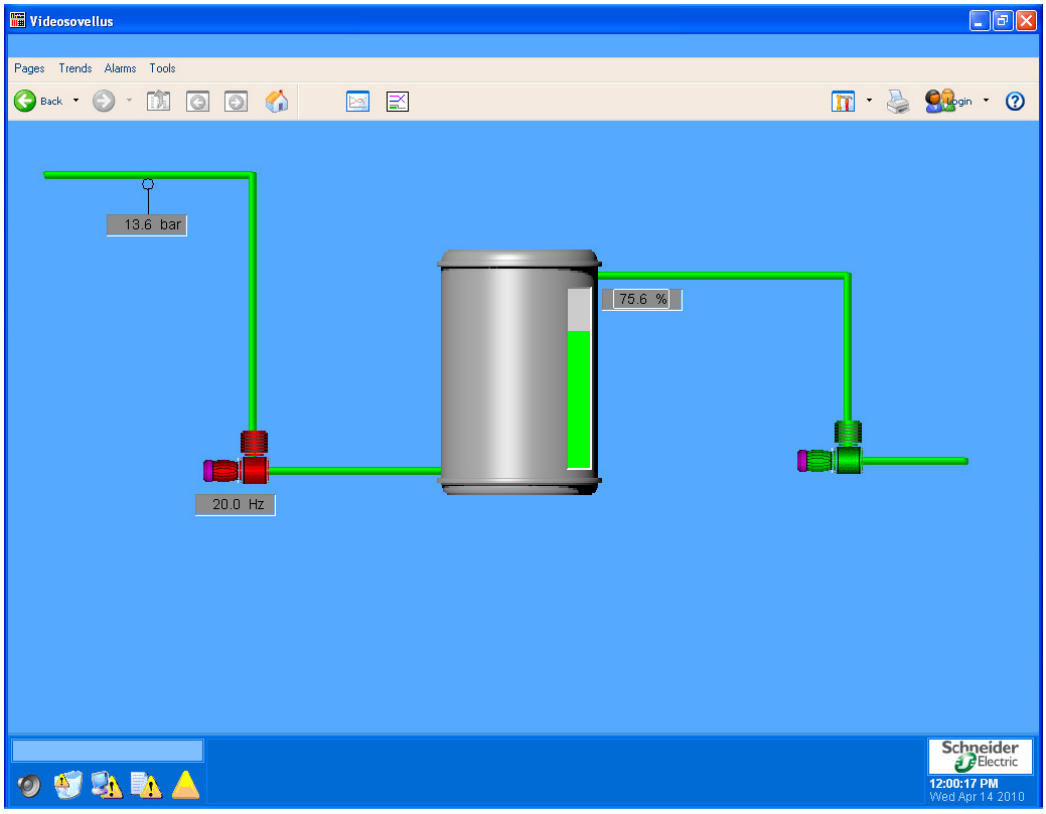
nopeusmittarin viisari tai läppäventtiilin läpän asento.

- Objektin kokoa voidaan muuttaa vaaka- ja/tai pystysuunnassa. Objektin paikka muutoksessa voidaan määritellä halutulla kiintopisteellä, esimerkkinä iiris-venttiilin aukon koko.
- Objektin väriä voidaan muuttaa niin, että koko objektin väri vaihtuu tai niin, että objektissa on kaksi väriä, joiden osuus objektin alasta on riippuvainen siihen liitetyn muuttujan tilasta, esimerkkinä vesisäilön täyttöastetta kuvaava palkki.
- Tekstikentän tekstiä voidaan muuttaa näyttämään jokin mittausarvo.
- Symbolin esittämää kuvaa voidaan vaihtaa, esimerkkinä moottorin tilaa (käy/ seis/ hälytys) esittävät kuvat.

Objekteille voidaan myös liittää käskyjä tai tapahtumia, joiden toteuduttua voidaan kutsua sovelluksessa olevaa funktiota tai muuttaa jonkin muuttujan tilaa. Tapahtumia ovat:

- Hiiren painikkeiden painallukset. Esimerkkinä moottorin ohjauspainikkeet tai hälytyksen kuittaus hälytystekstiä klikkaamalla.
- Hiirellä tehdyt liukukomennot (eng. slide) symbolin paikkaa muuttamalla. Esimerkkinä moottorin nopeusohjeen muutos.
- Näppäimistön komennot. Esimerkkinä moottorin nopeusohjeen syöttäminen näppäimistöltä.

Kuva 3 esittää erästä prosessisivua, jossa esiintyy edellä mainittuja komponentteja. Kuvassa pumppujen symbolit ovat dynaamisia. Pumppujen tila indikoidaan värillä. Tässä tapauksessa punainen väri tarkoittaa pumpun vikaa ja vihreä käynnissä olevaa pumppua. Vesisäiliö on staattinen objekti. Säiliön täyttöaste, tyhjennyspumpun nopeuden ja putkiston paineen numerokentät ovat mittaustietojen näyttöä. Vesisäiliön oikeassa laidassa oleva graafinen palkki kertoo myös säiliön täyttöasteen.



Kuva 3. Prosessisivu.

2.2.3 Tagit ja I/O-yksiköt

Tieto prosessin ja valvomosovelluksen välillä välitetään muuttujissa. VijeoCitect-sovelluksessa käytettäviin muuttujiin tulee määritellä niin sanottu tagi, jolla muuttuja tunnistetaan sovelluksessa. Sovelluksen tagit voidaan jakaa varsinaisiin muuttujatageihin, paikallisiin muuttujiin, trenditageihin ja SPC-tageihin. SPC-tagit liittyvät VijeoCitectin tilastointiominaisuuksiin, eikä niitä tässä opinnäytetyössä käsitellä tämän enempää.

Muuttujatagien määrittelyyn liittyy oleellisesti I/O-yksikkö, johon tagit itse asiassa liitetään. Jokaiselle tagille on varattu I/O-yksikössä osoite, johon tagimäärittelyssä viitataan. Tagimäärittelyn ansiosta sovelluksen suunnittelijan ei tarvitse muistaa, mihin I/O-yksikköön muuttuja kuuluu tai mikä on sen osoite.

I/O-yksiköt ja tagit määritellään Project Editor -työkalun avulla. I/O-yksikkö voi olla esimerkiksi tietokoneen sisäinen I/O-yksikkö tai valvomotietokoneeseen liitetty ohjelmoitava logiikka. Yksi sovellus voi sisältää useamman kuin yhden I/O-yksikön. Hyvin usein sovellukseen määritellään tarpeen mukaan yksi tai useampi ulkopuolinen I/O-yksikkö ja yksi sisäinen I/O-yksikkö. Sisäiseen I/O-yksikköön liitettyjä tageja ei huomioida, kun määritellään sovelluksen run time -lisenssin laajuutta.

I/O-yksikön määrittelyyn liittyy olennaisena osana väyläprotokolla, jonka välityksellä sovellus ja I/O-yksikkö kommunikoi. Eri laitevalmistajat varustavat laitteistonsa oletusarvoisesti omalla väyläprotokollallaan, jolloin eri vaihtoehtoja on markkinoilla jopa useita satoja. Esimerkkinä voidaan mainita Schneider Electricin edustamat laitteet, jotka tukevat lähes aina Modbus RTU -liityntää, ja Siemensin laitteet, joissa on usein Profibus DP -liityntä vakiona. VijeoCitect sisältää useita kymmeniä väyläprotokollia eri valmistajien laitteistojen kanssa käytettäviksi. VijeoCitect sisältää tunnetuimmat protokollat, mutta tilanteessa, jossa protokollaa ei löydy, se on hankittava erikseen esimerkiksi laitevalmistajalta.

Väyläprotokollan perusteella määritellään siihen I/O-yksikköön liitettyjen tagien muuttujatyypit, osoitteet ja osoitteiden syntaksi. Eri protokollissa käytettävät syntaksit saattavat erota hyvinkin paljon toisistaan. Modbus RTU -protokollan osoitteet ovat numeerisesti esitetty, kun taas Telemecaniquen Unitelway-protokollan osoitteet alkavat %-merkillä, jota seuraa M ja muuttujan tyyppiä kuvaava merkki. Esimerkkinä muistipaikassa 0 oleva 16-bitin kokonaislukumuuttuja, jonka osoite merkittäisiin %MW0.

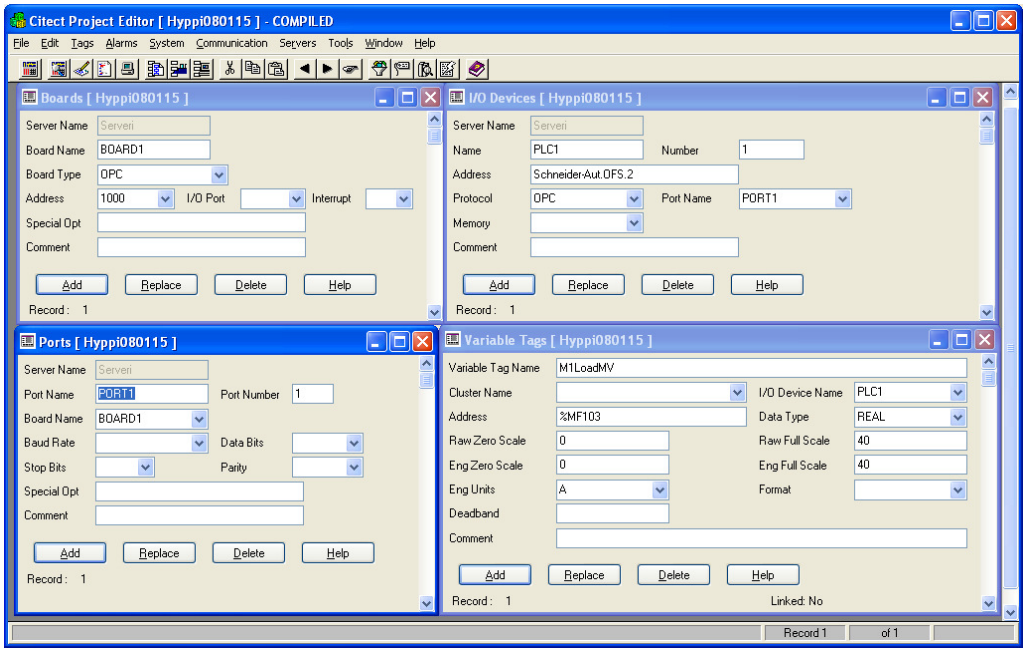
I/O-yksiköiden ja osoitteiden lisäksi jokaiselle muuttujatagille tulee määritellä vähintään yksilöllinen nimi ja muuttujatyyppi. Näiden lisäksi muuttujatageille voidaan määritellä skaalausarvot ja mittayksikkö, joka käyttöliittymässä näytetään, esimerkiksi veden tilavuusvirran mittayksikkö voisi olla l/s, litraa sekunnissa.

Paikalliset muuttujat ovat muuttujia, joita ei liitetä mihinkään I/O-yksikköön. Niillä ei näin ollen ole myöskään osoitetta, johon ne viittaisivat. Muuttujatagien tavoin niille

tulee määritellä tyyppi. Tarvittaessa paikallisille muuttujille voidaan määrittää skaalausarvot ja mittayksikkö.

Trenditagit ovat muuttujatagien ilmentymiä, joiden avulla voidaan esittää muuttujista talletettua historiatietoa graafisessa muodossa trendinäytöllä. Trenditagien pakolliset määrittelyt ovat viittaus siihen muuttujatagiin, jota trenditagi ilmentää sekä tapa jolla muuttujatagin tilaa talletetaan. Muuttujatagin tila voidaan tallettaa aikaperusteisesti, tapahtumaperusteisesti tai näiden kombinaationa. Aikaperusteisessa tallennuksessa muuttujan tila talletetaan näytteenottovälillä määritellyn ajanjakson välein. Tapahtumaperusteisessa tallennuksessa trenditagille määritellään laukaisin eli muuttujatagi, jonka tila määrää tallennushetken.

Kuva 4 esittää erään valvomosovelluksen ja prosessia ohjaavan laitteen välisen yhteyden määrittelyn sekä prosessin ohjauslaitteeseen liitetyn muuttujatagin.



Kuva 4. Valvomon ja prosessia ohjaavan laitteen välisen yhteyden määrittely.

2.2.4 Hälytykset

Hälytykset ovat jonkin muuttujan tai usean muuttujan yhtäaikaisen ei-toivotun tilan ilmentymiä. VijeoCitectissa hälytykset määritellään Project Editor -työkalulla. Hälytykselle määritellään hälytystagi, joka liitetään muuttujatagiin tai -tageihin, joiden tilojen perusteella hälytys muodostetaan. Hälytystagille määritellään muuttujatagin lisäksi nimi ja kuvaus, joilla se esitetään ja tunnistetaan käyttöliittymän hälytyslistassa.

VijeoCitectissa hälytys voidaan muodostaa digitaalisen tai analogisen muuttujatagin perusteella. Digitaalisessa hälytyksessä hälytystagi liitetään digitaaliseen muuttujatagiin ja hälytys voidaan määritellä muuttujatagin ollessa joko tosi tai epätosi. Digitaalinen hälytys voidaan muodostaa myös useamman digitaalisen muuttujatagin kombinaationa.

Analoginen hälytys muodostetaan liittämällä hälytystagiin muuttujatagi, jonka tyyppi on kokonais- tai reaaliluku. Sen lisäksi analogiselle hälytykselle määritellään raja-arvo, johon muuttujatagia hälytystä muodostettaessa verrataan. Raja-arvo voi olla myös muuttujatagi. Yhdelle analogiselle hälytystagille voidaan määritellä kaksi yläraja-arvoa (high ja high high), joiden perusteella muodostetaan hälytys, kun muuttujatagin arvo ylittää raja-arvon. Sen lisäksi analogiselle tagille voidaan määritellä kaksi alaraja-arvoa (low ja low low), joiden perusteella muodostetaan hälytys, kun muuttujatagin arvo alittaa raja-arvon.

Mittausarvosta voidaan muodostaa hälytys myös I/O-yksikössä, jolloin arvon ylittäessä yläraja-arvon tai alittaessa alaraja-arvon, ohjataan I/O-yksikössä digitaalista muuttujaa, ja tämän muuttujan tilan perusteella valvomosovelluksessa muodostetaan digitaalinen hälytys. Näin usein myös käytännössä tehdään, koska mittausarvoista muodostettavilla hälytyksillä, kuten hälytyksillä yleensäkin, on prosessin ohjauksessa jokin tehtävä. Esimerkiksi vesisäiliön pinnan ollessa alhainen hälytys pysäyttää pumpun ja tällä tavalla estää pumpun ns. kuivakäynnin ja vaurioitumisen.
Syy miksi edellä mainitussa tilanteessa hälytys kannattaa muodostaa I/O-yksikössä eikä valvomosovelluksessa, on se, että sekä mitta- että toimilaitteet ovat kytketty I/O-

yksikköön, jolloin ei-toivottuun tilaan reagoiminen on nopeampaa ja varmempaa. Valvomosovelluksen ja I/O-yksikön välisen kommunikaation toimiessa huonosti tai ei ollenkaan, ei valvomosovelluksessa muodostettavilla hälytyksillä voida vaikuttaa prosessin ohjaukseen.

2.2.5 Tapahtumat

Tapahtumia, joista VijeoCitectissa käytetään nimitystä events, käytetään suorittamaan toimenpiteitä ilman käyttäjän suorittamaa käskyä. Käyttöliittymää koskevassa luvussa puhuttiin tapahtumista ja käskyistä, kuten hiiren painikkeen painallukset, joilla käyttäjä voi ohjata prosessia. Tässä luvussa tapahtumilla käsitetään valvomosovelluksen automaattisesti tekemiä toimintoja esimerkiksi jonkin muuttujan tilan tai ajan perusteella.

Tapahtumat voivat olla aikaperusteisia, esimerkiksi vuorokauden vaihtuessa toteutettavia, tai ne voivat perustua laukaisuun (eng. trigger), jolloin jokin prosessissa esiintyvä toiminto laukaisee tapahtuman.

Esimerkki aikaperusteisesta tapahtumasta on vuororaportin muodostaminen kahdeksan tunnin välein. Kun vuoro päättyy, talletetaan vuoron aikana tehtyjen tuotteiden määrä järjestelmään ja niistä tulostetaan johdolle menevä raportti. Esimerkki laukaisupohjaisesta tapahtumasta on moottorin tilaa indikoivan tekstin muuttaminen prosessisivulla. Kun moottorin tila muuttuu, moottori esimerkiksi pysähtyy, laukaistaan tapahtuma, jossa kutsutaan funktiota, jonka tehtävänä on muuttaa moottorin tilaa indikoivan tekstin sisältöä.

2.3 Ohjelmointi VijeoCitectilla

Ohjelmointityökaluna VijeoCitectissa toimii Cicode Editor. VijeoCitectissa voidaan ohjelmia kirjoittaa VijeoCitectin omalla strukturoidulla ohjelmointikielellä Cicodella tai VBA-(Visual Basic for Applications) ja VBScript-yhteensopivalla CitectVBA:lla.

Cicoden ja CitectVBA:n välillä on eroa käskyjen suorittamisessa. Cicode suoritetaan käskykohtaisesti ja CitectVBA suoritetaan rivikohtaisesti [1]. Tästä syystä CitectVBA:ssa lausekkeet tulee sijoittaa omille riveilleen. Jos samalla rivillä on useita peräkkäisiä lausekkeita, se saattaa aiheuttaa ongelmia datan oikeellisuudessa.

2.3.1 Cicode

Cicodella tehty ohjelmakoodi talletetaan tekstitiedostoihin, joiden tarkenne on .ci. Cicodella tehty ohjelmakoodi on proseduraalista. Olio-ohjelmointiin Cicode ei sovellu. Cicodella ei voida myöskään tehdä omia tietotyyppejä.

Cicodessa käytettävät muuttujat tulee aina määritellä ennen käyttöä. Tämä ei kuitenkaan koske muuttujia, jotka ovat määritelty pysyvästi muuttujatietokantaan (esim. ulkopuolisen laitteen muuttujatagit). Näitä voidaan käyttää sellaisenaan Cicodessa. Muuttujan määrittely koostuu muuttujan määrittelyalueesta, tyypistä ja nimestä.

Muuttujan määrittelyalueella määritellään tiedostot ja funktiot, joissa muuttujaa voidaan käyttää. Muuttuja voidaan määritellä käytettäväksi kaikissa Cicode-tiedostoissa ja funktioissa, jolloin sen määrittelyalue on GLOBAL. MODULE-määrittelyalue rajoittaa muuttujan käytön tiedostoon, jossa muuttuja määritellään. Tällöin muuttuja tulee määritellä kaikkien funktioiden ulkopuolella.

Muuttujia, joiden rajamäärittely on LOCAL, voidaan käyttää vain funktiossa, jossa muuttuja on määritelty. Alueen määrittely ei kuitenkaan ole pakollista. Jos aluetta ei ole määritelty, ovat funktioiden ulkopuolella määriteltujen muuttujien määrittelyalue

MODULE ja funktioiden sisällä määriteltyjen muuttujien määrittelyalue LOCAL. Ohjelmassa suositellaan käytettäväksi muuttujia, joiden määrittelyalue on LOCAL aina, kun se on mahdollista. GLOBAL- ja MODULE-muuttujia tulisi mahdollisimman paljon välttää, koska ne saattavat aiheuttaa sekaannusta, varsinkin ohjelman virheitä paikannettaessa [2].

Cicodessa muuttuja voidaan määritellä neljän eri tyypin muuttujaksi. Vaihtoehtoina ovat 32-bitin integer-tyyppinen muuttuja, 32-bitin floating point -tyyppinen muuttuja, 128 tavun mittainen string-tyyppinen muuttuja tai ActiveX-tyyppinen objektimuuttuja. Taulukosta 1 nähdään Cicodessa käytettävien muuttujien tyypit.

Taulukko 1. Cicoden muuttujien tyypit.

Tyyppi	Määrittely	Koko	Arvo
Integer	INT	32 bittiä	-2 147 483 648– 2 147 483 647
Floating point	REAL	32 bittiä	-3.4E38–3.4E38
String	STRING	128 tavua	

Cicoden taulukkomuuttujat koostuvat kokoelmasta saman tyypin muuttujia [3]. Taulukkomuuttujat määritellään Cicodessa samoilla periaatteella kuin yksittäiset muuttujatkin. Ensin siis määritellään muuttujan tyyppi ja seuraavaksi muuttujan nimi. Erona yksittäinmuuttujiin Cicodessa on, että taulukkomuuttujien koko tulee myös määritellä muuttujan tyypin ja nimen yhteydessä. Tällöin esimerkiksi viiden kokonaislukumuuttujan taulukko määritellään seuraavasti:

```
INT iLuku[5]
```

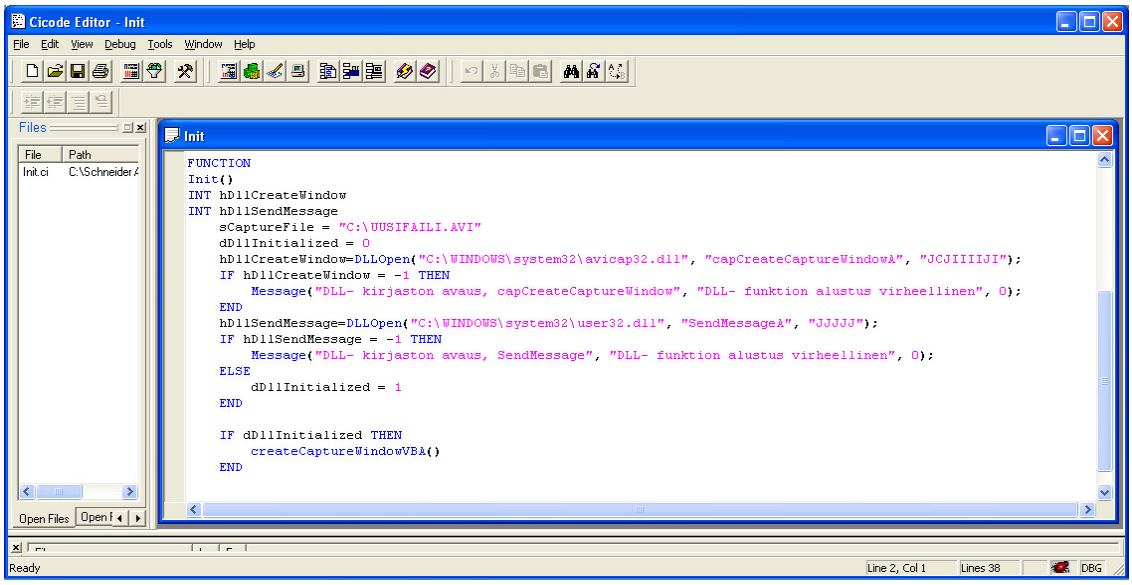
Cicode-funktiot koostuvat funktion määrittelyalueesta, funktion palautustyypistä, funktion nimestä, funktion parametreista ja funktion lausekkeista. Määrittelyalueella määritellään tiedostot, joissa funktiota voidaan kutsua. Palautustyypillä määritellään funktion palauttaman arvon tyyppi. Funktion palautustyyppi ei ole pakollinen, jos funktion ei haluta palauttavan mitään. Funktion nimen perusteella funktiota kutsutaan muualla sovelluksessa. Funktion parametrit ovat arvoja, joita tarvitaan itse funktion suorituksessa..

CitectVBA:lla tehtyjä proseduureja voidaan kutsua Cicodessa kolmea valmista Cicode-funktiota käyttämällä [4]. Nämä funktiot ovat VbCallOpen(), VbCallRun() ja VbCallReturn().VbCallOpen()-funktiolla saadaan CitectVBA:lla kirjoitetun proseduurin toteutukseen tarvittava kahva. Tämä kahva kirjoitetaan parametrina VbCallRun()-funktioon, joka suorittaa itse CitectVBA-proseduurin ja palauttaa suoritetun proseduurin kahvan, jota tarvitaan silloin, kun suoritettava CitectVBA-proseduuri palauttaa jonkin arvon. CitectVBA-proseduurin palauttama arvo saadaan VbCallReturn()-funktiolla, johon kirjoitetaan parametrina siis VbCallRun()-funktion palauttama kahva.

Jos CitectVBA-proseduurilla on omia parametreja, joita proseduurin suorituksessa tarvitaan, ne kirjoitetaan VbCallOpen()-funktion yhteydessä. VbCallOpen()-funktion parametrit ovat funktion nimi (string) sekä parametrilista, joka on pilkuilla eroitettu lista CitectVBA-proseduuriin kirjoitettavista parametreista. Tämän listan kirjoittamisessa on ohjelmoijan oltava erityisen tarkkana, koska VijeoCitectin kääntäjä ei ota kantaa parametrilistan oikeellisuuteen.

VijeoCitect havaitsee järjestelmän aiheuttamat virheet automaattisesti [5]. VijeoCitect generoi hälytyksen, kun järjestelmässä tapahtuu virhe. Hälytyksessä ilmenee hälytyksen kuvaus ja hälytyksen virhekoodi.

Kuva 5 esittää ohjelmakoodia Cicodella tehtynä.



Kuva 5. Cicode.

2.3.2 CitectVBA

CitectVBA on Visual Basic for Applications (VBA)- ja VBScript-yhteensopiva ohjelmointikieli.

CitectVBA:ssa muuttujat tulee määritellä ennen niiden käyttöä. Muuttujamäärittely aloitetaan sanalla Dim tai Static. Tämän jälkeen muuttujamäärittelyssä tulee muuttujan nimi ja sen jälkeen muuttujan tyyppi As-sanan jälkeen. Muuttujan tyypillä määritellään muuttujalle varattu muisti, muuttujaan kirjoitettavan tiedon tyyppi sekä muuttujan sisältämän arvon rajat. CitectVBA:ssa ei tarvitse välttämättä määritellä tyyppiä erikseen. Tällöin muuttujan tyypiksi määräytyy automaattisesti Variant, joka voi sisältää mitä tahansa tietoa. Taulukosta 2 nähdään CitectVBA:ssa käytettävien muuttujien tietotyypit.

Taulukko 2. CitectVBA:n tietotyypit.

Tyyppi	**Määrittely**	**Muistivaraus**	**Arvo**
Byte	Dim bByte As Byte	1 tavu	0–255
Boolean	Dim bBoolean	2 tavua	True tai False
String	Dim sString As String	10 tavua + 1 tavu/ kirjoitusmerkki	0–65535 merkkiä
Integer	Dim iInteger As Integer	2 tavua	-32768–32768
Long Integer	Dim lLong As Long	4 tavua	-2,147,483,648–2,147,483,647
Single Precision	Dim sSingle As Single	4 tavua	3.4E-38–3.4E+38
Double Precision	Dim dDouble As Double	8 tavua	1.79E-308–1.79E+308
Variant	Dim vVariant As Variant	16 tavua	Riippuu talletettavan tiedon tyypistä
Object	Dim oObject As Object	4 tavua	
Date/ Time	Dim dDate As Date	8 tavua	1.1.100–31.12.9999

CitectVBA:ssa muuttujien nimien oikeellisuus voidaan tarkistaa ohjelman käännös- tai suoritusvaiheessa. Jotta vältytään ohjelman suorituksen aikaisilta ongelmilta, on suositeltavaa määrittää muuttujien tarkistus käännösvaiheessa tehtäväksi. Tämä tapahtuu Option Explicit -määrittelyllä. Tämän määrittelyn jälkeen VijeoCitectin kääntäjä antaa virheilmoituksen kaikista määrittelemättömistä muuttujista. Option Explicit -määrittely tulee tehdä jokaisen CitectVBA-koodia sisältävän tiedoston alussa.

Taulukot ovat CitectVBA:ssa kokoelma muuttujia, jotka ovat tietotyypiltään samoja. Taulukot määritellään CItectVBA:ssa, kuten muutkin muuttujat nimellä ja tyypillä. Sen lisäksi taulukoihin annetaan lisämääreenä taulukon rajat ensimmäisen alkion indeksistä viimeisen alkion indeksiin. Taulukon yksittäisiin alkioihin viitataan siis indeksillä. Indeksit alkavat joko numerosta 0 tai 1. Tämä voidaan määritellä CitectVBA-tiedoston alussa Option Base -määritelmällä. Option Base 0 tarkoittaa, että alkiot alkavat numerosta 0 ja Option Base 1 vastaavasti tarkoittaa, että alkiot alkavat numerosta 1. Jos

Option Base määrittelyä ei käytetä, alkavat taulukot alkiosta 0. Taulukon rajat voidaan määritellä myös niin, että nimen jälkeen määritellään vain taulukon viimeisen alkion indeksi, jolloin taulukon alkuindeksi määräytyy Option Base -määritelmän mukaan.

Jos taulukon koko halutaan dynaamiseksi, eli sen kokoa halutaan muuttaa ohjelmassa tarpeen mukaan, ei nimen jälkeen tulevien sulkeiden väliin kirjoiteta mitään. Tällöin taulukon viimeisen alkion indeksi ja näin ollen myös taulukon koko jäävät avoimiksi. Kun taulukkoa sitten ohjelmassa käytetään, luodaan taulukon koko käyttöä varten ReDim-sanalla. Tässä tapauksessa edellä mainittu kokonaislukutaulukko määriteltäisiin ensin seuraavasti:

```
Dim iTaulukko() As Integer
```

Kun tätä taulukkoa sitten halutaan käyttää, määritellään taulukko seuraavasti:

```
ReDim iTaulukko(10) As Integer
```

Aina kun taulukko määritellään uudelleen ReDim-sanalla, tyhjennetään taulukon alkiot niiden sisältämistä arvoista automaattisesti. Jos taulukon alkioiden halutaan säilyttävän uudelleenmäärityksessä vanhat arvonsa, se onnistuu Preserve-sanan avulla seuraavasti:

```
Preserve ReDim iTaulukko(10) As Integer
```

Jos taulukon koko pienenee uudelleenmäärityksessä, menetetään taulukon ulkopuolelle jäävien alkioiden tiedot.

CitectVBA:ssa voidaan kirjoittaa omia proseduureja, joita voidaan muualla ohjelmassa kutsua. CitectVBA käsittää kahdentyyppisiä proseduureja. Toinen tyyppi on nimeltään aliohjelma ja toinen funktio. Näiden kahden erona on, että aliohjelma ei palauta mitään arvoa. Jos proseduurin halutaan palauttavan jotain, se täytyy määritellä funktioksi.

Kaikki proseduurit ovat CitectVBA:ssa globaaleja, eli yhdessä tiedostossa kirjoitettua proseduuria voidaan kutsua muissakin projektin tiedostoissa.

Proseduurin määrittely CitectVBA:ssa alkaa aina proseduurin tyypin määrittelyllä. Aliohjelma aloitetaan sanalla Sub ja lopetetaan sanoihin End Sub. Kaikki näiden sanojen väliin jäävät lauseet kuuluvat tähän proseduuriin, ja ne suoritetaan proseduuria kutsuttaessa. Funktion määrittely noudattaa samoja sääntöjä kuin aliohjelman määrittely, sillä erotuksella että funktion määrittelyssä Sub-sanan tilalle sijoitetaan

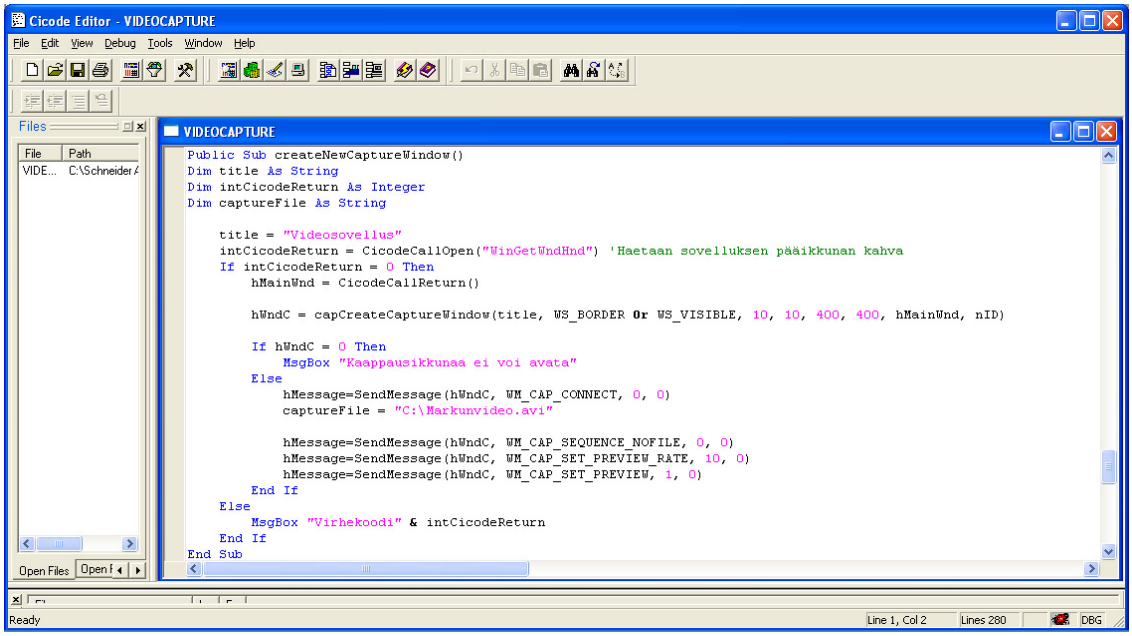
Function. Kun proseduurin tyyppi on määritetty, annetaan proseduurille nimi. Sen jälkeen proseduurille määritellään suluissa parametrit, jos proseduurille halutaan välittää kutsussa joitain parametreja. Parametrin määrityksessä vaaditaan vähintään parametrin nimi. Usein myös määritetään parametrin tyyppi kuten muuttujamäärittelyssäkin. Jos tyyppiä ei määritetä, se on oletusarvoisesti tyyppiä Variant.

Parametrien määrityksessä voidaan nimen ja tyypin lisäksi käyttää ByRef- ja ByVal-määritteitä. ByRef-määrittelyllä proseduurille välitetään parametrin osoite, eli pointteri. Tällöin parametrin arvoa on mahdollisuus muuttaa proseduurissa [6]. ByVal-määrittelyllä proseduurille välitetään parametrin arvo eli kopio parametrista. Tällöin parametrin arvoa ei voida muuttaa proseduurissa. Näiden kahden määrittelyn eron ymmärtäminen on ohjelman tekijälle erittäin tärkeää, varsinkin kun kutsutaan käyttöjärjestelmän API-kirjastoa. Väärällä tavalla välitetty parametri saattaa aiheuttaa suuria ongelmia [7, s.15]. ByRef-määrittely on oletusarvona CitectVBA-proseduureissa, eli sitä ei tarvitse erikseen määritellä.

Cicode-funktioiden kutsu CitectVBA:ssa toteutetaan kahdella CitectVBA-funktiolla: CicodeCallOpen() ja CicodeCallReturn() [8]. Itse funktiokutsu toteutetaan CicodeCallOpen()-funktiolla, joka palauttaa kokonaisluvun funktiokutsun onnistumisesta riippuen seuraavasti:
0 = CicodeCallOpen onnistui.
1 = CicodeCallOpen-virhe.
2 = kutsussa määriteltyä Cicode-funktiota ei löydy.
3 = Cicode-funktion argumenttien määrä ei ole oikea.
CicodeCallOpen()-funktion argumenttien määrä vaihtelee kutsuttavan Cicode-funktio argumenttien määrän mukaan. CicodeCallOpen()-funktion ensimmäisenä argumenttina on kutsuttavan Cicode-funktion nimi. Muut argumentit ovat Cicode-funktion argumentteja pikuilla eroitettuna.

Cicode-funktion palauttama arvo saadaan CicodeCallReturn()-funktiolla, joka palauttaa viimeeksi CicodeCallOpen()-funktiolla kutsutun Cicode-funktion palautusarvon. Palautusarvon tyyppi vaihtelee Cicode-funktion palautusarvosta riippuen.

Kuva 6 esittää ohjelmakoodia CitectVBA:lla tehtynä.



Kuva 6. CitectVBA.

3 Kuvan ja äänen käsittely tietokoneessa

Tietokoneeseen talletettua kuvaa ja ääntä käsitellään digitaalisesti. Kuva saadaan tietokoneeseen kuvankaappauksella ja ääni äänen kaappauksella. Digitaalinen kuva ja ääni voidaan sellaisenaan siirtää tietokoneeseen. Analoginen kuva- ja äänisignaali muunnetaan tietokoneessa digitaaliseen muotoon. Signaali voi tulla tietokoneeseen esimerkiksi, mikrofonista, kamerasta, skannerista, videolaitteesta, televisiosta tai DVD-laitteesta.

3.1 AD-muunnos

Analogisen signaalin AD-muunnokseen (analog-to-digital) tarvitaan kaksi vaihetta: näytteenotto ja kvantisointi [9, s. 14]. Näytteenotolla tarkoitetaan toimenpidettä, jossa analogisesta signaalista otetaan näytteitä digitaaliseen muotoon muutettavaksi. Näytteenottoon liittyy käsite nimeltä näytteenottotaajuus (eng. sampling rate), eli kuinka tiheästi signaalista otetaan näytteitä. Mitä tiheämmin näytteitä otetaan, sen paremmin näytteistä muodostettu digitaalinen tulos vastaa alkuperäistä.

Analoginen signaali käsittää äärettömän määrän eri arvoja, eli analogisessa maailmassa kahden eri arvon välillä on ääretön määrä muita arvoja. Tietokone käsittelee tietoa kuitenkin digitaalisesti. Digitaalisessa maailmassa arvoja on rajallinen määrä ja arvot eroavat selkeästi toisistaan. Arvojen välillä ei ole mitään. Kvantisoinnilla tarkoitetaan näytteenotolla saatujen signaalinäytteiden arvojen muuntamista tietokoneelle sopivaan arvoon. Mitä enemmän kvantisointisarvoja on käytettävissä, sitä lähemmäksi alkuperäistä signaalia päästään.

Näytteenottotaajuudella ja kvantisoinnilla voidaan siis vaikuttaa lopputuloksen laatuun. Korkeamman laadun hintana on kuitenkin lopputuloksen suurempi tilantarve ja signaalin käsittelyyn tarvittavan ajan kasvu.

3.2 Äänen käsittely

Ääni on värähtelevän lähteen muodostamaa aaltoliikettä. Ääniaalto tarvitsee edetäkseen väliaineen, kuten ilman. Ääniaallon lähde saa ilmanpaineen muutoksia aikaan liikuttamalla ilmamolekyylejä edestakaisin. Nämä ilmanpaineen muutokset saavat korvan tärykalvon värähtelemään ja tämän värähtelyn korva muuttaa aivoille lähteväksi äänisignaaliksi. [9, s. 102.]

Äänisignaalissa taajuus määrittelee äänen korkeuden. Mitä korkeampi taajuus, sitä korkeampi ääni. [9, s. 104.] Signaalin voimakkuus määrää äänen voimakkuuden ja äänen tehon. Nämä kaksi käsitettä ovat lähellä toisiaan, mutta ne eivät ole kuitenkaan sama asia. Äänen voimakkuus on yksilöllinen havainto. Äänen teho on kahden eri äänen voimakkuuden suhde. Äänen teho ilmoitetaan desibeleinä, dB. [9, s. 105−106.]

Ääniaalto on analoginen ilmiö [9, s. 102], joten äänisignaalin digitointiprosessiin kuuluu näytteenotto ja kvantisointi. Äänisignaalin näytteenotossa pätee Nyquistin teoreema, eli näytteenottotaajuuden on oltava vähintään kaksinkertainen äänisignaalin taajuuteen verrattuna. Ihmiskorva havaitsee yleensä äänet, joiden taajuus on 20−20 000 Hz. Tämä on yksilöllistä, ja äänialue kapenee yleensä iän myötä. [9, s. 104.] CD-tason äänen näytteenottotaajuus on 44 100 Hz [9, s. 148].

Äänisignaalin kvantisoinnissa äänen laatuun vaikuttaa kvantisointitasojen määrä, eli näytteen koko sekä signaalin muutosalue (eng. dynamic range). CD-tasoisessa äänessä käytetään 16-bittistä näytteen kokoa [9, s. 148]. Tämä mahdollistaa 65536 erilaista äänen voimakkuuden arvoa.

Signaalin muutosalueella määritellään korkein ja alin mahdollinen arvo, jonka näyte voi saada digitointiprosessissa. Jos muutosalue on liian pieni, jäävät signaalin huiput alueen ulkopuolelle. Jos muutosalue on liian suuri, kärsii tuloksen tarkkuus kvantisointivirheen kasvaessa.

3.3 Pysäytyskuvan käsittely

Näkymä, jonka ihminen silmillään näkee, koostuu äärettömästä määrästä värejä . Eri värien välillä ei ole selkeää jakoa. Jokaisen pisteen väri sekoittuu pehmeästi viereisen pisteen väriin. Väri on siis analoginen ilmiö ja kuvissa esiintyvät värit täytyy digitoida, jotta kuvaa voidaan tietokoneella käsitellä. [9, s. 27.]

Kuvasignaalista otettua näytettä nimitetään pikseliksi. Näytteistetty kuva koostuu siis useasta pikselistä. Kuva on sitä lähempänä alkuperäistä, mitä enemmän pikseleitä kuvaan sisältyy. Jokainen pikseli sisältää kuvan tietyn kohdan väri-informaation. Pikselin kohta kuvassa määritellään vaaka- ja pystykoordinaateilla. Pikseliulottuvuus ilmaisee kuvan leveyden ja korkeuden pikseleissä. Esimerkiksi digitaalikameran ottamassa kuvassa voi olla 3000 x 2000 pikseliä. Tämä ilmaistaan myös mega-pikseleinä, eli tässä tapauksessa kyseessä olisi kuuden megapikselin kamera. [9, s. 27.] Kuvasignaalin näytteistyksessä näytteenottotaajuudella tarkoitetaan pikseleiden tiheyttä, eli kuvan resoluutiota [9, s. 29]. Suuremmalla resoluutiolla päästään lähemmäksi alkuperäistä tulosta.

Kuvasignaalin kvantisoinnissa näytteelle määritellään väri, joka digitoidussa kuvassa näytetään. Näytteen kvantiosointiin vaikuttaa värisyvyys, eli kuinka monta väriä on käytettävissä. Esimerkiksi RGB-värimallissa jokaiselle komponentille on varattu kahdeksan bittiä, joten kokonaisuudessaan käytössä on 24 bittiä, mikä mahdollistaa yli 16 miljoonaa väriä [9, s. 32].

3.4 Videokuvan käsittely

Jatkuva videosignaali koostuu peräkkäin esitetyistä kuvista (eng. frame). Videosignaalin näytteistys ja kvantisointi tapahtuu kuten yksittäisenkin kuvan kohdalla. Videosignaalin käsittelyyn liittyy näytteenoton ja kvantisoinnin lisäksi kuvanopeus. Kuvanopeuden yksikkö on fps, frames per second. Tämä ilmaisee, kuinka monta kuvaa sekunnissa näytteistetään. Mitä suurempi kuvanopeus on, sen pehmeämpänä liikkeet näkyvät videokuvassa. Suurempi kuvanopeus tarkoittaa myös suurempaa tiedostokokoa.

3.5 Videokuvan kaappauslaitteisto

Kuvankaappauslaitteistona voidaan käyttää digitaalista kameraa, jolloin videokuvan digitointia ei tarvitse tehdä tietokoneessa, vaan kameralla kuvattu video voidaan suoraan siirtää tietokoneeseen. Digitaalinen videokamera voidaan liittää tietokoneeseen FireWire-liitännällä tai USB-väylällä. FireWire on Applen tavaramerkki. Sonyn laitteissa samasta liitännästä käytetään nimeä i.Link. Kuvassa 7 on 4-pinninen FireWire-liitin ja kuvassa 8 on 6-pinninen FireWire-liitin.

Kuva 7. 4-pinninen FireWire [10].

Kuva 8. 6-pinninen FireWire [10].

Kuva 9. A-tyypin USB- liitin [11].

Kuva 10. B-tyypin USB- liitin [11].

USB-liittimiä on kahta päätyyppiä, A ja B. Kuvassa 9 on esitettynä A-tyypin USB-liitin ja kuvassa 10 B-tyypin liitin.

Digitaalista kuvaa voidaan vastaanottaa tietokoneeseen myös IP-kamerasta. IP-kameran kuvaa ja ohjausta siirretään tietokoneen ja kameran välillä IP (Internet Protocol)-teknologiaa käyttäen Fast Ethernet -verkon välityksellä.

Digitaaliset kamerat käyttävät kuvankäsittelyyn DV-standardia, joka sisältää digitaalisen videon pakkauksen kuvauksen. DV-standardista on useita variaatioita. Ensimmäiset versiot digitaalisista videostandardeista esiteltiin vuonna 1987 [12, s. 342]. 1990-luvulla DV-standardia laajennettiin käsittämään myös HDTV-formaatti. Tähän päivään tultaessa standardin eri variaatioita on yli 20. Taulukko 3 esittää digitaalisia videostandardeja ja taulukko 4 niitä standardeja, jotka sisältävät HDTV-formaatin.

Taulukko 3. Digitaalisia videostandardeja [12, s. 342]

Tyyppi	Vuosi	Pakkaustapa	Mb/s	Laatu
D1	1987	pakkaamaton	172 video, 225 kaikki	Ammattilaatu
D2	1990	pakkaamaton	94 video, 127 kaikki	Ammattilaatu
D3	1993	pakkaamaton	94 video, 127 kaikki	Ammattilaatu
Digital Betacam	1993	2.7:1	90 video, 128 kaikki	Ammattilaatu
D5	1994	pakkaamaton	220 video, 330 kaikki	Ammattilaatu
DVCAM	1996	DV pakkaus, 5:1	25 video, 42 kaikki	Kuluttaja
DV ja Mini-DV	1996	DV pakkaus, 5:1	25 video, 42 kaikki	Kuluttaja
Betacam SX	1996	MPEG-2, 10:1	18.7 video, 44 kaikki	Ammattilaatu
D7	1998	DV pakkaus, 5:1	25 video, 42 kaikki	Ammattilaatu
Digital-8	1998	DV pakkaus 5:1	25 video, 42 kaikki	Kuluttaja
DC-PRO 50	1998	DV pakkaus 3.3:1	50 video, 80 kaikki	Ammattilaatu
D9	1999	DV pakkaus 3.3:1	50 video, 80 kaikki	Ammattilaatu
D10 IMX	2001	MPEG-2	50 video, 105 kaikki	Ammattilaatu

Taulukko 4. Digitaalisia HDTV videostandardeja [12, s. 343]

Tyyppi	Vuosi	Pakkaustapa	Mb/s	Tallennus
D5 HD	1994	motion-JPEG, 5:1	270	½”
D6	1995	pakkaamaton	1188	3/4”
D9	1996	MPEG-2	50	½”
D11 HDCAM	1997	M-JPEG-type	140	½”
D12 DVCPRO HD	2000	eri yhdistelmiä	100	½”
HDCAM SR	2003	MPEG-4 studio profile	440 tai 880	½”
HDV	2003	MPEG-2	17 tai 25	¼”
XDCAM	2003	MPEG-2, DV25 ja MPEG-4	18, 35 tai 50	optinen levy
AVCHD	2006	MPEG-4 H.264	12–24	DVD, kovalevy, muistitikku tai sisäinen flash
XDCAM EX	2007	MPEG-2 lomg GOP	25 tai 35	SxS muistikortti
Red 3K Scarlet, 4K One, 5K, Epic	2006–2008	pakkaamaton tai Redcode raw	26–36 MB/s	kovalevy tai flash-muisti

Digitaalisen kuvan siirtämisen kuvan kaappaus- tai tallennuslaitteelta näytölle tarkoitettuja formaatteja on kahta päätyyppiä, DVI (digital video interface) ja HDMI (high definition multimedia interface) [12, s. 338]. DVI (kuva 11) on tullut VGA-liitännän tilalle, ja ennen HDMI-liitäntää sitä käytettiin yleisesti tietokoneen ja LCD- tai plasmanäytön välisen signaalin siirtämiseen. DVI-formaatti voidaan jakaa kolmeen alaformaattiin, DVI-D:hen, DVI-A:han ja DVI-I:hin. DVI-D:llä siirretään signaalia digitaalisesta laitteesta toiseen, DVI-A:ta käytetään, kun signaalissa täytyy tehdä AD-muunnos ja DVI-I:tä voidaan käyttää sekä digitaalisten että analogisten laitteiden välillä.

Kuva 11. DVI- liitin [13].

HDMI (kuva 12) on digitaalisen kuvan ja äänen siirtoa varten kehitetty formaatti. HDMI on takaisinpäin yhteensopiva DVI-formaatin kanssa. HDMI:tä käytetään signaalin siirtoon digitaalisten laitteiden, kuten DVD:n digitaaliseten kameroiden ja pelikonsoleiden ja digitaalisten äänisoittimien ja monitorien välisen signaalin siirtoon.

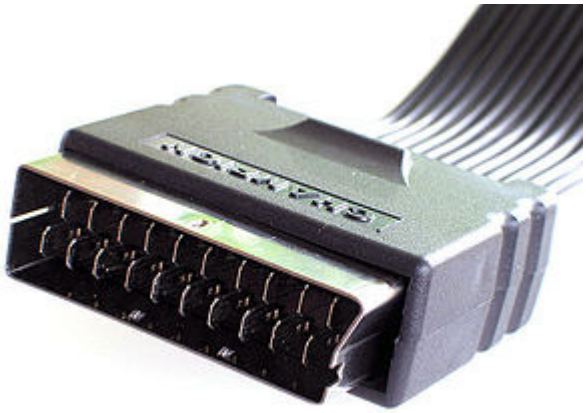
Kuva 12. HDMI- liitin [14].

Kuvankaappaus voidaan tehdä myös analogisen kameran kuvasignaalista. Tällöin tietokoneeseen tarvitaan videokortti, jolla videokuvan digitointi suoritetaan. Analogisen signaalin liittämiseen on kolme tapaa: komponentti, komposiitti ja S-video [12, s. 337]. Komponenttisignaali koostuu kolmesta komponentista Y, U ja V. Y-komponentti tarkoittaa kuvan valoisuutta, eli luminanssia. U ja V sisältävät kuvan väritiedon eli krominanssin. Komponenttisignaalissa jokainen komponentti siirretään omassa johtimessa. Komponenttivideota käytetään enimmäkseen ammattilaitteissa.
S-video on laadultaan komponenttivideota heikompi. S-videossa on käytössä kaksi signaalia, yksi luminanssia varten ja yksi krominanssikomponentteja varten. S-videossa käytetään 4-pinnisiä min-Din-liittimiä (kuva 13).

Komposiittivideo on näistä signaaleista laadultaan heikoin. Komposiittivideossa käytetään vain yhtä kaapelia komponenttien siirtoon, joten krominanssi ja luminanssi sekoittuvat samassa signaalissa aiheuttaen laadun heikkenemistä. Komposiittivideossa signaali siirretään RCA-liittimellä (kuva 14) varustetulla kaapelilla tai SCART-kaapelilla (kuvat 15 ja 16).

Kuva 13. S-videoliitin [15].

Kuva 14. Komposiittivideoliitin [16].



Kuva 15. SCART-liitin, uros [17].

Kuva 16. SCART-liitin, naaras [17].

Komposiittivideossa keltainen liitin on videosignaalia varten. Valkoinen ja punainen liitin ovat stereoääntä varten.

Taulukossa 5 on esitetty analogisia videoformaatteja ja niiden signaalien siirtotapoja.

Taulukko 5. Analogiset videosignaalit [12, s. 337].

Videoformaatti	Signaalin siirtotapa	Laatu
VHS	Komposiitti	Kuluttajalaatu
Betamax	Komposiitti	Kuluttajalaatu
8 mm (Video 8)	Komposiitti	Kuluttajalaatu
S-VHS	S-video	Korkea kuluttajalaatu
Hi-8	S-video	Korkea kuluttajalaatu
U-Matic	Komposiitti	Ammattilaislaatu
M-II	Komponentti	Ammattilaislaatu
Betacam	Komponentti	Kuluttajalaatu
Betacam SP	Komponentti	Ammattilaislaatu

4 Kuvan ja äänen käsittely Windows-käyttöjärjestelmässä

Kuvankaappaus Windows-käyttöjärjestelmässä voidaan toteuttaa käyttämällä sovelluksessa käyttöjärjestelmän multimediaominaisuuksia ja käyttöjärjestelmän näihin ominaisuuksiin tarjoamaa ohjelmointirajapintaa (API, Application Programming Interface). Yleisesti ottaen tämä rajapinta tarjoaa palveluita, joita käytetään Windows-pohjaisissa sovelluksissa [18]. Tämän rajapinnan funktioita voidaan yleisesti käyttää 32- ja 64-bittisissä versioissa.

Windowsin ohjelmointirajapinta voidaan jakaa toiminnallisesti seitsemään kategoriaan: hallintaan, diagnostiikkaan, grafiikkaan ja multimediaan, verkkoon, turvallisuuteen, järjestelmäpalveluihin ja Windows-käyttöliittymään. Kuvankaappaukseen tarvittavat funktiot löytyvät pääasiassa grafiikka- ja multimedia-kategoriasta ja erityisesti sen Video Capture -alakategoriasta.

4.1 Video Capture

Windowsin kuvankaappausominaisuuksia hyödynnetään käyttämällä AVICap-ikkunaluokkaa [19]. Tämä luokka tarjoaa viestipohjaisen rajapinnan sovelluksen ja tietokoneen äänen ja kuvankaappauslaitteiston välille ja ohjausrajapinnan, jolla voidaan hallita videokuvan tallennusta tietokoneen kovalevylle.

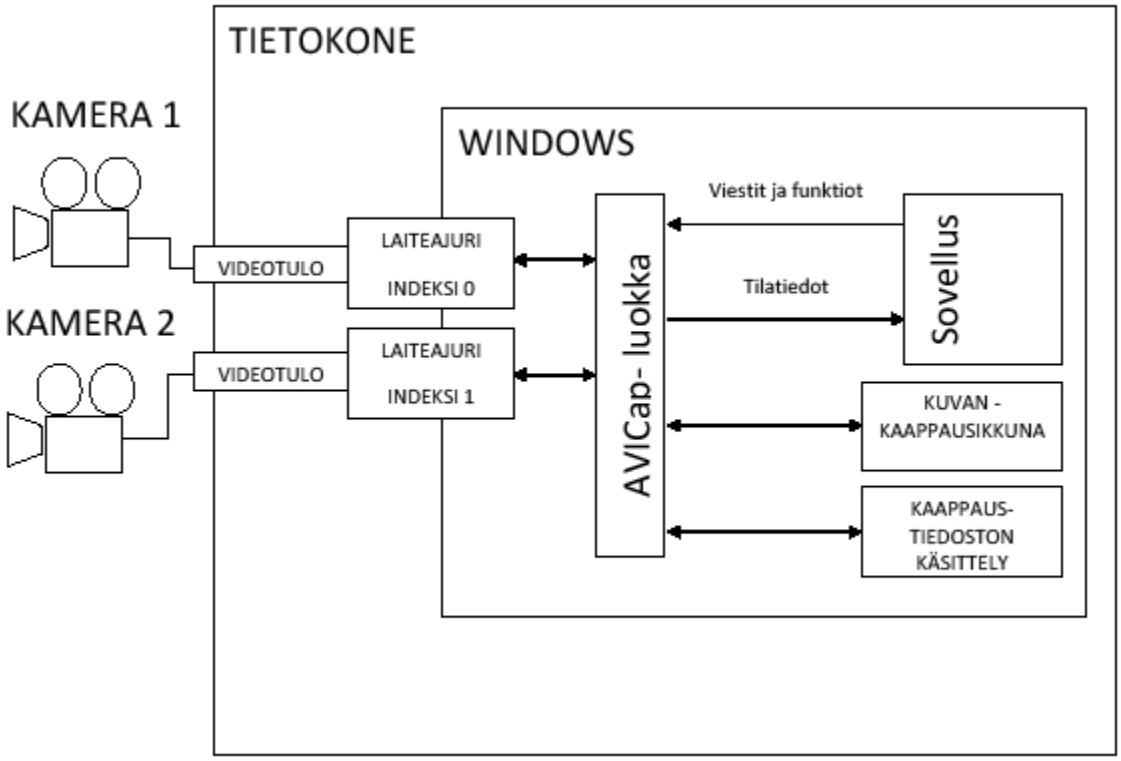
Viestipohjaisella rajapinnalla tarkoitetaan tapaa, jolla sovellus kommunikoi kuvankaappausikkunan kanssa. Sovellus lähettää ikkunalle viestejä, jotka sisältävät käskyn jonkin toiminnon suorittamiseen.

AVICap-ikkunaluokka sijoittaa ääni- ja kuvadataa AVI-tiedostoihin. Tällöin sovelluksen ei tarvitse huolehtia AVI-tiedoston käsittelystä, video- ja äänipuskureiden hallinnasta, tai laiteajureista [20].

AVICap-ikkunaluokka tarjoaa funktiot seuraavien toimintojen toteutukseen [21]:

- kuvan- ja äänenkaappauksen tallennus AVI-tiedostoon (audio- video interleaved)
- kuvan- ja äänenkaappauslaitteiston kytkentä kuvankaappausikkunaan
- reaaliaikaisen kuvan esitys kuvankaappausikkunassa overlay- ja preview-metodien avulla
- AVI-tiedostojen käsittely kuvankaappauksen aikana ja tiedoston kopiointi toiseen tiedostoon
- kuvankaappausnopeuden määritys
- kuvan lähteen ja formaatin hallintaikkunoiden esittäminen
- väripalettien hallinta
- yksittäisten kuvien hallinta.

Kuva 17 havainnollistaa kuvankaappausta Windowsin API-rajapinnan avulla.



Kuva 17. Kuvankaappauksen periaatekaavio.

Kuvasta nähdään, että AVICap-luokka huolehtii laitteiden ohjauksesta, eikä sovelluksen tarvitse ottaa kantaa niihin.

4.2 AVICap-luokan käyttö

Lyhyesti selostettuna AVICap-luokkaa käytetään kuvankaappaukseen siten, että ensin luodaan ikkuna, jossa videokuvaa halutaan esittää. Tämän jälkeen ikkunaan liitetään kuvankaappauslaite ja sen jälkeen käynnistetään kuvankaappaus. Yksinkertaisimmillaan kuvankaappauksen toteutus onnistuu kolmella ohjelmalauseella [20]:

```
hWndC = capCreateCaptureWindow ( "My Own Capture Window",
WS_CHILD | WS_VISIBLE , 0, 0, 160, 120, hwndParent, nID);

SendMessage (hWndC, WM_CAP_DRIVER_CONNECT, 0, 0L);
```

```
SendMessage (hWndC, WM_CAP_SEQUENCE, 0, 0L);
```

Ensimmäisessä funktiossa luodaan kuvankaappausikkuna. Ikkunan kahva palautetaan hWndc-muuttujaan. Ikkunan nimeksi on esimerkissä annettu ”My Own Capture Window”. Toisella parametrilla määritellään ikkunan tyyppi. Seuraavat kaksi parametria ovat ikkunan vasemman ylänurkan koordinaatiti tietokoneen näytöllä. Näiden jälkeen tulevat ikkunan leveys ja korkeus. hwndParent-muuttuja sisältää pääsovelluksen ikkunan kahvan.

Toisessa funktiossa kuvankaappauslaitteisto kytketään kuvankaappausikkunaan lähettämällä ikkunalle viesti. Ensimmäisenä parametrina on kuvankaappausikkunan kahva. Toinen parametri on vakio, jolla määritetään käsky kytkeä laitteisto ikkunaan. Kolmas parametri on ikkunaan kytkettävän laitteiston indeksi. Jokainen tietokoneeseen liitetty laitteisto on indeksoitu. Indeksin arvo voi olla 0–9. Neljäs parametri ei ole käytössä tässä tapauksessa.

Kolmas funktio sisältää viestin, jolla kuvankaappaus aloitetaan ja videokuva talletetaan CAPTURE.AVI-tiedostoon. Parametrit ovat samat kuin edellisessä viestissä. Kuvankaappausta jatketaan, kunnes käyttäjä painaa ESC-painiketta, tallennusmedia tulee täyteen tai sovellus lopettaa kaappauksen. Kuvankaappaus pysäytetään viesteillä

```
SendMessage (hWndC, WM_CAP_STOP, 0, 0L); tai
SendMessage (hWndC, WM_CAP_ABORT, 0, 0L);
```

Kuvankaappausikkunaan liitetty ajuri saadaan kytkettyä pois lähettämällä ikkunalle viesti

```
SendMessage (hWndC, WM_CAP_DRIVER_DISCONNECT, 0, 0L);
```

AVICap-luokka tarjoaa edellisten lisäksi useita funktioita ja viestejä, joilla kuvankaappausta voidaan hallita. Osa käskyistä ohjaa ikkunaa tekemään toimintoja, osa käskyistä pyytää ikkunalta tietoa ikkunasta tai siihen liitetystä ajurista.
Tiedot, joita ikkunalta saadaan, sijoitetaan struktuureihin. Esimerkkinä on viesti

```
SendMessage (hWndC, WM_CAP_DRIVER_GET_CAPS,
sizeof (CAPDRIVERCAPS), (LONG) (LPVOID) &CapDrvCaps);
```

Tässä viestissä toisena parametrina on vakio WM_CAP_DRIVER_GET_CAPS, jolla saadaan tietoa ajurin toimintakyvystä CAPDRIVERCAPS-struktuuriin kirjoitettuna. Tämä struktuuri sisältää tietoa muun muassa ajurin tukemista hallintadialogeista ja siitä, onko ajuri kytketty ikkunaan onnistuneesti. Kuvankaappauksessa käytettävät funktiot on selostettu tarkemmin liitteessä 1 ja struktuurit liitteessä 2. Multimediaviestien vakioiden arvot on esitetty liitteessä 3.

Kuvankaappausikkunan tila saadaan lähettämällä ikkunalle viesti

```
SendMessage (hWndC, WM_CAP_GET_STATUS,
sizeof (CAPSTATUS), (LONG) (LPVOID) &CapStatus);
```

jossa toisena parametrina oleva vakio WM_CAP_DRIVER_GET_CAPS palauttaa tilan CAPSTATUS-nimiseen struktuuriin. Tämä struktuuri sisältää tietoa muun muassa ikkunan koosta, kaappaustiedostosta ja kaapatun äänen ja kuvan määrästä.

Tietoa kaappausprosessista saadaan lähettämälle ikkunalle viesti

```
SendMessage (hWndC, WM_CAP_GET_SEQUENCE_SETUP,
sizeof (CAPTUREPARMS), (LONG) (LPVOID) &CaptureParms);
```

Tämä viesti tallettaa tietoa CAPTUREPARMS-struktuuriin.

Videokuva asetetaan esikatselutilaan lähettämällä ikkunalle viestit

```
SendMessage (hWndC, WM_CAP_SET_PREVIEW, 1, 0);
SendMessage (hWndC, WM_CAP_SET_PREVIEW_RATE, 100, 0);
```

Ensimmäisessä viestissä asetetaan esikatselutila päälle ja toisessa viestissä asetetaan kehysten esitysnopeudeksi 100 ms/kehys.

Videokuva talletetaan oletusarvoisesti C:\CAPTURE.AVI tiedostoon. Tiedosto voidaan nimetä uudelleen lähettämällä ikkunalle viesti

```
SendMessage (hWndC, WM_CAP_SET_CAPTURE_FILE, 0,
”C:\NEWCAPTURE.AVI”);
```

Viestin viimeisenä parametrina on uuden tiedoston nimi.

Videon tallennus aloitetaan lähettämälle ikkunalle viesti

```
SendMessage (hWndC, WM_CAP_SEQUENCE, 0, 0);
```

Videokuva voidaan kopioida toiseen tiedostoon viestillä

```
SendMessage (hWndC, WM_CAP_FILE_SAVEAS, 0,
”C:\MYVIDEO.AVI”);
```

Viestin viimeisenä parametrina on tiedosto, johon video tallennetaan. Tämä viesti ei vaikuta sen hetkisen kaappaustiedoston nimeen tai sisältöön.

5 Kuvankaappaus VijeoCitectissa

5.1 DLL-funktioiden alustus

Käyttöjärjestelmän DLL-funktioita voidaan kutsua sekä Cicodella, että CitectVBA:lla. Ennenkuin DLL-funktioita voidaan VijeoCitectissa kutsua, on ne alustettava. Tämä onnistuu vain Cicoden DLLOpen()-funktiolla. Jos DLL-funktioita aiotaan kutsua CitectVBA:lla, on DLL-funktiot ensin esiteltävä. Nämä funktiot eivät esittelystä huolimatta toimi sovelluksessa ennen kuin ne on alustettu DLLOpen()-funktiolla.

DLLOpen()-funktio palauttaa siinä määritellyn DLL-funktion kahvan, jota tarvitaan itse DLL-funktion kutsussa. Jos haluttua funktiota ei voida jostain syystä alustaa, palauttaa DLLOpen()-funktio arvon -1. DLLOpen()-funktio sisältää kolme argumenttia: DLL-kirjaston nimi, alustettavan DLL-funktion nimi ja DLL-funktion argumentit. Kaikki DLLOpen()-funktion kolme argumenttia ovat tyyppiä string.

Kutsuttavan DLL-funktion argumentit määritellään alustuksessa merkkijonona siinä järjestyksessä kuin ne itse funktion kutsussa määritellään. Ensimmäinen merkki jonossa ilmaisee kutsuttavan funktion palautustyypin. Tässä on huomioitava se, että argumenttien tyypit on määriteltävä oikein, koska Cicoden kääntäjä ei tarkista kyseisten merkkien oikeellisuutta. DLL-funktion argumenttien tyypit ilmaistaan seuraavasti:

- A = looginen muuttuja.
- B = IEEE 64-bittinen liukulukumuuttuja.
- C = NULL-päätteinen merkkijonomuuttuja, jonka maksimipituus on 255 merkkiä.
- D = merkkijonomuuttuja, jonka ensimmäinen tavu ilmaisee merkkijonon pituuden. Maksimipituus 255 merkkiä.
- H = etumerkitön 32-bittinen kokonaislukumuuttuja.
- I = etumerkillinen 16-bittinen kokonaislukumuuttuja.

- J = etumerkillinen 32-bittinen kokonaislukumuuttuja.

Esimerkkinä alustuksesta user32.dll-kirjastossa esiintyvä SendMessageA()-funktion alustus:

```
INT hDll;
hDll=DLLOpen("C:\WINDOWS\system32\user32.dll",SendMessageA",
"JJJJJ");
```

Ensimmäisellä rivillä määritellään muuttuja, johon alustuksessa kirjoitetaan funktion kahva. Toisella rivillä on funktion alustus DLLOpen()-funktiolla. Ensimmäisenä argumenttina alustuksessa on funktiokirjaston nimi ja sijainti, toisessa funktion nimi kirjastossa ja kolmantena palautusarvon ja argumenttien esittely.

5.2 DLL-funktioiden kutsu Cicodessa

Kun DLL-funktio on alustettu, sitä voidaan kutsua sovelluksessa. Cicodessa DLL-funktiota kutsutaan DLLCall()-funktiolla. DLLCall()-funktio palauttaa DLL-funktion palauttaman arvon string-tyyppisenä. DLLCall()-funktiossa on kaksi argumenttia: DLL-funktion alustuksesta saatu DLL-funktion kahva, sekä DLL-funktion argumentit merkkijonona, pilkuilla eroitettuna. Esimerkkinä DLL-funktion kutsusta edellämainitun SendMessageA()-funktion kutsu:

```
hWndC=DLLCall(hDll, "^""+"Video For
Citect"+"^""+","+WS_VISIBLE+",100,100,100,200,"+IntToStr(hWn
d)+",0");
```

5.3 DLL-funktioiden kutsu CitecVBA:ssa

CitectVBA:ssa DLL-funktioiden esittely aloitetaan sanalla Declare, jota seuraa sana Function. Nämä sanat ilmaisevat CitectVBA:lle, että sovelluksessa tullaan kutsumaan esiteltyjä DLL-funktioita. Function-sanan jälkeen esittely jatkuu DLL-funktion nimellä. Sen jälkeen tulee sana Lib, joka esittelee DLL-kirjaston, josta esiteltävä funktio löytyy. Jos funktion nimi eroaa kirjastossa esiintyvästä nimestä, liitetään sovelluksessa käytettävä nimi kirjastossa esiintyvää nimeen alias-sanalla. Tämä sana sijoitetaan

esittelyssä DLL-kirjaston jälkeen. Alias-sanan perään liitetään nimi, jolla DLL-funktio kirjastossa esiintyy. Seuraavaksi esittelyssä tulee sulkeiden sisään funktion argumentit ja viimeisenä funktion palautusarvo. SendMessageA()-funktio esitellään seuraavalla tavalla:

```
Declare Function CapDriverGetName Lib _
"C:\WINDOWS\system32\user32.dll" Alias "SendMessageA" _
(Byval hWnd As Long, Byval wMsg As Long, _
Byval wParam As Long, lParam As Long) As Long
```

Esittelyssä funktion nimeksi sovelluksessa määritellään CapDriverGetName, joka liitetään user32.dll-kirjastossa esiintyvään SendMessageA()-funktioon. Funktion uudelleennimeämiseen voi olla syynä se, että DLL-kirjastossa esiintyvä nimi ei sovi CitectVBA:n nimeämiskäytäntöön tai kuten tässä tapauksessa, uudella nimellä halutaan kuvata funktion käyttöä sovelluksessa. SendMessageA()-funktiota käytetään lähettämään viestejä kuvankaappausikkunalle. Viestien avulla saadaan muun muassa tietoa ikkunaan kytketystä ajurista. Haluttu viesti määritellään funktion argumenttina. Tässä tapauksessa CapDriverGetName-nimisellä funktiolla halutaan saada ikkunaan kytketyn ajurin nimi.
Kun funktio on esitelty, sitä voidaan sovelluksessa kutsua esittelyssä esiintyneellä nimellä:

```
SendMessage(hWndC, WM_CAP_CONNECT, 0, 0)
```

5.4 DLL-funktioiden sulkeminen

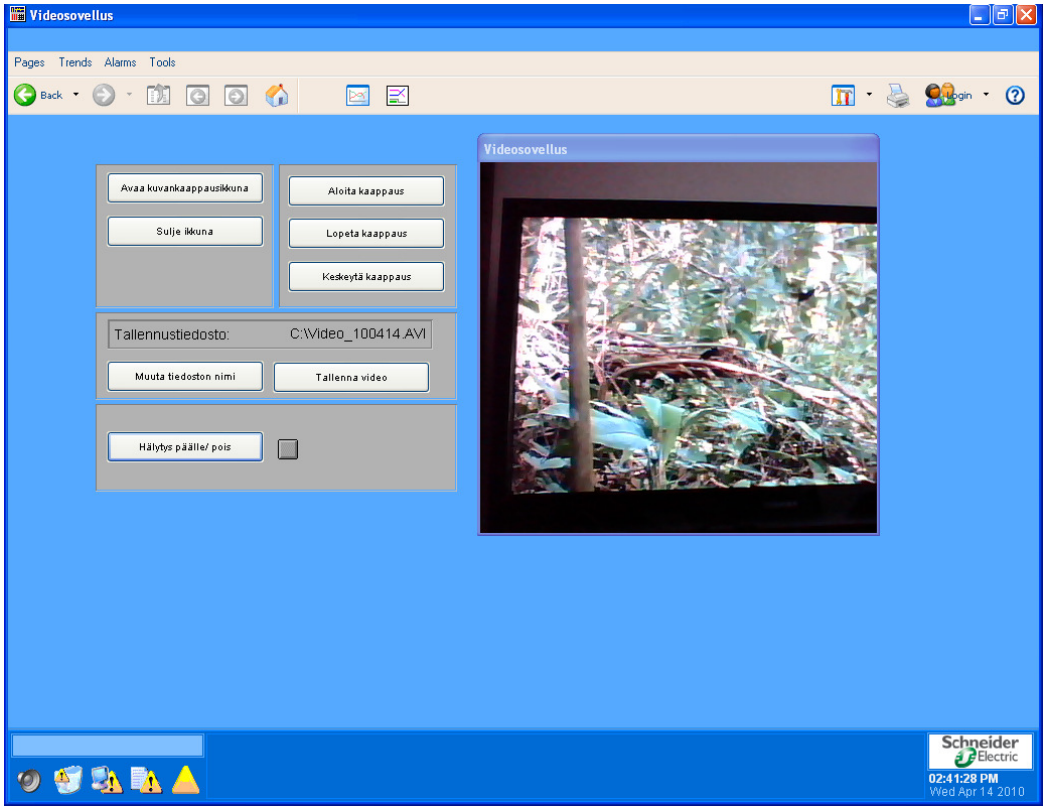
Alustetut DLL-funktiot voidaan sulkea Cicoden DLLClose()-funktiolla. Tällöin DLL-funktiolle varattu muistitila vapautuu. DLLClose()-funktiolla on parametrina suljettavan DLL-funktion kahva. DLLClose()-funktio palauttaa kokonaisluvun 0, jos DLL-funktion sulkeminen onnistui. Muussa tapauksessa DLLClose()-funktio palauttaa Cicoden virhekoodin.

Kun DLL-funktio on suljettu, sitä ei voida sovelluksessa kutsua. Kaikki alustetut DLL-funktiot suljetaan automaattisesti, kun VijeoCitect-sovellus lopetetaan.

6 Kuvankaappaussovellus VijeoCitectissa

6.1 Käyttöliittymä

VijeoCitectilla tehdyn kuvankaappaussovelluksen käyttöliittymä sisältää yhden sivun, joka sisältää painikkeet kameran ohjausta ja videon käsittelyä varten. Kuva 18 esittää sovelluksen käyttöliittymää.

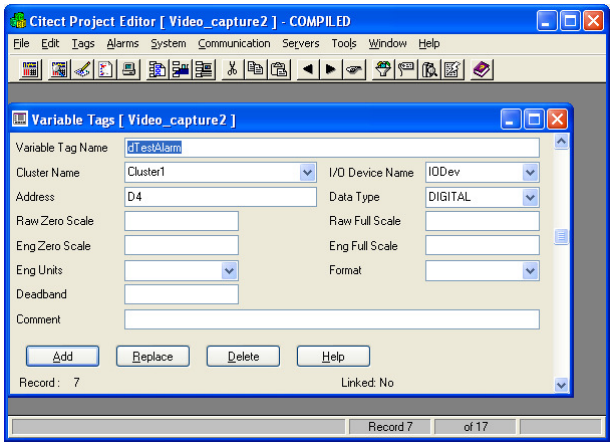


Kuva 18. Sovelluksen pääsivu.

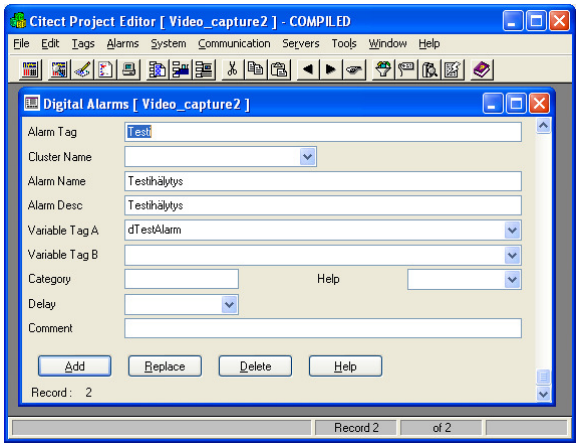
6.2 Sovelluksen toiminta

VijeoCitectilla tehdyssä sovelluksessa aloitetaan kuvankaappaus määritellyn muuttujan tilan perusteella (kuva 19). Kun muuttuja on TOSI, muodostetaan sovelluksessa hälytys (kuva 20). Muuttuja asetetaan testisovelluksessa päälle ja pois päältä käyttäjän toimesta

sovelluksen pääsivulla sijaitsevista painikkeista. Painikkeeseen on määritelty muuttujan ohjaus Toggle-komennolla. Komennolla muuttujan tila vaihtuu joka painalluksella. Hälytyksen tultua voimaan aloitetaan kuvankaappaus tietokoneeseen liitetyllä kameralla. Loppukäyttäjän sovelluksessa hälytys muodostettaisiin jonkin prosessissa tapahtuvan vikatilanteen perusteella, esimerkiksi vesisäiliön pinnankorkeuden ylitäyttöhälytyksestä.

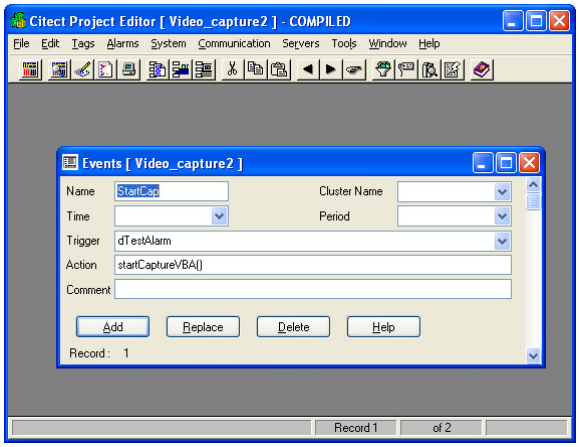


Kuva 19. Testimuuttujan määritys.

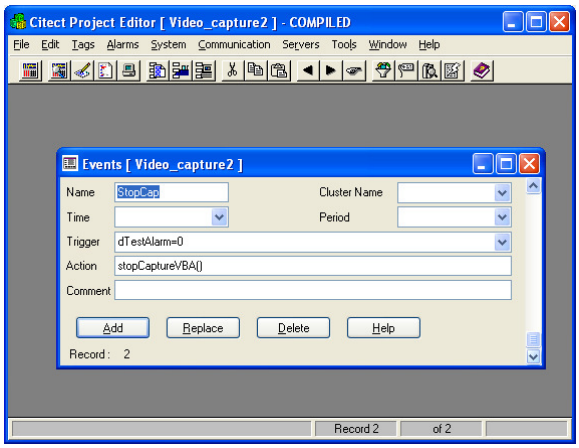


Kuva 20. Testimuuttujasta määritelty hälytys.

Kuvankaappaus lopetetaan, kun hälytys on poistunut tai käyttäjä on pysäyttänyt kuvankaappauksen. Kaapattu kuva voidaan kopioida käyttäjän määrittelemään tiedostoon. Kuvankaappauksen automaattinen käynnistys ja pysäytys tehdään kahdella event-toiminnolla (kuvat 21 ja 22). Myös käyttäjä voi kaapata kuvaa pääsivulla olevan painikkeen avulla.



Kuva 21. Kuvankaappauksen käynnistävä event-toiminto.



Kuva 22. Kuvankaappauksen pysäyttävä event-toiminto

Opinnäytetyön valvomosovellus sisältää pääsivun lisäksi toiminnot videokuvan kaappausta ja tallennusta varten. Kaappaustoiminnot sisältävät kuvankaappausikkunan avaamisen, kamera-ajurin kytkennän ikkunaan, kaappauksen aloituksen, kuvan tallennuksen ja kaappauksen lopetuksen. Sen lisäksi sovellus sisältää tallennustiedoston nimen syöttökentän ja kaappauksen aloitus- ja pysäytyspainikkeet ja videotiedoston kopiointipainikkeen.

6.3 DLL-kutsut

6.3.1 DLL-kutsujen alustukset ja esittelyt

Sovelluksessa käytettävät DLL-funktioiden alustus toteutetaan sovelluksen käynnistyksen yhteydessä kutsuttavassa, Cicodella kirjoitetussa Init()-funktiossa. Tämä funktiokutsu määritellään Citect Explorer:in Computer Setup-toiminnolla, joka löytyy Tools-valikosta.
Init()-funktiossa alustetaan avicap32.dll-kirjastossa sijaitseva capCreateCaptureWindowA-funktio ja user32.dll-kirjastossa sijaitseva SendMessageA-funktio seuraavasti:

```
FUNCTION
Init()
            INT hDllCreateWindow
            INT hDllSendMessage
            sCaptureFile = "C:\UUSIFAILI.AVI"
            dDllInitialized = 0

            hDllCreateWindow=DLLOpen("C:\WINDOWS\system32\avicap32.dll",
            "capCreateCaptureWindowA", "JCJIIIIJI");

            IF hDllCreateWindow =-1 THEN
                        Message("DLL- kirjaston avaus,
                        capCreateCaptureWindow", "DLL- funktion alustus
                        virheellinen", 0);
            END

            hDllSendMessage=DLLOpen("C:\WINDOWS\system32\user32.dll",
            "SendMessageA", "JJJJJ");

            IF hDllSendMessage =-1 THEN
                        Message("DLL-kirjaston avaus, SendMessage", "DLL-
                        funktion alustus virheellinen", 0);
            ELSE
```

```
            dDllInitialized = 1
        END
        IF dDllInitialized THEN
            createCaptureWindowVBA()
        END
END
```

Init()-funktion alussa määritellään muuttujat, joihin kirjoitetaan DLLOpen()-funktion palautusarvo. Jokaisen alustuksen jälkeen tutkitaan, onnistuiko alustus. Jos alustus ei onnistunut, generoidaan virheestä ponnahdusikkuna Cicoden Message()-funktiolla. Tässä ponnahdusikkunassa viestinä on ilmoitus alustuksen epäonnistumisesta. Alustuksessa käsitellään dDllInitialized-muuttujaa siten, että alustuksen alussa muuttuja nollataan. Jos alustukset onnistuvat, asetetaan muuttuja päälle. Tämän muuttujan tilan perusteella estetään tai sallitaan kuvankaappaukseen liittyvät toiminnot sovelluksessa.

Kuvankaappaus toteutetaan sovelluksessa CitectVBA-kielellä, jolloin funktiot esitellään, ennen kuin niitä kutsutaan CitectVBA-koodissa. Edellä mainitut funktiot esitellään seuraavasti:

```
' DLL-funktioiden esittely
Declare Function capCreateCaptureWindow _
Lib "C:\WINDOWS\system32\avicap32.dll" _
Alias "capCreateCaptureWindowA" _
(Byval WindowName As String, Byval dwStyle As Long, _
Byval x As Integer, Byval y As Integer, _
Byval nWidth As Integer, Byval nHeight As Integer, _
Byval hwndParent As Long, Byval nID As Integer) As Long

Declare Function SendMessage Lib _
"C:\WINDOWS\system32\user32.dll" Alias "SendMessageA" _
(Byval hWnd As Long, Byval wMsg As Long, Byval wParam As _
Long, lParam As Long) As Long

Declare Function SaveAsFileMsg Lib _
"C:\WINDOWS\system32\user32.dll" Alias "SendMessageA" _
(Byval hWnd As Long, Byval wMsg As Long, Byval wParam As _
Long, Byval lParam As String) As Long
```

```
Declare Function DestroyWindow Lib _
"C:\WINDOWS\system32\user32.dll" (Byval hWnd As Long) As _
Boolean
```

Ensimmäisessä esittelyssä capCreateCaptureWindowA-funktio nimetään capCreateCaptureWindow-funktioksi. Toisessa esittelyssä SendMessageA-funktio nimetään SendMessage-funktioksi. Kolmannessa esittelyssä SendMessageA-funktio nimetään SaveAsFileMsg-funktioksi. Sovelluksessa käytettävät SendMessage- ja SaveAsFileMsg-funktiot eroavat toisistaan viimeisten parametrien osalta. SendMessage-funktiossa parametrina on parametrin osoitteeseen viittaava long-typpinen parametri. SaveAsFileMsg-funktiossa parametri on parametrin arvoon viittaava string-tyyppinen parametri. Nämä erot johtuvat sittä, että user32-kirjasto SendMessageA-funktiota käytetään viestien lähettämiseen ja viestien parametrit eroavat toisistaan sen käyttötarkoituksen perusteella.

6.3.2 Ikkunan avaus ja sulkeminen

Ikkuna luodaan sovelluksessa kutsumalla CitectVBA:lla toteutettua proseduuria nimeltä createNewCaptureWindow. Proseduurin toteutus on seuraava:

```
' Kaappausikkuna
Public Sub createNewCaptureWindow()
Dim title As String
Dim intCicodeReturn As Integer
Dim captureFile As String

          title = "Videosovellus"
          intCicodeReturn = CicodeCallOpen("WinGetWndHnd")
          If intCicodeReturn = 0 Then
                    hMainWnd = CicodeCallReturn()

                    hWndC = capCreateCaptureWindow(title, WS_BORDER _
                    Or WS_VISIBLE, 10, 10, 400, 400, hMainWnd, nID)
```

```
                    If hWndC = 0 Then
                              MsgBox "Kaappausikkunaa ei voi avata"
                    Else

                              hMessage=SendMessage(hWndC, _
                              WM_CAP_CONNECT, 0, 0)

                              captureFile = "C:\Markunvideo.avi"

                              hMessage=SendMessage(hWndC, _
                              WM_CAP_SEQUENCE_NOFILE, 0, 0)
                              hMessage=SendMessage(hWndC, _
                              WM_CAP_SET_PREVIEW_RATE, 10,0)
                              hMessage=SendMessage(hWndC, _
                              WM_CAP_SET_PREVIEW, 1, 0)
                    End If
          Else
                    MsgBox "Virhekoodi" & intCicodeReturn
          End If
End Sub
```

Ensimmäisenä proseduurissa määritellään proseduurin sisäiset muuttujat, merkkijonotyyppinen ”title” ja kokonaislukutyyppinen intCicodeReturn. Ensin mainittua muuttujaa käytetään ilmaisemaan kuvankaappausikkuna otsikko. intCicodeReturn-muuttujaa tarvitaan, kun sovelluksesta haetaan pääikkunan kahva, jota tarvitaan kuvankaappausikkunan generoimisessa. intCicodeReturn-muuttujaan kirjoitetaan CicodeCallOpen()-funktion tulos. CicodeCallOpen()-funktion tehtävänä on kutsua Cicoden WinGetWndHnd()-funktiota, joka palauttaa sovelluksen pääikkunan kahvan. Jos CicodeCallOpen()-funktio palauttaa luvun 0, haetaan WinGetWndHnd()-funktion palauttama arvo CicodeCallReturn()-funktiolla ja jatketaan kuvankaappausikkunan luontiin. Jos CicodeCallOpen()-funktio palauttaa jonkin muun kuin 0, luodaan ponnahdusikkuna, joka ilmoittaa funktion palauttaman arvon.

Kuvankaappauskkuna luodaan aiemmin esiteltyä capCreateCaptureWindow-funktiota kutsumalla. Tämä funktio palauttaa kuvankaappausikkunan kahvan, jota tarvitaan viestien lähettämiseen ikkunalle.

Ikkuna suljetaan DestroyWindow-funktiolla, jonka parametrina on kuvankaappausikkunan kahva.

Ikkuna avataan sovelluksen Init-funktiossa, jos DLL-kirjastojen alustus on onnistunut. Ikkuna voidaan avata myös käyttöliittymän Avaa kuvankaappausikkuna -painikkeella.

6.3.3 Kuvan kaappaus

Tietokoneeseen liitetyn kameran ajuri kytketään kuvankaappausikkunaan lähettämällä ikkunalle viesti, joka sisältää käskyn kytkeä ajurin ikkunaan:

```
SendMessage(hWndC, WM_CAP_CONNECT, 0, 0)
```

hWndC on aiemmassa ikkunan luonnissa saatu ikkunan kahva. WM_CAP_CONNECT on vakio, joka on määritelty aiemmin CitectVBA-koodissa.

Kuvankaappaus aloitetaan viestillä

```
SendMessage(hWndC, WM_CAP_SEQUENCE_NOFILE, 0, 0)
```

Tällä viestillä kuvaa ei vielä talleteta kaappaustiedostoon.

Kuvan talletus aloitetaan viestillä

```
SendMessage(hWndC, WM_CAP_SEQUENCE, 0, 0)
```

Tämä viesti tallettaa kuvaa tiedostoon, joka on oletusarvoisesti C:\CAPTURE.AVI. Tämä viesti lähetetään ikkunalle hälytyksen tultua voimaan tai käyttäjän toimesta käyttöliittymän Aloita kaappaus -painikkeesta. Kaappaus lopetetaan hälytyksen poistuttua tai käyttäjän toimesta Lopeta kaappaus- tai Keskeytä kaappaus -painikkeista.

6.3.4 Videokuvan tallennus

Videokuva kopioidaan sovelluksessa käyttäjän toimesta käyttöliittymän Tallenna video-painikkeella, käyttäjän määrittelemään tiedostoon viestillä

```
hMessage=SaveAsFileMsg(hWndC, WM_CAP_FILE_SAVEASA, 0,
fileName)
```

Tässä viestissä filename-muuttuja sisältää kopioidun tiedoston nimen. Käyttäjä voi muuttaa tiedoston nimen lomakkeessa, jonka saa auki käyttöliittymän Muuta tiedoston nimi -painikkeella.

7 Sovelluksen testaus

7.1 Laitteisto

Laitteistona tässä sovelluksessa käytetään kannettavaa tietokonetta, johon on liitetty web-kamera USB-väylän kautta.

Tietokone:

- Packard Bell Computer
- Intel Pentium-prosessori 1,73 GHz
- RAM 1 GB
- 17”:n TFT-näyttö.

Käyttöjärjestelmä:

- Microsoft Windows XP Professional
- Versio 2002
- Service Pack 2

Valvomosovellusalusta:

- VijeoCitect-versio 7.0r3

Web-kamera:

- Logitech Webcam 200
- VGA sensori (640 x 480 pikseliä)
- Videokuva 640 x 480 pikseliä
- Still-kuvat 1,3 megapikseliä
- Kuvankaappausnopeus 30 fps
- Sisäänrakennettu mikrofoni
- Hi-Speed USB 2.0

Kello:

- Polar F6

7.2 Testitapahtuma

Sovelluksen testitapahtuman tarkoitus oli todeta seuraavien toimintojen toimivuus:

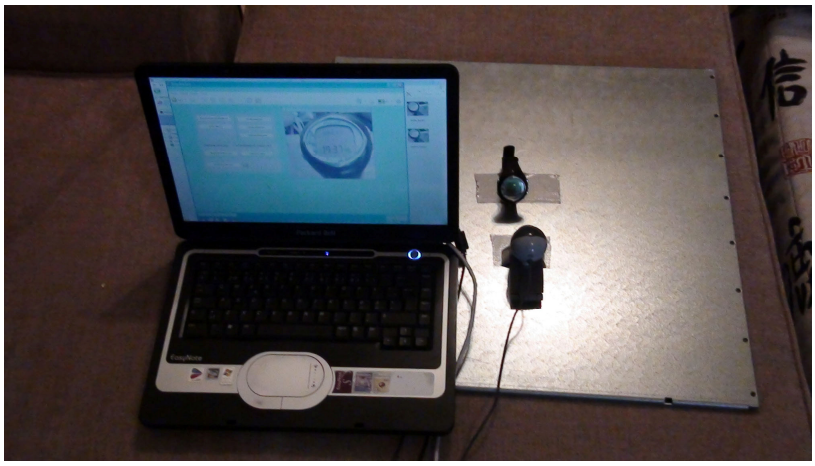
- DLL- kirjastojen alustus
- kuvankaappausikkunan luonti
- kamera-ajurin kytkeminen ikkunaan
- kuvankaappauksen käynnistys ja pysäytys
- kaapatun videotiedoston kopiointi.

Testisovellukseen ei ohjelmoitu tai määritelty muita toimintoja, koska testi haluttiin rajata vain edellä mainittuihin toimintoihin.

Sovelluksen testaus tehtiin niin, että videokamera asetettiin kuvaamaan testissä käytettyä kelloa. Jokaisen kaappauksen aloitus- ja lopetusaika kirjattiin pöytäkirjaan. Jokainen kaapattu video kopioitiin nimellä ”testi*n*_kopio1.AVI”, jossa *n* on testin järjestysnumero.

Kaapattujen ja kopioitujen videoiden tarkistus tehtiin kirjaamalla pöytäkirjaan videoissa näkyvät kellonajat.

Kuva 23 esittää sovelluksen testaukseen käytettyä laitteistoa.



Kuva 23. Sovelluksen testilaitteisto.

Sovelluksen testi toistettiin kymmenen kertaa seuraavin toimenpitein:

1. Sovelluksen käynnistys. Käynnistyksen yhteydessä tarkistetaan onnistuiko DLL-kirjastojen alustus ja kuvankaappausikkunan luonti.
2. Kuvankaappaus aloitus asettamalla hälytysmuuttuja päälle.
3. Kuvankaappauksen lopetus asettamalla hälytysmuuttuja pois päältä.
4. Kaapatun videon sisällön tarkistus.
5. Kopiotiedoston uudelleennimeäminen.
6. Kaapatun videon kopiointi.
7. Kopion tarkistaminen.

Testin päätteeksi sovellus lopetetaan.

7.3 Tulokset

Testi toistettiin kymmenen kertaa. Jokaisesta testitapahtumasta kirjattiin taulukon 6 mukaiset tiedot.

Taulukko 6. Kaappaustestin tulokset

Testin numero	1	2	3	4	5	6	7	8	9	10
Sovelluksen käynnistyminen	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK
Kaappauksen aloitus[mm:ss]	23:02	32:38	41:50	47:01	51:38	58:54	08:16	18:23	23:28	27:30
Aloitusaika videossa	23:03	32:40	41:51	47:02	51:40	58:55	08:18	18:25	23:30	27:31
Kaappauksen lopetus[mm:ss]	23:25	33:24	42:04	47:11	52:47	59:20	08:41	18:53	23:40	27:41
Lopetusaika videossa	23:28	33:23	42:25	47:22	52:47	59:21	09:13	18:54	23:48	27:41
Kaapatun videon pituus [s]	26	43	35	20	68	26	54	29	19	10
Videon pituus-testiajankohtien ero[s]	3	-3	21	10	-1	0	29	-1	7	-1
Kopiointi	OK	OK	OK	OK	OK	OK	OK	OK	OK	OK

7.4 Tulosten analysointi

Sovelluksen käynnistyminen onnistui jokaisella kerralla. Sovelluksen käynnistymisen yhteydessä tosin havaittiin ohjelman pysähtymistä lyhyeksi aikaa, eli ohjelma ei vastannut testaajan komentoihin. Pysähtymisaikaa ei mitattu, mutta sitä kesti arviolta muutamia sekunteja jokaisella käynnistyskerralla.

Kopiointitoiminnot tulkittiin onnistuneiksi, jos kopiotiedostot vastasivat pituudeltaan sekä aloitus- ja lopetusajoiltaan alkuperäistä kaapattua videotiedostoa. Jokainen kopiointitoiminto onnistui tällä tavalla tarkasteltuna.

Vastaavanlaista ohjelman pysähtymistä, jota havaittiin sovelluksen käynnistyksen yhteydessä, havaittiin myös tallennuksen lopetusvaiheessa. Näin ei tapahtunut jokaisella tallennuskerralla, mutta aina kun sitä tapahtui, kaapatun videon pituus oli huomattavasti pidempi kuin aloitus- ja lopetusajankohtien välinen ero.

Ohjelma pysähtyi myös jokaisella kerralla videon kopioinnin ajaksi. Tämä ei kuitenkaan vaikuttanut itse kopioinnin kohteena olleisiin tiedostoihin. Ohjelma vastasi normaalisti testaajan komentoihin kopioinnin valmistuttua. Syitä ohjelman epätavalliseen käyttäytymiseen ei tässä testauksessa selvitetty. Todellisessa tilanteessa tämäntyyppiset ongelmat saattaisivat haitata ohjelman käytettävyyttä tai aiheuttaa häiriöitä muissa toiminnoissa, kuten kommunikoinnissa ulkopuolisten laitteiden kanssa tai tiedon tallennuksessa.

8 Yhteenveto

Opinnäytetyön tarkoituksena oli tutustua kuvan ja äänen tallettamiseen ja käsittelyyn tietokoneessa, tutkia mitä mahdollisuuksia Windows-käyttöjärjestelmä tarjoaa kuvan ja äänen käsittelyyn, sekä toteuttaa videokuvan kaappaussovellus valvomosovellusalustalla.

Videokuvan kaappaussovelluksen osalta tavoite täyttyi siltä osin, että kuvaa saatiin kaapattua ja talletettua tietokoneen muistiin, mutta sovelluksen toiminta ei testissä vastannut odotuksia ohjelman käyttäytymisessä ilmenneiden ongelmien johdosta, joten sovellus vaatii lisäkehitystä, jotta siitä saataisiin loppukäyttäjälle toimiva versio.

AVICap-luokka tarjoaa useita palveluita, joilla sovellus voi kommunikoida kuvan kaappauslaitteiden kanssa. VijeoCitectin ja AVICap-luokan välisen kommunikaation toteutuksessa on kuitenkin se hankaluus, että VijeoCitectissa käytettävät ohjelmointikielet, Cicode ja CitectVBA, eivät tarjoa mahdollisuutta luoda omia tietotyyppejä, joiden avulla saadaan kaikki käyttöjärjestelmän ohjelmointirajapinnan tarjoamat mahdollisuudet käyttöön. Näin ollen laiteajurien ominaisuudet, kyvyt ja tila jäävät VijeoCitectilla tehdyltä sovellukselta ”piiloon”. Sen lisäksi laitteistoa ei kyetä ohjaamaan kaikilla mahdollisilla tavoilla, joita AVICap-luokka tarjoaa. Esimerkkinä CAPTUREPARMS-struktuurissa oleva wTimeLimit, jolla voidaan määritellä kuvankaappauksen aikaraja sekunneissa. Jotta VijeoCitectilla saataisiin luotettavasti toimiva kuvankaappaussovellus, on sovelluksen omiin tietotyyppeihin liittyvä ongelma ratkaistava.

Kuvan ja äänen käsittelyyn tietokoneessa ja Windowsin tarjoamien mahdollisuuksien tutustumisen osalta tavoitteiden voidaan sanoa täyttyneen. Vaikka itse sovelluksen toteutus jäi sovelluksen toiminnan osalta kesken, sai tekijä runsaasti tietoa siitä, mitä kuvankaappauksen toteuttaminen Windows-ympäristössä vaatii.

Lähteet

1 VijeoCitect Online Help. CitectVBA Programming Reference > Integrating CitectVBA with Vijeo Citect > Multithread Considerations with CitectVBA. Luettu 2.4.2010.

2 Vijeo Citect Online Help. Cicode Programming Reference > Using variables > Using Variable Scope. Luettu 28.11.2009.

3 VijeoCitect Online Help. Cicode Programming Reference > Using Arrays. Luettu 5.12.2009.

4 Vijeo Citect Online Help. CitectVBA Programming Reference > Integrating CitectVBA with Vijeo Citect > Calling CitectVBA from Cicode. Luettu 4.12.2009.

5 VijeoCitect Online Help. Cicode Programming Reference > Cicode Errors > Hardware/Cicode Errors.

6 VijeoCitect Online Help. CitectVBA Programming Reference > Understanding CitectVBA Language Basics > Accessing Functions in DLLs > Passing variables ByRef and ByVal.

7 Roman, Steven. Win32 API Programming with Visual Basic. Sebastopol, CA, USA: O'Reilly & Associates Inc. 2000.

8 VijeoCitect Online Help. CitectVBA Programming Reference > Integrating CitectVBA with Vijeo Citect > Calling Cicode from CitectVBA.

9 Wong, Yue-Ling. Digital Media Primer. Upper Saddle River, NJ, USA: Pearson Education Inc. 2009.

10 FireWire (WWW-dokumentti). <http://fi.wikipedia.org/wiki/FireWire>. 23.12.2009. Luettu 12.4.2010.

11 Category:USB plugs. (WWW-dokumentti.) <http://commons.wikimedia.org/wiki/Category:USB_plugs>. 25.2.2009. Luettu 12.4.2010.

12 Burg, Jennifer. Science of Digital Media. Upper Saddle River, NJ, USA: Pearson Education Inc. 2009.

13 DVI. (WWW-dokumentti.) <http://en.wikipedia.org/wiki/DVI>. 13.2.2010. Luettu 14.4.2010.

14 HDMI – High Deinition Multimedia Interface. (WWW-dokumentti.) AfterDawn Oy. <http://fin.afterdawn.com/oppaat/arkisto/high_definition_multimedia_interface_fin.cfm>. 13.7.2009. Luettu 14.4.2010.

15 S-Video. (WWW-dokumentti.) <http://fi.wikipedia.org/wiki/S-Video>. 3.1.2020. Luettu 12.4.2010.

16 Komposiittivideo. (WWW-dokumentti.) <http://fi.wikipedia.org/wiki/Komposiittivideo>. 23.12.2009. Luettu 12.4.2010.

17 SCART. (WWW-dokumentti.) <http://en.wikipedia.org/wiki/SCART>. 29.3.2010. Luettu 14.4.2010.

18 Windows API. (WWW-dokumentti). <http://msdn.microsoft.com/en-us/library/cc433218(VS.85).aspx>. Build date: 7/30/2009. Luettu 24.10.2009.

19 Video Capture. (WWW-dokumentti.) <http://msdn.microsoft.com/en-us/library/dd757692(VS.85).aspx>. Build date: 10/9/2009. Luettu 25.10.2009.

20 Video Capture: Minimal Approach. (WWW-dokumentti.) <http://msdn.microsoft.com/en-us/library/dd757699(v=VS.85).aspx >. Build date: 3/5/2010. Luettu 11.4.2010.

21 About Video Capture. (WWW-dokumentti.) <http://msdn.microsoft.com/en-us/library/dd742882(VS.85).aspx>. Build date: 10/9/2009. Luettu 25.10.2009.

Liite 1: Kuvankaappaussovelluksessa käytettävät DLL-kutsut

```
capCreateCaptureWindow(LPCTSTR lpszWindowName, DWORD dwStyle,
        int x, int y, int nWidth, int nHeight, HWND hWnd,
        int nID);
```

lpszWindowName: NULL-päätteinen merkkijono, mikä sisältää kuvankaappausikkunan nimen.

dwStyle: Kuvankaappausikkunassa käytettävä tyyli.

x: Kuvankaappausikkunan vasemman yläkulman x-koordinaatti.

y: Kuvankaappausikkunan vasemman yläkulman y-koordinaatti.

nWidth: Kuvankaappausikkunan leveys.

nHeight: Kuvankaappausikkunan korkeus.

nID: Kuvankaappausikkunan tunnistin.

Palautusarvo: Kuvankaappausikkunan kahva, jos ikkunan avaus onnistuu. Muussa tapauksessa NULL.

```
SendMessage(HWND hWnd, UINT Msg, WPARAM wParam, LPARAM lParam);
```

hWnd : Viestin vastaanottavan ikkunan kahva.

Msg: Ikkunalle lähetettävä viesti.

wParam: Lähetettävästä viestistä riippuvaa lisäinformaatiota.

lParam: Lähetettävästä viestistä riippuvaa lisäinformaatiota.

Palautusarvo määräytyy lähetettävän viestin mukaan.

```
BOOL VFWAPI capGetDriverDescription(WORD wDriverIndex, LPTSTR
lpszName,INT cbName, LPTSTR lpszVer, INT cbVer );
```

wDriverIndex: Ajurin indeksi väillä 0–9.

lpszNa e: Osoitin NULL-päätteiseen merkkijonoon, mikä sisältää ajurin nimen.

cbName: Em. merkkijonon pituus tavuina.

lpszVer: Osoitin NULL-päätteiseen merkkijonoon, mikä sisältää ajurin version.

cbVer: Em. merkkijonon pituus tavuina.

Palautusarvo on TOSI, jos funktio suoritetaan onnistuneesti, muussa tapauksessa EPÄTOSI.

Liite 2: Kuvankaappausikkunan hallinnassa käytettävät struktuurit

CAPDRIVERCAPS

```
UINT   wDeviceIndex;
BOOL   fHasOverlay;
BOOL   fHasDlgVideoSource;
BOOL   fHasDlgVideoFormat;
BOOL   fHasDlgVideoDisplay;
BOOL   fCaptureInitialized;
BOOL   fDriverSuppliesPalettes;
HANDLE hVideoIn;
HANDLE hVideoOut;
HANDLE hVideoExtIn;
HANDLE hVideoExtOut;
```

wDeviceIndex: Ajurin indeksi.

fHasOverlay: Ilmoittaa kykeneekö laitteisto toimimaan overlay-moodissa.

fHasDlgVideoSource: Ilmoittaa tukeeko laitteisto videon lähteen valintadialogia.

fHasDlgVideoFormat: Ilmoittaa tukeeko laitteisto videon muotoiludialogia.

fHasDlgVideoDisplay: Ilmoittaa tukeeko laitteisto videon uudelleen esitys dialogia.

fCaptureInitialized: Ilmoittaa onko ajuri kytketty ikkunaan onnistuneesti.

fDriveSuppliesPalettes: Ilmoittaa kykeneekö ajuri luomaan omia väripaletteja.

Muut jäsenet eivät ole käytössä Win32 sovelluksissa.

CAPSTATUS

```
UINT     uiImageWidth;
UINT     uiImageHeight;
BOOL     fLiveWindow;
BOOL     fOverlayWindow;
BOOL     fScale;
POINT    ptScroll;
BOOL     fUsingDefaultPalette;
BOOL     fAudioHardware;
BOOL     fCapFileExists;
DWORD    dwCurrentVideoFrame;
DWORD    dwCurrentVideoFramesDropped;
DWORD    dwCurrentWaveSamples;
DWORD    dwCurrentTimeElapsedMS;
HPALETTE hPalCurrent;
BOOL     fCapturingNow;
DWORD    dwReturn;
UINT     wNumVideoAllocated;
UINT     wNumAudioAllocated;
```

uiImageWidth: Kuvan leveys pikseleinä.

uiImageHeight: Kuvan korkeus pikseleinä.

fLiveWindow: Ilmoittaa näyttääkö ikkuna videota esikatselu-tilassa.

Liite 2: Kuvankaappausikkunan hallinnassa käytettävät struktuurit

fOverlayWindow: Ilmoittaa näyttääkö ikkuna videota overlay-tilassa.

fScale: Ilmoittaa skaalaako ikkuna videokuvan näyttöalueen mukaan esikatselu-tilassa

ptScroll: Vasemman yläkulman pikselin x- ja y-koordinaatit.

fUsingDefaultPalette: Ilmoittaa käyttääkö ikkuna oletusväripalettia.

fAudioHardware: Ilmoittaa, onko äänilaitteisto asennettu.

fCapFileExists: Ilmoittaa, onko kaappaustiedosto luotu.

dwCurrentVideoFrame: Ilmoittaa käsiteltyjen kehysten määrän. Sisältää myös hylätyt kehykset.

dwCurrentVideoFramesDropped: Hylättyjen kehysten lukumäärä.

dwCurrentWaveSamples: Käsiteltyjen ääninäytteiden määrä.

dwCurrentTimeElapsedMS: Kaappausaika kaappauksen alusta lähtien millisekunneissa.

hPalCurrent: Käytössä olevan väripaletin kahva.

fCapturingNow: Ilmoittaa onko kaappaus käynnissä.

dwReturn: Virhekoodi.

wNumVideoAllocated: Varattujen videopuskureiden määrä.

wNumAudioAllocated: Varattujen äänipuskureiden määrä.

CAPTUREPARMS

```
DWORD dwRequestMicroSecPerFrame;
BOOL  fMakeUserHitOKToCapture;
UINT  wPercentDropForError;
BOOL  fYield;
DWORD dwIndexSize;
UINT  wChunkGranularity;
BOOL  fUsingDOSMemory;
UINT  wNumVideoRequested;
BOOL  fCaptureAudio;
UINT  wNumAudioRequested;
UINT  vKeyAbort;
BOOL  fAbortLeftMouse;
BOOL  fAbortRightMouse;
BOOL  fLimitEnabled;
UINT  wTimeLimit;
BOOL  fMCIControl;
BOOL  fStepMCIDevice;
DWORD dwMCIStartTime;
DWORD dwMCIStopTime;
BOOL  fStepCaptureAt2x;
UINT  wStepCaptureAverageFrames;
DWORD dwAudioBufferSize;
BOOL  fDisableWriteCache;
UINT  AVStreamMaster;
```

Liite 2: Kuvankaappausikkunan hallinnassa käytettävät struktuurit

dwRequestMicroSecPerFrame: Haluttu kuvanopeus mikrosekunneissa.
fMakeUserHitOKToCapture: Jos tämä on TOSI, AVICap ilmoittaa käyttäjälle kaappauksen aloituksesta dialogilla.
wPercentDropForError: Suurin sallittu hylättyjen kehysten määrä prosenteissa.
fYield: Jos tama on TOSI, kaappausikkuna suorittaa kaappauksen erillisessä taustaprosessissa.
dwIndexSize: Kaapattujen kehysten maksimimäärä AVI-tiedostossa.
wChunkGranularity: AVI-tiedoston koko tavuina.
fUsingDOSMemory: Ei käytössä Win32 sovelluksissa.
wNumVideoRequested: Videopuskureiden maksimimäärä.
fCaptureAudio: Ilmoittaa kaapataanko ääntä kuvan mukana.
wNumAudioRequested: Äänipuskureiden maksimimäärä.
vKeyAbort: Kaappauksen keskeytysnäppäin.
fAbortLeftMouse: Jos tämä on TOSI, kaappaus lopetetaan hiiren vasenta painiketta painamalla.
fAbortRightMouse: Jos tämä on TOSI, kaappaus lopetetaan hiiren oikeaa painiketta painamalla.
fLimitEnabled: Jos tama on tosi kaappaus lopetetaan wTimeLimit:in ilmoittaman ajan jälkeen.
wTimeLimit: Kaappauksen aikaraja sekunneissa.
fMCIControl: Jos tämä on TOSI, AVICap ohjaa MCI-yhteensopivia laitteita kaappauksessa.
fStepMCIDevice: Jos tämä on TOSI, askelkaappaus MCI-laitetta käyttämällä on sallittu.
dwMCIStartTime: MCI- laitteen kaappauksen aloitushetki millisekunneissa.
dwMCIStopTime: MCI-laitteen kaappauksen lopetushetki millisekunneissa.
fStepCaptureAt2x: Jos tämä on TOSI, kuvaa kaapataan tuplaresoluutiolla määriteltyyn resoluutioon verrattuna.
dwAudioBufferSize: Äänipuskurin koko.
fDisableWriteCache: Ei käytössä Win32 sovelluksissa.
AVStreamMaster: Hallitseeko äänidata kellotusta AVI-tiedostoon kirjoitettaessa.

Liite 3: Windowsin multimediaviestit

WM_CAP_START = 0x400
WM_CAP_UNICODE_START = WM_CAP_START +100
WM_CAP_PAL_SAVEA = WM_CAP_START + 81
WM_CAP_PAL_SAVEW = WM_CAP_UNICODE_START + 81
WM_CAP_UNICODE_END = WM_CAP_PAL_SAVEW
WM_CAP_ABORT = WM_CAP_START + 69
WM_CAP_DLG_VIDEOCOMPRESSION = WM_CAP_START + 46
WM_CAP_DLG_VIDEODISPLAY = WM_CAP_START + 43
WM_CAP_DLG_VIDEOFORMAT = WM_CAP_START + 41
WM_CAP_DLG_VIDEOSOURCE = WM_CAP_START + 42
WM_CAP_DRIVER_CONNECT = WM_CAP_START + 10
WM_CAP_DRIVER_DISCONNECT = WM_CAP_START + 11
WM_CAP_DRIVER_GET_CAPS = WM_CAP_START + 14
WM_CAP_DRIVER_GET_NAMEA = WM_CAP_START + 12
WM_CAP_DRIVER_GET_NAMEW = WM_CAP_UNICODE_START + 12
WM_CAP_DRIVER_GET_VERSIONA = WM_CAP_START + 13
WM_CAP_DRIVER_GET_VERSIONW = WM_CAP_UNICODE_START + 13
WM_CAP_EDIT_COPY = WM_CAP_START + 30
WM_CAP_END = WM_CAP_UNICODE_END
WM_CAP_FILE_ALLOCATE = WM_CAP_START + 22
WM_CAP_FILE_GET_CAPTURE_FILEA = WM_CAP_START + 21
WM_CAP_FILE_GET_CAPTURE_FILEW = WM_CAP_UNICODE_START + 21
WM_CAP_FILE_SAVEASA = WM_CAP_START + 23
WM_CAP_FILE_SAVEASW = WM_CAP_UNICODE_START + 23
WM_CAP_FILE_SAVEDIBA = WM_CAP_START + 25
WM_CAP_FILE_SAVEDIBW = WM_CAP_UNICODE_START + 25
WM_CAP_FILE_SET_CAPTURE_FILEA = WM_CAP_START + 20
WM_CAP_FILE_SET_CAPTURE_FILEW = WM_CAP_UNICODE_START + 20
WM_CAP_FILE_SET_INFOCHUNK = WM_CAP_START + 24
WM_CAP_GET_AUDIOFORMAT = WM_CAP_START + 36
WM_CAP_GET_CAPSTREAMPTR = WM_CAP_START + 1
WM_CAP_GET_MCI_DEVICEA = WM_CAP_START + 67
WM_CAP_GET_MCI_DEVICEW = WM_CAP_UNICODE_START + 67
WM_CAP_GET_SEQUENCE_SETUP = WM_CAP_START + 65
WM_CAP_GET_STATUS = WM_CAP_START + 54
WM_CAP_GET_USER_DATA = WM_CAP_START + 8
WM_CAP_GET_VIDEOFORMAT = WM_CAP_START + 44
WM_CAP_GRAB_FRAME = WM_CAP_START + 60
WM_CAP_GRAB_FRAME_NOSTOP = WM_CAP_START + 61
WM_CAP_PAL_AUTOCREATE = WM_CAP_START + 83
WM_CAP_PAL_MANUALCREATE = WM_CAP_START + 84
WM_CAP_PAL_OPENA = WM_CAP_START + 80
WM_CAP_PAL_OPENW = WM_CAP_UNICODE_START + 80
WM_CAP_PAL_PASTE = WM_CAP_START + 82
WM_CAP_SEQUENCE = WM_CAP_START + 62
WM_CAP_SEQUENCE_NOFILE = WM_CAP_START + 63
WM_CAP_SET_AUDIOFORMAT = WM_CAP_START + 35

Liite 3: Windowsin multimediaviestit

WM_CAP_SET_CALLBACK_CAPCONTROL = WM_CAP_START + 85
WM_CAP_SET_CALLBACK_ERRORA = WM_CAP_START + 2
WM_CAP_SET_CALLBACK_ERRORW = WM_CAP_UNICODE_START + 2
WM_CAP_SET_CALLBACK_FRAME = WM_CAP_START + 5
WM_CAP_SET_CALLBACK_STATUSA = WM_CAP_START + 3
WM_CAP_SET_CALLBACK_STATUSW = WM_CAP_UNICODE_START + 3
WM_CAP_SET_CALLBACK_VIDEOSTREAM = WM_CAP_START + 6
WM_CAP_SET_CALLBACK_WAVESTREAM = WM_CAP_START + 7
WM_CAP_SET_CALLBACK_YIELD = WM_CAP_START + 4
WM_CAP_SET_MCI_DEVICEA = WM_CAP_START + 66
WM_CAP_SET_MCI_DEVICEW = WM_CAP_UNICODE_START + 66
WM_CAP_SET_OVERLAY = WM_CAP_START + 51
WM_CAP_SET_PREVIEW = WM_CAP_START + 50
WM_CAP_SET_PREVIEWRATE = WM_CAP_START + 52
WM_CAP_SET_SCALE = WM_CAP_START + 53
WM_CAP_SET_SCROLL = WM_CAP_START + 55
WM_CAP_SET_SEQUENCE_SETUP = WM_CAP_START + 64
WM_CAP_SET_USER_DATA = WM_CAP_START + 9
WM_CAP_SET_VIDEOFORMAT = WM_CAP_START + 45
WM_CAP_SINGLE_FRAME = WM_CAP_START + 72
WM_CAP_SINGLE_FRAME_CLOSE = WM_CAP_START + 71
WM_CAP_SINGLE_FRAME_OPEN = WM_CAP_START + 70
WM_CAP_STOP = WM_CAP_START + 68