Videotallenteen lähetys pilvipalveluun automaatiojärjestelmän häiriötilanteessa

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Tietojenkäsittelyn koulutus

Syksy 2021

Jori Nikkilä

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutus Tiivistelmä
Tekijä Jori Nikkilä Vuosi 2021
Työn nimi Videotallenteen lähetys pilvipalveluun automaatiojärjestelmän häiriötilanteessa
Ohjaaja Lasse Seppänen

Opinnäytetyön tavoitteena oli toteuttaa teollisuusautomaatiojärjestelmiä tuottavalle Orfer Oy:lle kamerajärjestelmä automaatiojärjestelmän häiriötilanteiden videotallennukseen. Järjestelmän tarkoituksena oli puskuroida IP-kameran stream-kuvaa jatkuvasti muutaman sekunnin ajalta tietokoneen keskusmuistiin ja automaatiojärjestelmästä virheen yhteydessä tulevan triggeritiedon laukaisemana tallentaa puskuroitu kuva tietokoneen tallennustilaan ja lähettää video edelleen Microsoft Azure pilvipalveluun.

Teoriaosuudessa käydään ensin läpi tietoja työn tilaajasta, yleistietoa teollisuusautomaatiojärjestelmistä, teollisuusroboteista, IP-kameroista ja Microsoft Azuren palvelumalleista ja tallennusratkaisuista.

Käytännönosuudessa kuvataan kamerajärjestelmän toteutus vaiheittain ja käydään läpi valittuja toteutusmenetelmiä.

Opinnäytetyön tuloksena saatiin toteutettua kuvatunlainen järjestelmä ja se asennettiin tilaajan omaan automaatiojärjestelmään, jossa sen käytöstä kerätään kokemuksia.

Avainsanat Teollisuusautomaatiojärjestelmä, IP-kamera, teollisuusrobotti, Microsoft Azure.

Sivut 33 sivua ja liitteitä 1 sivu.

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Author Jori Nikkilä
Subject Sending a video recording to the cloud service in the event of an automation system failure
Supervisor Lasse Seppänen

Abstract
Year 2021

The aim of the thesis was to implement a camera system for video recording of industrial automation system failures for Orfer Oy. The purpose of the system was to continuously buffer stream image from the IP camera for a few seconds into computer's main memory and when triggered by the automation system, save the buffered image to the computer's storage and forward the video to Microsoft Azure cloud service.

The theoretical part first reviews information about the client, general information about industrial automation systems, industrial robots, IP cameras and Microsoft Azure service models and its storage solutions.

In the practical part, the implementation of the camera system is described in stages and the selected implementation methods are reviewed.

As a result of the thesis, targeted system was implemented and it was installed in the customer's own automation system, where experiences from its use are collected.

Keywords Industrial automation system, IP camera, industrial robot, Microsoft Azure.

Pages 33 pages and appendices 1 page.

Sisällys

Liitteet

Kuvat

1 Johdanto

Tämä opinnäytetyö käsittelee teollisuusautomaatiojärjestelmiä, robotiikkaa, IP-kameratekniikkaa, Microsoft Azure pilvipalvelua ja toteutettavana olevan järjestelmän ohjelmointia.

Työn teoriaosuudessa kuvataan automaatiojärjestelmien perusominaisuuksia, niiden ohjausjärjestelmiä ja yleisimpiä virhetilanteita. Lisäksi käydään läpi IP-kameroiden ominaisuuksia, Microsoft Azure pilvipalvelun tallennusratkaisuja, tiedonsiirtoa Azureen ja lisäksi sen tarjoamia palvelumalleja.

Teollisuusautomaatio lisääntyy ja kehittyy jatkuvasti ja ihmisiä voidaan vapauttaa yksinkertaisesta, fyysisesti kuormittavasta ja usein vaarallisestakin työstä muihin tehtäviin automaatiojärjestelmien ja robottien avulla. Hyötynä saadaan myös tasaisempaa laatua robottien ja automaation suorittaessa tehtävänsä aina samalla tavalla ja tarkkuudella.

Automaatiojärjestelmät vaativat kuitenkin aina ihmisen valvontaa ja puuttumista. Järjestelmissä voi esiintyä erilaisia virheitä käsiteltävien kappaleiden poikkeamien, ohjelmistovirheiden tai mekaanisten kulumien tai vikojen takia. Monesti sama virhe voi toistua useinkin, ilman että sen varsinaista juurisyytä koskaan korjataan tai sen jäljille ei välttämättä yrityksistä huolimatta päästä, lisäksi jatkuva linjan virhetilanteiden seuranta ihmissilmin vaatisi kohtuuttomasti resursseja.

Tässä työssä kehitettävällä järjestelmällä pyritään luomaan työkalu automaatiolinjoissa ilmenevien virhetilanteiden juurisyiden selvitykseen ja niiden korjaamiseen pelkän toistuvan virheestä toipumisen sijaan. Lisäksi kehitettävän järjestelmän tallenteiden lähetysominaisuus pilvipalveluun tarjoaa mahdollisuuden keskitettyyn tiedon- ja pääsynhallintaan, sekä digitaalisten palveluiden tarjonnan lisäämiseen.

Tutkimuskysymykset:
Kuinka toteuttaa IP-kameran kuvaa puskuroiva ja tallentava ohjelmisto?
Kuinka lähettää tallennettu videotiedosto Microsoft Azureen?

2 Työn tarkoitus ja tavoitteet

Opinnäytetyössä on tarkoitus toteuttaa valvontakameratyyppinen ratkaisu teollisuusautomaatiojärjestelmän häiriötilanteiden videotallennukseen. Työssä kuvataan kuinka Microsoft Visual Studio-ympäristössä voidaan toteuttaa työpöytäohjelmisto, joka puskuroi IP-kameran videokuvaa, tallentaa videon automaatiojärjestelmän laukaisemalla hetkellä ja lähettää sen eteenpäin Microsoft Azure-pilvipalveluun.

2.1 Tavoitteet

Opinnäytetyön käytännön osuudessa tavoitteena on toteuttaa helppokäyttöinen ja yksinkertainen sovellus, jotta se on käyttöönotettavissa teollisuusautomaatiojärjestelmien ja tietotekniikan perustiedot omaavan henkilön toimesta. Automaatiojärjestelmään integroinnin tulisi olla helposti toteutettavissa huomioiden erilaiset ja eri aikakausien laitteistot. Sovelluksen tulee olla riittävän kevyt, jottei se käytä tietokoneen resursseja kohtuuttoman paljoa käyttötarkoitukseensa nähden. Sovelluksen tulee myös lähettää videotiedosto automaattisesti pilvipalveluun ja lähetyksen epäonnistuessa puskuroida videotallenteet tietokoneen tallennustilaan. Tuotettavan ohjelmiston on tarkoitus toimia mahdollisesti tulevaisuudessa työn toimeksiantajan asiakkailleen tarjoaman tukipalvelun vianselvitystyökaluna.

2.2 Työn rajaaminen

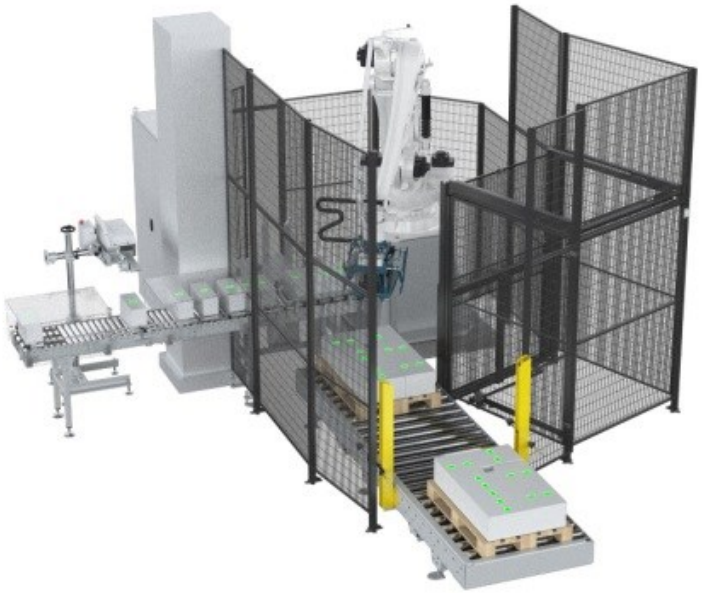
Työtä rajataan siten, ettei siinä käsitellä teollisuusautomaatiojärjestelmiä, pilvipalveluita ja video- ja kameratekniikkaa, kuin opinnäytetyöhön liittyen.
Teollisuusautomaatiojärjestelmillä tarkoitetaan tässä työssä kappaleenkäsittelyyn liittyviä järjestelmiä. Lukijalla oletetaan olevan vähintään perustason tiedot ja ymmärrys ohjelmoinnista ja sen käsitteistä sekä pilviteknologiasta ja sen palvelumalleista ja niiden hyödyistä. Työssä ei kuvata toimeksiantajan palveluprosesseja.

3 Toimeksiantaja Orfer Oy

Opinnäytetyötä tehdään toimeksiantona Orfer Oy:lle, joka kehittää ja valmistaa robottiteknologiaa hyödyntäviä kappaleenkäsittelyjärjestelmiä teollisuuden eri aloille Suomessa ja ulkomailla. Orfer tarjoaa myös huolto-, varaosa-, modernisointi- ja päivystyspalveluita toimittamiinsa järjestelmiin. Lisäksi Orfer tarjoaa valmistuspalveluita teräsrakenteisiin ja kokoonpanoihin. Orfer maahantuo ja myy Kawasaki- ja Shibaura Machine-teollisuurobotteja. (Orfer, n.d.-a)

Orferin tyypillisimmät toimitukset keskittyvät toisiopakkaamiseen. Esimerkiksi laatikointiin, sekä laatikoiden tai vastaavien tuotteiden lavaamiseen kuormalavalle, kuten kuvassa 1.

Kuva 1. Tyypillinen Orferin laatikoiden lavausta varten valmistama robottisolu (Orfer, n.d.-b).



Orfer Oy on perustettu vuonna 1970 ja sen päätoimipaikka on Orimattilassa, yhtiöllä on myös 2010-luvulla perustetut tytäryhtiöt Venäjällä ja Kanadassa ja se työllistää noin 100 henkilöä. Yrityksen päätoimiala 1970-luvulla oli sahateollisuuden osien, koneiden ja laitteiden valmistus alihankintana. 1980-luvulla yrityksen tuotanto alkoi painottumaan automaattisiin materiaalinkäsittelyjärjestelmiin. Vuonna 1995 Orfer alkoi maahantuomaan Kawasaki ja Toshiba -robotteja vuonna 2004 ja näitä robottimerkkejä yritys käyttää myös

omissa toimituksissaan. Toshiba vaihtoi vuonna 2020 nimensä Shibaura Machineksi. (Orfer, n.d.-c)

4 Teollisuusautomaatiojärjestelmät

Automaation ja robotiikan tarkoituksena on vapauttaa ihmisiä yksinkertaisesta, fyysisesti kuormittavasta ja toistuvasta työstä muihin tehtäviin. Useimmiten automaatiojärjestelmät myös suoriutuvat näistä tehtävistä ihmisiä tehokkaammin, nopeammin ja tarkemmin. (Koskinen, 2017, s. 4)

Automatisoidussa kappaleenkäsittelyjärjestelmässä koneita, toimilaitteita ja robotteja ohjataan ja valvotaan automaattisesti erilaisten valvontajärjestelmien, ohjelmoitavien logiikoiden ja muiden ohjausjärjestelmien avulla, ilman ihmisen jatkuvaa ohjaustarvetta. (Valmistajat, n.d.)

Automaatiojärjestelmä voi koostua esimerkiksi kappaleita tehtaassa kuljettavista kuljettimista, servoista, paineilmasylintereistä tai muista toimilaitteista, roboteista ja mistä tahansa näiden yhdistelmistä. Automaatiojärjestelmä voi olla yksittäinen ohjelmoitava laite tai kokonainen tehdasjärjestelmä. (Holloway, 2019)

Automaatiojärjestelmissä voi esiintyä virhetilanteita, jolloin linjan toiminta keskeytyy ja tuotanto pysähtyy. Syynä tällaiseen voi olla laitevika, suunnittelu- tai ohjelmistovirhe tai ihmisen aiheuttama virhe väärien toimintatapojen tai kunnossapidollisten virheiden takia. Myöskään käsiteltävien kappaleiden tai esineiden virheellisen olomuodon aiheuttamia virheitä ei voida poissulkea. (Ajo ym., n.d., s. 120)

Sanan robotti on keksinyt tšekkiläinen Josef Čapek vuonna 1920, jonka veli, kirjailija Karel Čapek, käytti sanaa näytelmässään R.U.R. Rossum’s Universal Robots. Sana tulee tšekin sanasta robota, joka tarkoittaa maanomistajalle tehtävää veron omaista työtä. (Lehtinen, n.d.)

Sarjatuotantokäyttöön ensimmäinen teollisuusrobotti on toimitettu vuonna 1961, kyseessä oli yhdysvaltalainen Unimate-robotti joka otettiin käyttöön autoteollisuudessa. Siinä oli kuusi liikeakselia ja se pystyi käsittelemään noin 230kg:n kuormaa, omamassan ollessa noin 1800Kg. (Robot hall of fame, n.d.)

Eniten teollisuusrobotteja on käytössä Kiinassa, jossa niiden määrä oli vuonna 2019 noussut 783 000 kappaleeseen. Toiseksi eniten robotteja oli käytössä Japanissa ja kolmanneksi eniten Yhdysvalloissa. Yhteensä vuonna 2020 käytössä olevien robottien määräksi arvioitiin noin 2,7 miljoonaa kappaletta. (IFR, 2020.)

Robotteja käytetään muun muassa erilaisissa hitsaussovelluksissa, kappaleiden käsittelyssä, erilaisten valukoneiden palvelijoina, maalauksessa, pakkauksessa, laatikoinnissa ja lavauksessa, kokoonpanotehtävissä erityisesti elektroniikan valmistuksessa sekä erilaisissa kappapaleiden työstötehtävissä. Koska käyttötarpeet ovat hyvin erilaisia, on myös roboteista kehitetty lukuisia eri malleja (Kuvat 2-4). (Shake, n.d.)

Kuva 2. Kuusiakselinen robotti, käytetään esimerkiksi autoteollisuudessa pistehitsaukseen (Kawasaki, n.d.-a)

Kuva 3. Neliakselinen robotti, käytetään muun muassa kappaleiden lavaukseen(Kawasaki, n.d.-b)

Kuva 4. Neliakselinen SCARA-robotti, käytetään esimerkiksi komponenttien kokoonpanoon (Shibaura-Machine n.d.)

Kuva 5. Neliakselinen suurnopeuksinen poimintarobotti (Kawasaki, n.d.-c)

5 Tyypilliset häiriötilanteet Orferin toimittamien järjestelmien tapauksessa

Orferin tapauksessa tyypillisessä häiriötilanteessa voi olla kyseessä esimerkiksi huonosti linjalla oleva kappale, jonka johdosta linja ei toimi odotetulla tavalla. Tilanteessa voi tapahtua esimerkiksi robotin törmäys huonosti poimintapaikalla olevaan kappaleeseen, joka aiheuttaa häiriötilanteen ja vaatii ihmisen tekemää selvitystä siitä, miten ja miksi tilanne on syntynyt.

Häiriöt voivat toistua harvoin ja ihminen ei niitä ole välttämättä näkemässä. Tällaiset tapaukset voivat helposti aiheuttaa useita yhteydenottoja samantyyppisestä häiriöstä Orferin asiakkailleen tarjoamaan tukipalveluun. Koska myöskään asiakkaan edustaja ei välttämättä ole tietoinen mistä häiriö johtuu, korjataan usein vain häiriötilanne linjalla ja tuotantoa jatketaan. Alkuperäinen juurisyy häiriötilanteeseen jää selvittämättä ja tilanne voi toistua säännöllisestikin, vaikka juurisyyn korjauksella siitä voitaisiin päästä kokonaan eroon.

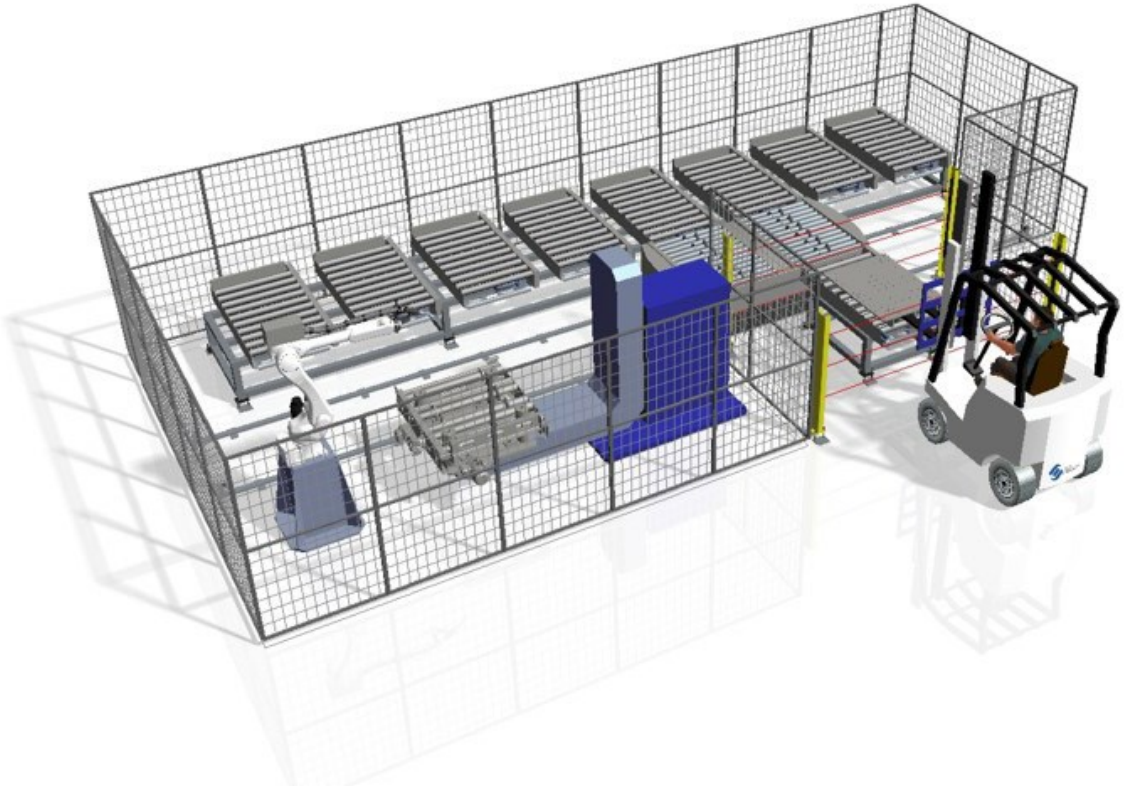
Tämän tyyppisissä tilanteissa työn tuloksena tavoiteltava kamerajärjestelmä voisi olla suureksi avuksi, kun häiriötilanteeseen johtanut tapahtumaketju päästäisiin näkemään videolta jälkikäteen ja häiriön aiheuttava ongelma voitaisiin korjata pelkän häiriöstä toipumisen sijaan. Videoiden lähetys pilvipalveluun tarjoaa myös keskitetyn tallennusratkaisun, pääsynhallinnan ja toimii osaltaan digitaalisena palveluna.

Työssä testikohteena käytettävä automaatiojärjestelmä on tilaajan tavanomaisista toimituksista poiketen robottihitsausjärjestelmä. Se koostuu hitsausrobotista, kuljettimista, siirtovaunusta ja pyörityspöydästä (Kuva 6), johon hitsattavat kappaleet lukitaan kiinni hitsauksen ajaksi, pöydän liikkeet ovat robotin ohjauksessa. Osa kuljettimista toimii varastopaikkoina, joilta robotti valitsee hitsaukseen tuotteita niiden prioriteetin mukaan.

Linjan ohjauksesta huolehtii Siemensin ET200S-PLC, Kawasaki-robotin E-sarjan kontrolleri sekä Beijer Electronicsin Windows-pohjainen paneeli-PC, joka toimii myös linjan käyttöliittymänä. Ohjausjärjestelmän muodostavat laitteet kommunikoivat keskenään LAN-verkon välityksellä.

Tämän tyyppisessä järjestelmässä voi myös tapahtua erilaisia häiriötilanteita, joiden syyt voivat olla selvitettävissä videolta jälkikäteen. Kyseessä voivat olla esimerkiksi jonkin toimilaitteen vajavaisesta toiminnasta tai robotin törmäyksestä johtuvat häiriöt jotka keskeyttävät linjan toiminnan. Ajatuksena on testata toteutettavan järjestelmän toimivuus ja kerätä siitä kokemuksia kyseisessä robottisolussa.

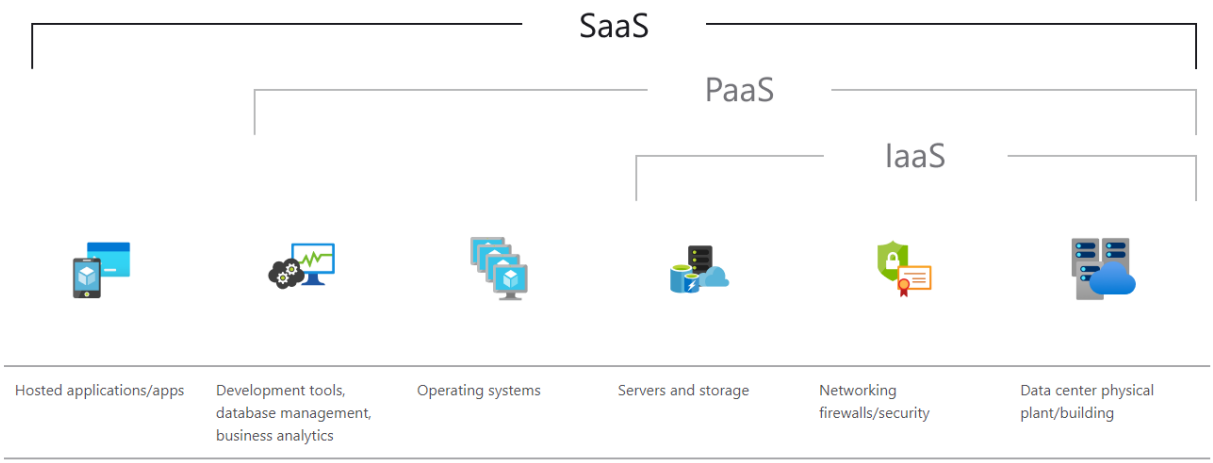
Kuva 6. Hitsausjärjestelmän 3D-kuva (Orfer, n.d.)



6 Microsoft Azure ja sen palvelumallit

Microsoft Azure on Microsoftin vuonna 2010 nimellä Windows Azure julkaisema pilvipalvelu, joka nimettiin vuonna 2014 Microsoft Azureksi (Itewiki n.d.). Azure tarjoaa kolmea julkipilvelle tyypillistä palvelumallia (Kuva 7), IaaS, PaaS ja SaaS (Microsoft, n.d.-a).

Kuva 7. Pilven palvelumallit (Microsoft, 2021)



6.1 Infrastructure-as-a-Service

IaaS (Infrastructure-as-a-Service) tarkoittaa pilvipalvelun tuottajan tarjoamia fyysisiä resursseja, kuten servereitä ja virtuaalikoneita ja näiden tarjoamaa laskentatehoa, lisäksi on erilaisia tietovarastoja, verkkoja ja käyttöjärjestelmiä joita voidaan hallinnoida keskitetysti internetin kautta. (Microsoft, 2021b)

6.2 Platform-as-a-Service

PaaS (Platform-as-a-Service) tarjoaa kehitystyökalut esimerkiksi sovelluskehitykseen ja niiden ajamiseen ja julkaisemiseen pilvipalvelun kautta. Tässä mallissa palveluntarjoaja huolehtii infrastruktuurin perustamisesta ja ylläpidosta kokonaisuudessaan ja tilaaja vastaa ise sovelluksistaan. (Microsoft, 2021b)

6.3 Software-as-a-Service

SaaS (Software as a Service) -mallissa pilvipalveluntarjoaja huolehtii infrastruktuurista ja pilvipalvelun kautta tarjottavista palveluista ja ohjelmistoratkaisuista ja näiden perustamisesta sekä ylläpidosta kokonaisuudessaan, eikä asiakkaalle jää tehtäväksi kuin korkeintaan joitain sovelluksen konfigurointitehtäviä. (Microsoft, 2021b)

6.4 Azure Blob Storage datan tallentamisessa

Azuren Blob Storage on Microsoftin skaalautuva IaaS-palvelu johon voidaan tallentaa rakenteetonta dataa, esimerkiksi tässä tapauksessa videoita. Storageen voidaan luoda muun muassa containereita, jota voi ehkä ajatella kansiona, container sisältää blobeja ja yksi tallennettava tiedosto muodostaa yhden blobin. Blobille voidaan luoda jaettava linkki, johon voidaan määritellä luku- ja pääsyoikeudet. (Snaplogic, n.d.)

Tallennettavaan dataan voidaan liittää itse määriteltyä metadataa key/value-tyyppisesti, kuten tämän työn yhteydessäkin tullaan tekemään. Tämä mahdollistaa tiedoston liittämisen esimerkiksi tiettyyn linjaan, asiakkaaseen tai muuhun mahdolliseen identifiointitietoon Azuressa. (Microsoft, 2021)

7 IP-kamerat

IP-kamera on valvontakamera joka kytketään tietoliikenneverkkoon ethernet-kaapeloinnilla tai langattoman WLAN-verkon välityksellä. Kamerassa voi olla PoE-virransyöttö, jolloin kamera saa myös käyttöjännitteensä verkkokaapelia pitkin mikäli kaapeli on kytketty PoE-ominaisuudella varustettuun verkkolaitteeseen tai kaapelin väliin on asennettu PoE-injektori. Kamera voi olla yhteydessä internetiin, jolloin sen tuottamaa kuvaa voidaan seurata mistä tahansa. (Tilavahti, n.d.)

IP-kameroissa on Web-käyttöliittymä, jolloin kameraa ja sen asetuksia ja kuvasäätöjä päästään hallinnoimaan internet-selaimen kautta tietokoneella tai mobiililaitteella ilman erillistä työpöytäohjelmistoa. IP-kamera voi tallentaa kuvaa muistikortille, mikäli kamera on varustettu kyseisellä ominaisuudella, tai lähettää kuvia esimerkiksi sovellukseen tai sähköpostiin liikkeentunnistuksen aktivoituessa. (Tilavahti, n.d.)

Yleensä kuitenkin kameralla tuotetaan jatkuvaa kuvaa streamina, jota luetaan NVR (Network Video Recorder) -tallentimella. Tallennin voi olla erillinen, varta vasten tähän tarkoitukseen kehitetty, verkossa toimiva laite tai ohjelmisto tietokoneella. (Swann, 2018)

8 Kameratallennusjärjestelmän toteutus

Työn tilaajalla oli muutamia toiveita niiden ominaisuuksien suhteen, joiden pohjalta sovellusta tulisi kehittää:

- Kamerariippumaton, ohjelmiston täytyisi siis lukea ja tallentaa IP-kameran tarjoamaa stream-kuvaa kamerasta riippumatta.
- Videokuvan puskurointi mahdollista muutamien viimeisten sekuntien ajalta ennen kuluvaa aikaa.
- Videon tallennus täytyisi pystyä triggaamaan automaatiojärjestelmästä.
- Tallennettu videotiedosto tulisi pystyä lähettämään automaattisesti Microsoft Azure -pilvialustalle.
- Videotiedoston mukana tulisi pystyä lähettämään metadataa Azureen.
- Ohjelmiston tulisi olla helposti käyttöönotettavissa myös asiaan perehtymättömämmän henkilön toimesta.

8.1 Kehitysympäristö

Kehitysympäristöksi valittiin Microsoft Visual Studion siitä kertyneen kokemuksen vuoksi. Ohjelmointikielenä käytettin C#:a. Lisäksi tarvittiin projektiin sopiva SDK, eli Software Development Kit. Aluksi tutkittiin lupaavalta vaikuttanutta Ozeki Camera SDK:ta, mutta sen korkeahkon kustannuksen takia selvitystyötä jatkettiin.

Seuraavana vaihtoehtona oli VisioForge, mutta sekin hylättiin hinnoittelun takia. Lopulta löytyi Emgu CV, joka on niin sanottu wrapperi, joka tuo OpenCV-kuvankäsittelykirjaston saataville .NET-ympäristöön. (Emgu, n.d.).

OpenCV on konenäkö-, kuvantunnistus- ja keinoälyohjelmointiin kehitetty avoimen lähdekoodin kirjasto. OpenCV mahdollistaa esimerkiksi kasvojentunnistuksen, liikkuvien kohteiden seurannan, 3D-kuvauksen, silmien liikkeiden seurannan ja lukemattomia muita erilaisia kuvankäsittely ja kuvantunnistustehtäviä (OpenCV, n.d.). Sillä on noin 47 000

käyttäjän muodostama yhteisö ja 5.10.2021 mennessä sillä oli yli 23 miljoonaa latauskertaa (Sourceforge, n.d.).

Kehityskäyttöön työn tilaajalta löytyi kaksi eri kameraa. HikVision DS-2CD2041G1 ja Axis M1065-L joista ensimmäiseksi testaukseen otettiin Axisin kamera. Molemmat kamerat ovat IP-kameroita ja niistä löytyy stream-ominaisuus.

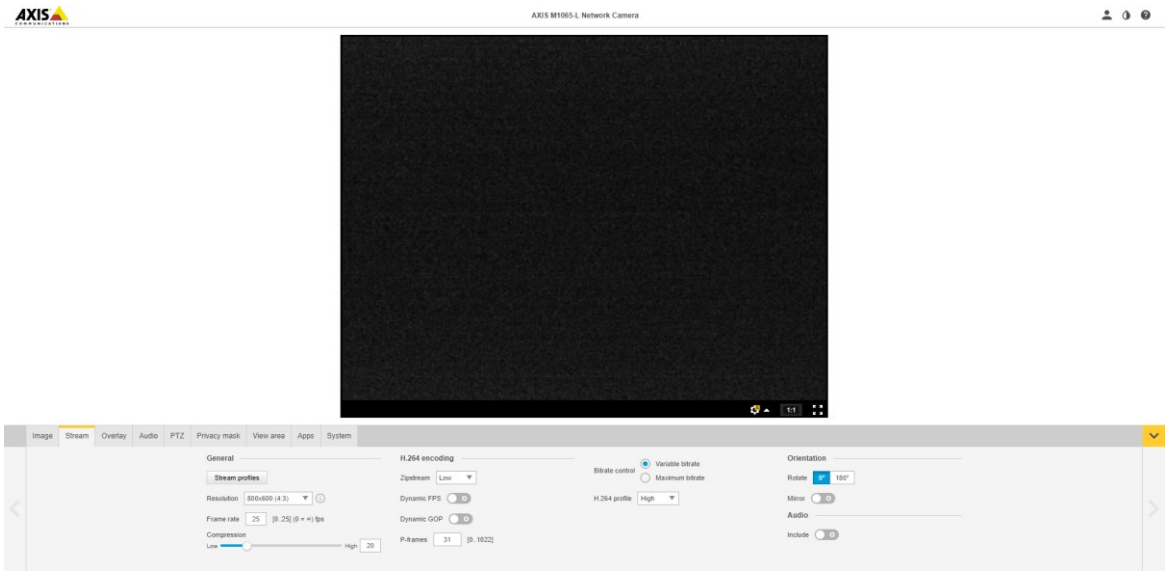
8.2 Sovelluksen toteutus

Seuraavissa kappaleissa kuvataan sovelluksen toteutusta ja siinä käytettyjä ohjelmallisia ratkaisuja ja niiden ominaisuuksia. Sovellus pyritään tekemään mahdollisimman pitkälle valmiiksi testiympäristössä, ennen sen ja laitteiston siirtoa varsinaiseen testikohteeseen.

Ensimmäiseksi haluttiin livekuvaa kamerasta, joten uusi kamera otettin käyttöön kytkemällä se lähiverkkoon ja kirjautumalla kameran ohjeiden mukaan web-käyttöliittymään. Axis-kameran käyttöliittymä esitelty kuvassa 8. Käyttöliittymässä luotiin ensin käyttäjätunnukset kirjautumista varten ja tämän jälkeen asetettiin kameralle kiinteän IP-osoitteen.

Seuraavaksi kameran kuva-asetuksista säädettiin resoluutioksi 800x600 ja frame rate-lukemaksi 20. Videokuva muodostuu yksittäisistä nopealla taajuudella otetuista kuvista ja frames per second eli FPS ilmaisee sitä, kuinka monta kuvaa kamera tuottaa yhden sekunnin aikana.

Kuva 8. Axis web-käyttöliittymä (Kuvakaappaus)



Kameravalmistajan nettisivuilta selvitettiin kyseisen kameramallin stream-url, jota käyttäen kameran kuvaa voidaan näyttää selaimessa menemättä käyttöliittymäsivulle. Samaa url-osoitetta käytetään myös luettaessa streamia ohjelmallisesti. Tässä tapauksessa osoite oli http://<username>:<password>@<ip-address>/axis-cgi/mjpg/video.cgi? Kirjautumistunnuksilla ja IP-osoitteella täydennettynä osoitteella saatiin kuva näkymään selaimessa.

8.3 Livekuvan näyttäminen sovelluksessa

Sovelluksen kehitys aloitettiin luomalla uusi Windows Forms Application -projekti Visual Studioon. Seuraavaksi projektille asennettiin NuGet-paketinhallintatyökalulla Emgu.CV-paketti.

Sovelluksen pääikkunaan lisättiin imageBox-objekti jossa kuvaa tultaisiin näyttämään, objektin kooksi määriteltiin 800x600 pikseliä.

Ohjelmaan kirjoitettiin Emgu CV:n dokumentaatiota hyödyntäen metodi, jolla videokaappaus päästäisiin tekemään kameran stream-osoitteesta (Ohjelmakoodi 1) ja joka kutsuu metodia

jossa UMat-luokkaa hyödyntäen videokuvan frameja asetetaan kuvaikkunaan sitä mukaa kun niitä kameran streamista luetaan (Ohjelmakoodi 2).

UMat-luokka hyödyntää kuvan käsittelyyn tietokoneen grafiikkaprosessoria eli GPU:ta (Graphics Processing Unit) ja sen keskusmuistia, mikäli näitä ei ole saatavilla, käytetään keskussuoritinta eli CPU:ta (Central Processing Unit) ja sen keskusmuistia. Mat-luokkaa käyttäen voitaisiin käsittely tehdä käyttäen ainoastaan tietokoneen keskussuoritinta. UMat-luokkaa käytetään kuvan ominaisuuksien tallentamiseen. Se koostuu kahdesta osasta joista ensimmäinen on Header, joka sisältää muun muassa kuvan kokotiedot. Toinen osa on Pointer, joka sisältää kuvan jokaisen pikselin väriarvot. (Tutorialspoint, n.d.)

Ohjelmakoodi 1. Kuvan kaappaaminen streamista.

```
private void Liveview()

        {
            if (streamAddress != "")
            {
                if (_captureInProgress)
                {
                    View_1.Text = "LiveView";
                    _capture.Pause();
                    imageBox2.Hide();
                }
                else
                {
                    try
                    {
                        if (_capture == null)
                        {
                            _capture = new VideoCapture(streamAddress);
                        }
                        View_1.Text = "Stop LiveView";
                        _capture.ImageGrabbed += ProcessFrame;
                        _capture.Start();
                        imageBox2.Show();
                    }
                    catch (NullReferenceException excpt)
                    {
                    }
                }
                _captureInProgress = !_captureInProgress;
            }
            else
            {
                MessageBox.Show("Stream address is missing");
            }
        }
```

Ohjelmakoodi 2. Livekuvan näyttäminen käyttöliittymässä.

```
private void ProcessFrame(object sender, EventArgs arg)
        {
            UMat frame = new UMat();
            _capture.Retrieve(frame, 0);
            imageBox2.Image = frame;
            frame.Dispose();

        }
```

Näillä toiminnoilla kameran livekuva saatiin näkymään sovelluksen pääikkunassa, sille varatussa tilassa (Kuva 9). Sovelluksen käyttöliittymään lisättiin vielä nappi, josta livekuvan näyttämisen voi aloittaa tai lopettaa.

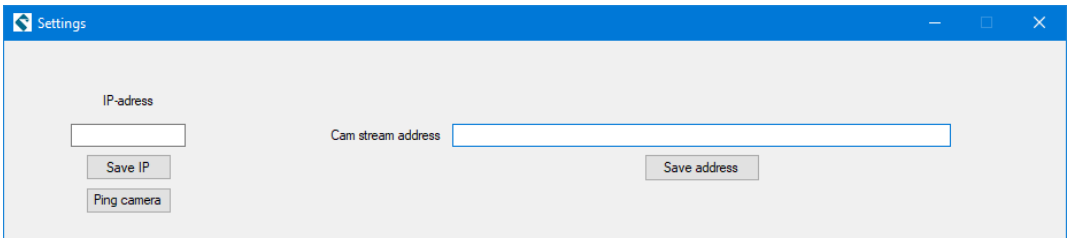
Kuva 9. Kameran livekuva ohjelman pääikkunassa (Kuvakaappaus)

8.4 Asetusikkuna ja kirjautuminen

Koska sovelluksessa täytyisi pystyä tekemään erilaisia asetuksia käyttöönoton yhteydessä, luotiin seuraavaksi uusi ikkuna asetusvalintoja varten. Ikkunaan pääsemiseksi tehtiin vielä kirjautuminen ja MD5-salausta käyttäen luotiin kovakoodatut tunnukset, jotka tallennettiin sovelluksen settings-tauluun. Käyttäjän kirjautuessa luodaan hänen syöttämistään kirjautumistiedoista MD5-tiiviste, jota verrataan tallennettuna olevaan tiivisteesen. Mikäli tunnukset täsmäävät, aukeaa asetusikkuna käyttäjän nähtäville.

Asetusikkunaan lisättiin tekstikenttä kameran stream-osoitteen tallennusta varten, osoite tallennetaan sovelluksen settings-tauluun. Mahdollisten ongelmien diagnosointia varten lisättiin tekstikenttä kameran IP-osoitteelle ja painike, jolla kutsutaan metodia, joka lähettää ICMP-paketin, eli niin sanotun pingin tallennettuun IP-osoitteeseen (Kuva 10). Mikäli kamera vastaa pingiin, näytetään viesti-ikkunassa asian ilmaiseva viesti ja samoin tehdään mikäli kamera ei vastaa.

Kuva 10. Kameran asetusvalikko alkuvaiheessa (Kuvakaappaus)



8.5 Videokuvan puskurointi

Kuvan puskurointia varten kirjoitettiin ohjelmaan metodi, joka lukee kameran streamia, kuten tehtiin aiemmin livekuvan näyttämistä varten. Tätä varten tehtiin käyttöliittymään aluksi painike, jolla puskurointi käynnistetään.

Tässä tapauksessa frameja tallennetaan listalle, jonka koon määritystä varten lisättiin asetusikkunaan tekstikenttä ja tallennuspainike. Mikäli puskurin kooksi yritetään asettaa

enemmän, kuin 375 framea pakotetaan ohjelmassa koko 375 frameen, jottei tietokoneen muistiresursseja päästä käyttämään liiallisesti ohjelman käyttötarkoitukseen nähden.

Mikäli tallennuksen alkaessa puskuri on tyhjä, täytetään se ensin ja siirrytään ohjelmassa seuraavaan vaiheeseen, jossa listan viimeiselle paikalle kirjoitetaan aina uusi frame ja ensimmäiseltä paikalta vastaavasti poistetaan. Tämä ei varmastikaan ole paras tapa tehdä puskurointia, vaan oikeampi tapa olisi tehdä rengaspuskuri. Koska EmguCV mahdollistaa framejen lukemisen yksittäin, voidaan puskurointi kuitenkin toteuttaa tälläkin tavalla (Ohjelmakoodi 3).

Ohjelmakoodi 3. Kuvan puskurointi.

```
        private void ProcessFrameRec(object sender, EventArgs arg)
        {
            if (_captureInProgressRec == true)
            {
                if (framelist.Count < (bufferSize - 1))
                {
                    using (UMat framerec1 = new UMat())
                    {
                        _captureRec.Retrieve(framerec1, 0);
                        do
                        {
                            _captureRec.Read(framerec1);
                            long frameSize = framerec1.Size.Height *
framerec1.Size.Width * 4;
                            GC.AddMemoryPressure(frameSize);
                            framelist.Add(framerec1.Clone());
                            framerec1.Clone().Dispose();
                            GC.RemoveMemoryPressure(frameSize);
                        } while (framelist.Count < bufferSize);
                    }
                }
                if (framelist.Count >= (bufferSize - 1))
                {
                    using (UMat framerec1 = new UMat())
                    {
                        _captureRec.Retrieve(framerec1, 0);
                        do
                        {
                            _captureRec.Read(framerec1);
                            long frameSize = framerec1.Size.Height *
framerec1.Size.Width * 4;
                            GC.AddMemoryPressure(frameSize);
                            framelist.Insert(bufferSize, framerec1.Clone());
                            framelist.RemoveAt(0);
                            framerec1.Clone().Dispose();
                            GC.RemoveMemoryPressure(frameSize);
                        } while(_captureInProgressRec);
                    }
                }
            }
        }
```

8.6 Videon tallennus puskurista

Kun sovellukselle lähetetään triggaus automaatiojärjestelmästä, video täytyy saada tallennettua tietokoneen tallennustilaan. Tämän ominaisuuden kehittämistä varten tehtiin käyttöliittymään ensin painike, jolla triggausta voidaan simuloida kutsumalla tallennuksen suorittavaa aliohjelmaa.

Tallennukseen käytetään VideoWriter-luokkaa, jolle annetaan tiedoiksi tallennuspolku, tallennettavan tiedoston nimi ja tiedostopääte, käytettävä koodekki, videon FPS-lukema, resoluutio ja vielä lopuksi tieto, onko tallennettava video värikuvaa vai ei (Ohjelmakoodi 4).

VideoWriterille kerrotaan mitä koodekkia sen tulee käyttää videon pakkaamiseen FourCC-koodilla (Four Character Code). Videon pakkaamiseen käytettiin tässä projektissa H.264-koodekkia, jonka FourCC-koodi on H264. H264 on tällä hetkellä yksi käytetyimmistä koodekeista. (Butler, 2020)

Tällä pakkaustavalla ja aiemmin asetetuilla kameran kuva-asetuksilla kuuden sekunnin videon tiedostokooksi saatiin noin 300 Kt, kun sen raakaversion koko ilman pakkausta oli noin 8 Mt. Videon tiedostomuotona käytetään mp4-formaattia.

Ohjelmakoodi 4. Videon tallentaminen puskurista tietokoneen tallennustilaan:

```
        public void SaveVideo(bool callerIsUdp)
         {
             bool recWas = false;
             int fps = 20;
             if (framelist.Count > 0)
             {
                 if (_captureInProgressRec)
                 {
                     recWas = true;
                     _captureInProgressRec = false;
                 }
                 width =
Convert.ToInt32(_captureRec.Get(Emgu.CV.CvEnum.CapProp.FrameWidth));
                 height =
Convert.ToInt32(_captureRec.Get(Emgu.CV.CvEnum.CapProp.FrameHeight));
                 fourcc = VideoWriter.Fourcc('H', '2', '6', '4');
                 VideoWriter vw1 = new VideoWriter(path +
DateTime.Now.ToString() + ".mp4", fourcc, fps, new Size(width, height), true);
                 for (int ii = 0; ii < framelist.Count; ii++)
                 {
                     vw1.Write(framelist[ii]);
                 }
                 vw1.Dispose();
                 framelist.RemoveRange(0, framelist.Count);
                 if (callerIsUdp && Settings1.Default.autosend)
                 {
                     callAzureClient();
                 }
                 if (recWas == true)
                 {
                     _captureInProgressRec = true;
                 }
             }
             else
             {
                 MessageBox.Show("Buffer is empty, nothing to save.");
             }
         }
```

8.7 Triggausmahdollisuuden ohjelmointi sovellukseen

Triggaamiseen päätettiin käyttää UDP-sanomaa, koska se on kevyt toteuttaa ja helposti järjestettävissä automaatiojärjestelmästä. Tätä varten sovellukseen täytyi rakentaa ominaisuus viestien vastaanottamista varten. UdpClient-luokkaa hyödyntämällä tehtiin aliohjelma UdpListener (Ohjelmakoodi 5), joka kuuntelee UDP-porttia 15001. Mikäli ohjelma saa sanoman ”record”, se kutsuu aiemmin luotua aliohjelmaa, joka tallentaa videon puskurista tietokoneen tallennustilaan.

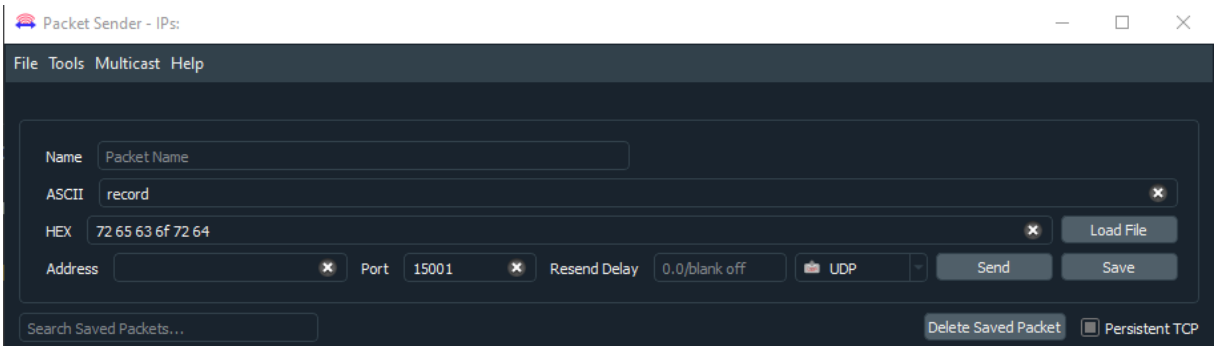
Koska UdpListener-ohjelmassa on jatkuvasti pyörivä while-silmukka, täytyi ohjelma asettaa käynnistymään niin sanotussa child threadissa eli lapsisäikeessä, jolloin sitä ajetaan omassa säikeessään ja pääohjelmaa ajetaan sovelluksen main threadissa, eli vanhempisäikeessä. Ohjelma asetettettiin käynnistymään, kun pääohjelma käynnistyy ja lapsisäie määriteltiin taustalla toimivaksi. Tällöin sen toiminta lopetetaan, jos vanhempisäikeen toiminta loppuu. Mikäli näin ei tehtäisi, jäisi lapsisäie käyntiin taustaprosessina vaikka pääohjelma sammuu, koska ikuinen while-silmukka ei loppuisi koskaan.

Ohjelmakoodi 5. UDP-porttia kuunteleva aliohjelma.

```
        private void UdpListener()
        {
            int listenPort = 15001;
            UdpClient listener = new UdpClient(listenPort);
            IPEndPoint groupEP = new IPEndPoint(IPAddress.Any, listenPort);
            try
            {
                while (true)
                {
                    byte[] bytes = listener.Receive(ref groupEP);
                    if (Encoding.ASCII.GetString(bytes, 0, bytes.Length) ==
"record")
                    {
                        SaveVideo(true);
                    }
                }
            }
            catch (SocketException e)
            {
                MessageBox.Show(e.ToString());
            }
            finally
            {
                listener.Close();
            }
        }
```

UDP-sanomalla triggeröinnin toimivuutta testattiin internetistä ilmaiseksi ladattavalla Packet Sender-työpöytäohjelmalla (Kuva 11), jolla voidaan lähettää TCP-,UDP- ja SSL-sanomia. Ohjelmassa voidaan määritellä muun muassa kohdelaitteen IP-osoite, portin numero ja sanoman sisältö. UDP-sanomalla triggeröinti toimi odotetusti toiselta tietokoneelta sanomia lähetettäessä ja lisäksi tehtiin onnistunut testaus mobiililaitteella.

Kuva 11. Packet Sender ohjelman näkymä (Kuvakaappaus)

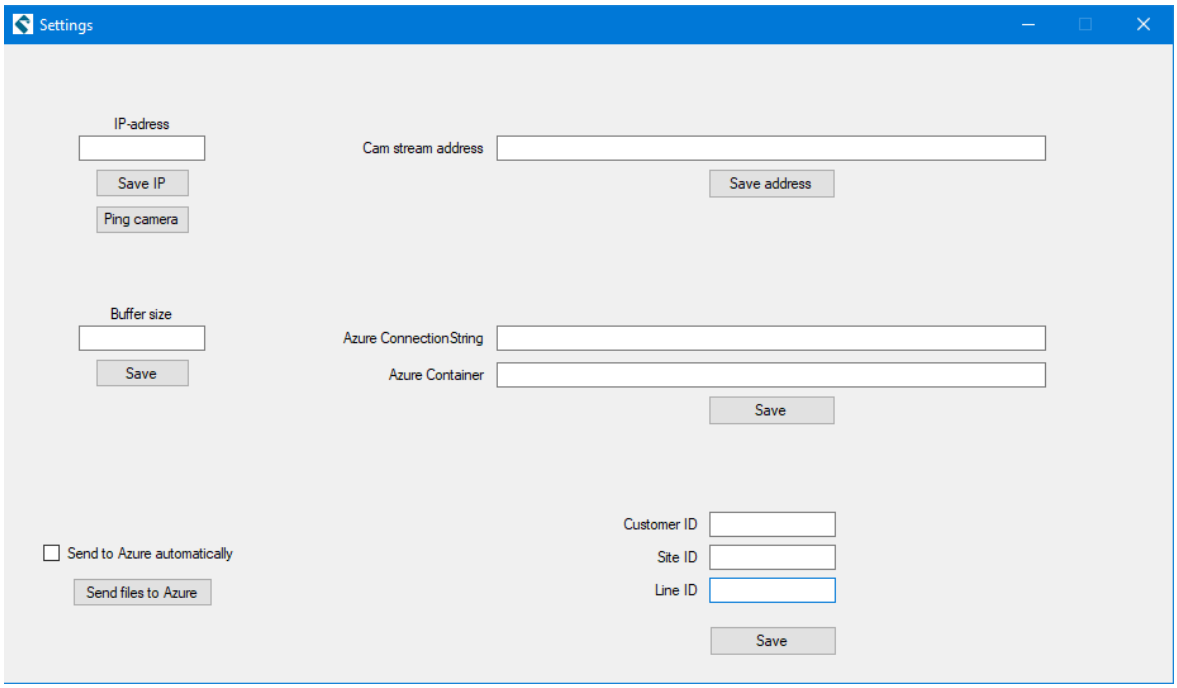


8.8 Videon lähetyksen mahdollistaminen Azureen

Videotiedostojen lähetystä varten käytettiin BlobContainerClient-luokkaa, joka mahdollistaa Azure Storagen containereiden ja blobien ominaisuuksien asettamisen, muokkaamisen ja manipuloinnin. Microsoft, n.d.-c)

Jotta storageen saadaan yhteys, tarvitaan sitä varten avain tai Connection String, tässä työssä käytettiin jälkimmäistä ja sen tallentamista varten luotiin sovelluksen asetusikkunaan jälleen tarvittavat ominaisuudet. Lisäksi tehtiin samalla kertaa myös tallennusmahdollisuus kohteena olevan containerin nimen tallennukseen sekä kolmelle eri metadata-parille. Metadan avaimiksi määriteltiin Customer ID, Site ID ja Line ID, joiden arvojen perusteella voidaan eritellä ja ohjailla dataa asiakaskohtaisesti pilvipalvelussa. Lisäksi luotiin painike, jolla tiedostot lähettävää aliohjelmaa voitaisiin kutsua manuaalisesti sekä asetusvalinta jolla voidaan päättää lähetetäänkö videotiedostot Azureen automaattisesti, vai jätetäänkö ne tietokoneen tallennustilaan (Kuva 12).

Kuva 12. Sovelluksen asetusikkuna valmiina (Kuvakaappaus)



Lähetysohjelma asetettiin käymään lähdekansiossa olevat tiedostot läpi ja lähettämään sen jälkeen yksitellen tiedostot pilvipalveluun metadan kanssa, onnistuneen lähetyksen jälkeen tiedosto poistetaan (Ohjelmakoodi 6). Mikäli lähetys ei onnistu, yrittää ohjelma sitä seuraavan kerran kutsuttaessa lähettää kansiossa olevat tiedostot uudestaan. Lähetysohjelmaa kutsutaan SaveVideo-ohjelmasta videon tallennuksen jälkeen, mikäli SaveVideo-ohjelmaa on kutsuttu UdpListener-ohjelmasta. Lisäksi kutsu voidaan tehdä asetusvalikkoon aiemmin luodun painikkeen avulla.

Ohjelmakoodi 6. Tiedostojen lähettäminen Azureen.

```
BlobHttpHeaders blobHttpHeaders = new BlobHttpHeaders();
blobHttpHeaders.ContentType = "video/mp4";
var containerClient = new BlobContainerClient(connectionString, container);
foreach (var file in files)
{
     try
     {
          var blobClient = containerClient.GetBlobClient(file.Name);
          using (var fileStream = File.OpenRead(file.FullName))
          {
               IDictionary<string, string> metadata = new
Dictionary<string,string>();
               await blobClient.UploadAsync(fileStream);
               blobClient.SetHttpHeaders(blobHttpHeaders);
               metadata.Add("CustomerID", Settings1.Default.customerID_mem);
               metadata.Add("SiteID", Settings1.Default.SiteID_mem);
               metadata.Add("LineID", Settings1.Default.LineID_mem);
               blobClient.SetMetadata(metadata);
```

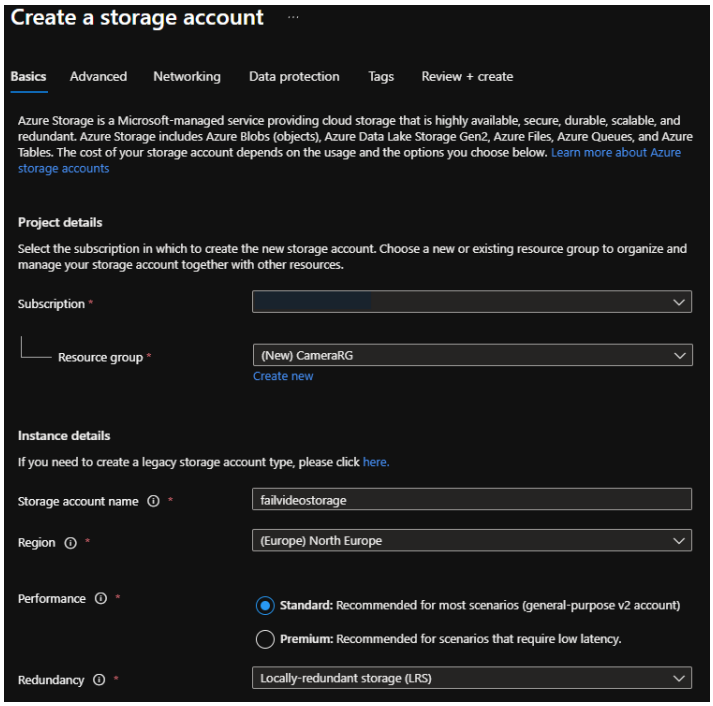
```
            }
            File.Delete(file.FullName);
        }
        catch (Exception ex)
        {
        }
}
```

9 Videotiedostojen lähetys Azureen

Tässä luvussa kuvataan tallennustilan perustamisprosessi Azure-ympäristössä jo olemassa olevaan tilaukseen.

Azuren hallintaportaalista lähdettiin luomaan uutta resurssia, Storage Accountia (Kuva 13). Samassa yhteydessä luotiin uusi Resource group, jolloin muut aiheeseen liittyvät ja mahdollisesti myöhemmin lisättävät palvelut voidaan liittää samaan ryhmään. Sijainniksi valittiin Pohjois-Eurooppa ja replikoinniksi otettiin halvin vaihtoehto, jossa varastossa oleva data kopioidaan kolme kertaa saman datakeskuksen sisälle. Kun tallennustila oli provisioitu ja valmis käytettäväksi, tehtiin sen sisälle vielä uusi, kansion omainen Container, jolle annettiin nimeksi Videos. Lopuksi käytiin vielä kopioimassa Connection string tallennustilan Access Keys-valikon alta, tämä toimii avaimena jolla kamerasovellus autentikoituu tallennustilaan ja pystyy tämän ja containerin nimen perusteella lähettämään videotiedostot pilvipalveluun.

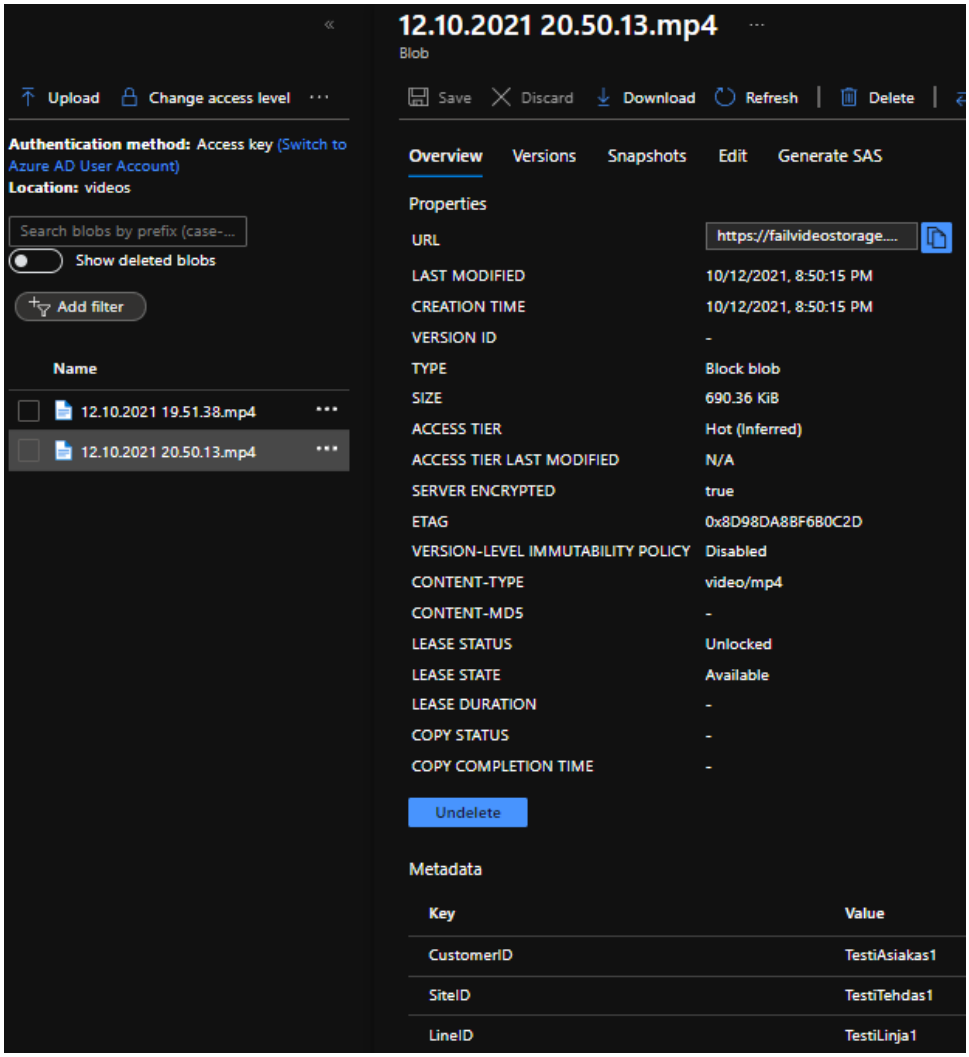
Kuva 13. Valinnat Storage accountia luotaessa (Kuvakaappaus)



Connection string ja containerin nimi syötettiin lopuksi kamerasovellukseen, niitä varten aiemmin luotuihin kenttiin asetusvalikossa ja testattiin videoiden lähetystä pilvipalveluun

manuaalisesti. Lisäksi testattiin vielä videon tallennuksen ja automaattisen lähetyksen toimivuus UDP-sanomalla triggausta käyttäen. Videotiedostot saapuivat containeriin ja myös metadatan lähetys toimi toivotulla tavalla (Kuva 14), videotiedostoille voitiin nyt luoda jaettavia linkkejä, joiden kautta videota voi katsoa suoraan selaimessa. Linkeille on asetettavissa myös voimassaoloaika. Videoita voi myös ladata Azuren portaalin kautta omalle työasemalle katsottavaksi.

Kuva 14. Azureen lähetetyn videotiedoston ominaisuudet ja metadata tarkasteltuna Azuren portaalin kautta (Kuvakaappaus)



10 Kamerajärjestelmän asennus testikohteeseen

Automaatiojärjestelmältä virhetilanteen yhteydessä lähtevän triggauksen toteuttajaksi valittiin tämän työn yhteydessä Kawasaki-hitsausrobotti. Hitsausrobotin työkalussa on törmäyksentunnistusta varten anturi, mikäli anturin robotin ohjaimelle välittämä signaali katkeaa, tulkitaan tämä törmäykseksi ja robotin liike ja linjan toiminta pysähtyy. Robotti antaa tästä hälytyksen omassa käsiohjaimessaan ja lisäksi virheestä hälytetään linjan käyttöliittymässä. Robotin ohjelmistoon lisättiin aliohjelma, joka suorittaa määritellyn UDP-sanoman lähetyksen haluttuun IP-osoitteeseen ja porttiin samassa lähiverkossa. Aliohjelmaa voidaan kutsua robotin ohjelmistosta sen mukaan kuin nähdään tarpeelliseksi. Tällä hetkellä triggauksen tekevää ohjelmaa kutsutaan aina, kun robotti menee sisäiseen häiriöön. Tällöin robotti itsessään on jonkinlaisessa häiriössä ja kyseessä on joko törmäys, vika hitsauslaitteiston virtalähteessä tai mahdollisesti mekaaninen tai elektroninen vika itse robotissa tai sen ohjaimessa.

Kamera, sille virtaa syöttävä PoE-injektori, sekä kameraohjelmiston ajamisesta vastaava PC siirrettiin Orferin hitsaussoluun. PC asennettiin linjan sähkökeskukseen ja kytkettiin solun ethernet-verkkoon. Kamerasovellus asetettiin käynnistymään Windowsin käynnistyksen yhteydessä automaattisesti mahdollisten virtakatkosten varalta. Kameralle valittiin paikka, josta se saisi mahdollisimman hyvin kuvaa robotin toiminnasta, kamera kytkettiin myös solun verkkoon.

Kamerajärjestelmän toimintaa testattiin muun muassa ajamalla robotin hitsauskolvi käsiajolla kiinteää estettä vasten siten, että törmäyksen tunnistava anturi juuri ja juuri aktivoituu. Triggeröinti toimi ja PC:n puskuroima video tallentui koneen tallennustilaan, jonka jälkeen se lähetettiin automaattisesti Azuren storage containeriin.

Järjestelmää seurattiin vielä muutamina päivinä etäyhteyden välityksellä, jotta sen toimintaa mahdollisesti haittaavat virheet tulisivat ilmi. Järjestelmä on tähän mennessä kuitenkin toiminut ilman virheitä ja siitä kerätään parhaillaan käyttökokemuksia.

11 Yhteenveto

Opinnäytetyön tarkoituksena oli toteuttaa ohjelmisto teollisuusautomaatiojärjestelmien virhetilanteiden videomuotoista taltiointia varten. Järjestelmän oli myös kyettävä lähettämään tallennetut videotiedostot automaattisesti pilvipalveluun. Järjestelmän on mahdollisesti tulevaisuudessa tarkoitus palvella tilaajana toimivan Orfer Oy:n asiakkailleen tarjoaman tukipalvelun vianselvitystyökaluna.

Työssä perehdyttiin IP-kameroihin, Microsoft Azuren tallennusratkaisuihin ja itse järjestelmän toteutukseen ja ohjelmointiin. Työn edetessä huomasin, että erilaisia aihealueeseen liittyviä ohjelmakirjastoja ja myös toteutustapoja on lukuisia ja niiden vertailuun olisi voinut käyttää enemmän aikaa. Esimerkiksi valmista rengaspuskuria tarjoava kirjasto olisi voinut olla tämän työn yhteydessä parempi vaihtoehto käytetyn kirjaston sijaan.

Työn aikana sain runsaasti täysin uutta tietoa kamera- ja kuvankäsittelytekniikasta sekä datan ohjelmallisesta lähettämisestä Microsoft Azureen. Lisäksi sain arvokasta kokemusta koodin optimoinnista, sillä tulevien kohdejärjestelmien rautapuolen resurssit täytyi ottaa myös huomioon. Uskon, että näistä keräämistäni tiedoista ja taidoista on minulle runsaasti hyötyä myös tulevaisuudessa.

Työ onnistui mielestäni hyvin ja etenkin toteutusvaihe eteni vauhdilla. Kirjallisen osuuden valmiiksi saattaminen venyi hieman pidemmälle kuin olin suunnitellut, mutta tämän johdosta sain miettiä sisältöä hieman tarkemmin.

Lähteet

Ajo, R., Hakonen, S., Harju, H., Järvi, J., Kaskes, K., Lenardic, E., Niukkanen, E., Nurminen, T., Ritala, P., Tolppanen, M. & Tommila, T. (n.d.) *Laatu automaatiossa*, s. 120. Haettu 16.10 2021 osoitteesta https://www.automaatioseura.fi/site/assets/files/1426/laatuautomaatiossa.pdf

Butler, S. (2020) The Most Common Video Formats and Codecs Explained. Haettu 6.10.2021 osoitteesta https://helpdeskgeek.com/windows-xp-tips/the-most-common-video-formats-and-codecs-explained/

Emgu. (n.d.) Emgu wiki. Haettu 3.10.2021 osoitteesta https://www.emgu.com/wiki/index.php/Main_Page

Holloway Houston. (2019) Manual vs. automated material handling: know the difference. Haettu 10.10.2021 osoitteesta https://www.hhilifting.com/manual-vs-automated-material-handling

IFR. (2020) Haettu 9.11.2021 osoitteesta https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe

Itewiki. (n.d.) Microsoft Azure. Haettu 2.10.2021 osoitteesta https://www.itewiki.fi/opas/microsoft-azure/#:~:text=Microsoft%20Azure%20on%20Microsoftin%20pilvipalvelu,yhdist%C3%A4%C3%A4%20osaksi%20paikallista%20IT%2Dinfrastruktuuria

Kawasaki. (n.d.-a) Haettu 9.11.2021 osoitteesta https://robotics.kawasaki.com/en1/products/robots/spot-welding/BX200L/

Kawasaki. (n.d.-b) Haettu 9.11.2021 osoitteesta https://robotics.kawasaki.com/en1/products/robots/palletizing/CP180L/

Kawasaki. (n.d.-c) Haettu 9.11.2021 osoitteesta
https://robotics.kawasaki.com/en1/products/robots/pick-place/YF003N/

Koskinen, K. (2017) Automaatio – mistä se on tullut? *Automaatioväylä* 5/2017, s. 8-11. Haettu 6.10.2021 osoitteesta
https://www.automaatioseura.fi/site/assets/files/1380/automaatio_ennen_nyt_ja_tulevaisuudessa_av_artikkelisarja_2018.pdf

Lehtinen, H. (n.d.) Robotit. *Automaatioseura* Haettu 10.10.2021 osoitteesta
http://automaatioseura.planeetta.com/index/tiedostot/Robotit.pdf

Microsoft. (n.d.-a) What are the different types of cloud computing services? Haettu 3.10.2021 osoitteesta
https://azure.microsoft.com/en-us/overview/types-of-cloud-computing/

Microsoft. (n.d.-b) What is PaaS? Haettu 3.10.2021 osoitteesta
https://azure.microsoft.com/en-us/overview/what-is-paas/

Microsoft (n.d.-c) BlobContainerClient Class. Haettu 4.10.2021 osoitteesta
https://docs.microsoft.com/en-us/dotnet/api/azure.storage.blobs.blobcontainerclient?view=azure-dotnet

Microsoft. (2021) About properties and metadata. Haettu 4.10.2021 osoitteesta
https://docs.microsoft.com/en-us/azure/storage/blobs/storage-blob-properties-metadata?tabs=dotnet

OpenCV. (n.d.-a). About. Haettu 3.10.2021 osoitteesta
https://opencv.org/about/

Orfer. (n.d.-a) Liiketoiminta. Haettu 2.10.2021 osoitteesta
https://www.orfer.fi/suomeksi/ORFER/LIIKETOIMINTA/tabid/12874/language/en-US/Default.aspx

Orfer. (n.d.-b) Lavausjärjestelmä. Haettu 6.10.2021 osoitteesta https://www.orfer.fi/suomeksi/TUOTTEET/LAVAUS/tabid/12715/language/en-US/Default.aspx

Orfer. (n.d.-c) Orferin historia. Haettu 6.10.2021 osoitteesta https://www.orfer.fi/suomeksi/ORFER/HISTORIA/tabid/12923/language/en-US/Default.aspx

Robot hall of fame. (n.d.) Unimate. Haettu 9.11.2021 osoitteesta http://www.robothalloffame.org/inductees/03inductees/unimate.html

Shake, M. (n.d.) Haettu 9.11.2021 osoitteesta https://www.jabil.com/blog/ten-popular-industrial-robot-applications.html

Shibaura-Machine. (n.d.) Haettu 9.11.2021 osoitteesta https://www.shibaura-machine.co.jp/en/product/robot/lineup/th/THE600.html

Snaplogic. (n.d.) Azure Blob Storage. Haettu 5.10.2021 osoitteesta https://www.snaplogic.com/glossary/azure-blob-storage

Sourceforge. (n.d.) OpenCV. Haettu 5.10.2021 osoitteesta https://sourceforge.net/projects/opencvlibrary/files/stats/timeline?dates=2001-09-20+to+2021-10-05

Swann. (2018) NVR vs. DVR. Haettu 3.10.2021 osoitteesta https://www.swann.com/blog/dvr-vs-nvr-whats-the-difference/

Tilavahti.(n.d.) Mikä ihmeen IP-kamera? Haettu 5.10.2021 osoitteesta https://www.tilavahti.com/page/10/mika-on-ip-kam

Tutorialspoint. (n.d.) OpenCV. Haettu 4.10.2021 osoitteesta https://www.tutorialspoint.com/opencv/opencv_storing_images.htm

Valmistajat. (n.d.) Automaatio ja automaatiojärjestelmät. Haettu 5.10.2021

osoitteesta https://valmistajat.fi/menetelmat/elektroniikka/automaatio-ja-automaatiojarjestelmat

Liite 1: Aineistonhallintasuunnitelma

Kehitettävän ohjelmiston lähdekoodia säilytetään työn tekemisen ajan opinnäytetyön tekijän tietokoneen D:-asemalla ja oppilaitoksen tarjoamassa OneDrive-palvelussa, lisäksi lähdekoodia säilytetään työn tekemisen ajan tilaajan tarjoamassa OneDrive-palvelussa. Valmis lähdekoodi tallennetaan myös tilaajan versionhallintajärjestelmään.