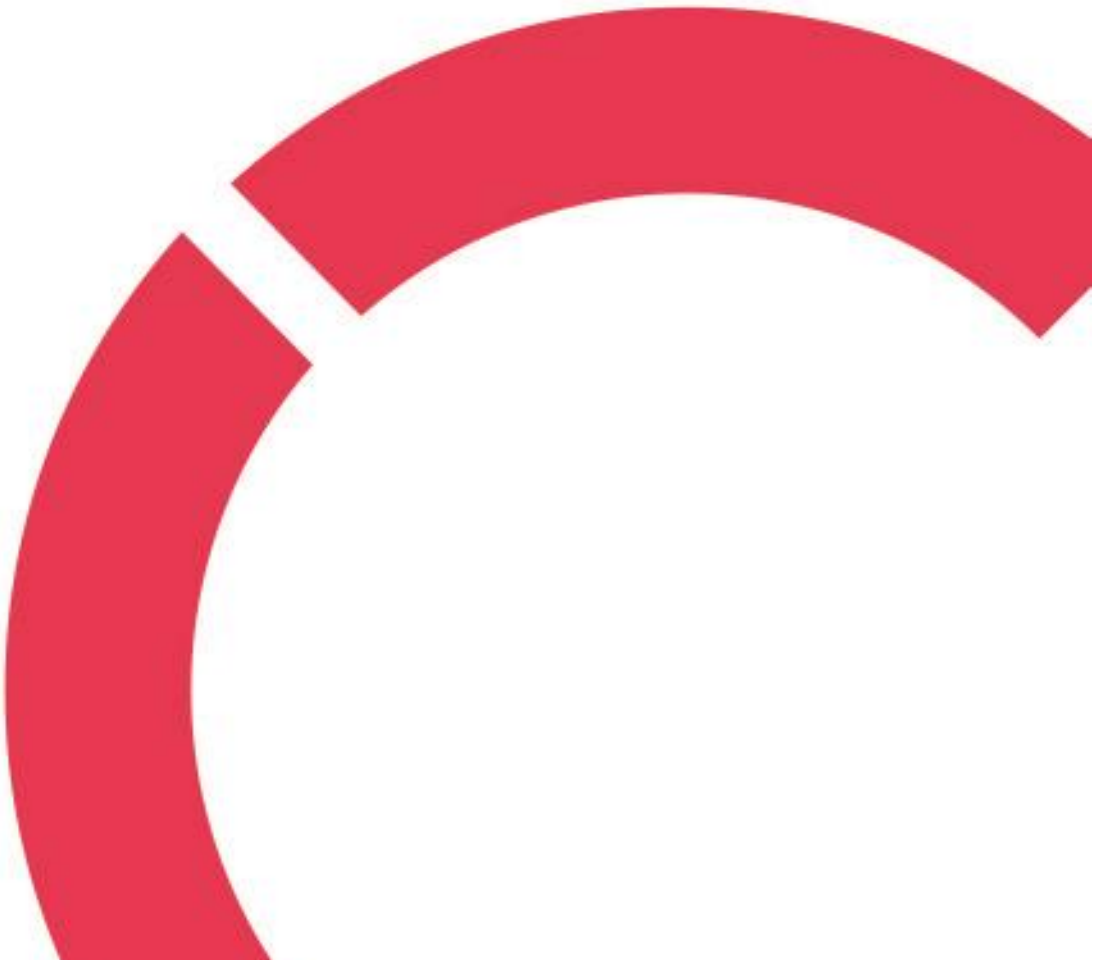
Toni Ngô-Lundell

AUTOMAATION TOTEUTTAMINEN MATERIAALIMIKROSKOOPIN KUVAUSPROSESSEILLE

Opinnäytetyö
CENTRIA-AMMATTIKORKEAKOULU
Tieto- ja viestintätekniikan koulutus
Toukokuu 2024



TIIVISTELMÄ OPINNÄYTETYÖSTÄ

Centria-ammattikorkeakoulu	**Aika** Toukokuu 2024	**Tekijä/tekijät** Toni Ngô-Lundell
Koulutus Tieto- ja viestintätekniikka		☒ AMK ☐ YAMK
Työn nimi AUTOMAATION TOTEUTTAMINEN MATERIAALIMIKROSKOOPIN KUVAUSPROSESSEILLE		
Työn ohjaaja Ari Lamberg		**Sivumäärä** 31 + 2
Työelämäohjaaja Tommi Palosaari		

Tämä teksti kuvaa projektia, jossa Sasken Finland Oy antoi toimeksiannon käsikäyttöisen, kameallisen mikroskooppinsa kuvausprosessien automatisoimiseksi. Yritys halusi mikroskoopin kuvaavan kuvattavia kappaleita isoina ruudukkoina, jotta kuvat voitaisiin myöhemmin yhdistää yhdeksi suureksi kuvaksi.

Projektin laitetarpeet selvitettiin ja sopivimmaksi todettu motorisoitu kuvaustaso tilattiin. Kuvaustason ohjaus toteutettiin LabVIEW-ohjelmointikielellä. Mikroskoopin kameran integrointi toteutettiin erillisellä laukaisupiirillä, joka tehtiin käsin projektia varten.

Projekti oli onnistunut ja lopputulos saavutti kaikki asiakasyrityksen määritelmät ja toiveet. Laitekokonaisuus ohjelmistoineen otettiin käyttöön yrityksessä.

Asiasanat
Automaatio, elektroniikka, LabVIEW, mikroskopia, ohjelmointi, valokuvaus

ABSTRACT

Centria University of Applied Sciences	**Date** May 2024	**Author** Toni Ngô-Lundell
Degree programme Information Technology		
Name of thesis AUTOMATING THE IMAGING PROCESSES OF A MATERIALS MICROSCOPE		
Centria supervisor Ari Lamberg		**Pages** 31 + 2
Instructor representing commissioning institution or company Tommi Palosaari		

This thesis report is a description of a project, in which Sasken Finland Oy commissioned the automation of the imaging processes of their manually controlled microscope with an integrated camera. The company intended for the microscope to image specimens in large grids, which could later be stitched into a single large image.

The necessary hardware was determined and ordered. The software to control this hardware was developed in LabVIEW. Interface with the microscope's camera was handled with a separate triggering circuit, which was handmade for the project.

The project was successful, and the resulting solution fulfilled all the customer's definitions and wishes. The hardware assembly and software were implemented as part of their imaging process.

Key words
Automation, electronics, LabVIEW, microscopy, photography, programming

KÄSITTEIDEN MÄÄRITTELY

COM-port

COM-portti on tietokoneista löydettävä tietoliikenneliitäntä, joka voi olla fyysinen liitäntäpiste laitteen rungossa, tai emuloitu yhteyspiste esimerkiksi langattomissa ratkaisuissa.

I/O Trigger

Input/Output Trigger on laitteista löydettävä liitäntä, joiden tarkoitus on vastaanottaa tai lähettää liipaisupulsseja muilta laitteilta tai laitteille.

1 JOHDANTO

Tämä teksti toimii raporttina kesällä 2023 alkaneesta projektista, jossa Sasken Finland Oy palkkasi minut tehostamaan toimintaansa. Yritys omisti kamerallisen Leica DM1750 -materiaalimikroskoopin. Yksi mikroskoopin käyttötarkoituksista oli erilaisten kappaleiden kuvaaminen pala palalta, joista sitten kuvankäsittelyohjelmassa luotiin ruudukko ja yhdistettiin kuvat yhdeksi suureksi kuvaksi. Yrityksen toiveena ja tämän projektin tavoitteena oli automatisoida yrityksen mikroskoopin käyttö tilanteissa, joissa haluttiin luoda näitä suuria kokonaiskuvan havainnollistavia kuvayhdistelmiä. Aiemmin näiden kollaasien luominen oli edellyttänyt jokaisen kuvan kohdistamista ja kameran laukaisemista käsin, ja lähdekuvia yhteen kuvayhdistelmään saattoi mikroskoopin suurennuksesta riippuen tarvita satoja.

Yrityksen tavoitteena oli kuvausmenetelmää automatisoimalla säästää insinööriensä arvokkaita työtunteja ja säästää heitä hitaalta ja puuduttavalta työltä, jossa heidän erityisosaamistaan ei tarvittu ja jota ei voitu kokea erityisen mielekkääksi. Hyväksyttyäni projektin tapasin yrityksen edustajia määrittelypalaverissa. Pääsin tutustumaan mikroskooppiin sekä sen ympäristöön kuvauslaboratoriossa. Yrityksellä ei ollut entuudestaan laboratoriossa minkäänlaisia laitteita automaation toteuttamiseen, joten laitteiden tarpeen kartoittaminen ja hankkiminen tulisi olemaan osa projektia. Lähtöasetelmassa mikroskoopin työasemalla oli ainoastaan mikroskooppi, siihen liitetty kamera, sekä tietokone, jolla ajettiin mikroskoopin valmistajan Leican omaa ohjelmistoa.

Ohjelmistossa ei ollut automaation tarvittavia toiminnallisuuksia, mutta itse tietokonetta voitaisiin käyttää automaation ohjaamiseen. Mikroskoopin omaa näytetasoa ei myöskään ollut käytännöllistä täysin motorisoida, joten se tulisi poistaa, jotta motorisoitu taso mahtuisi tilalle. Tämä tulisi myös edellyttämään, että pelkkä automaation toteuttaminen ei olisi riittävää; tulevalle motorisoidulle tasolle olisi välttämätöntä toteuttaa myös jonkinlainen suora käsiohjaus, jotta itse mikroskooppi pysyisi käytettävissä muuhun, kuin projektin aiheena olevaan automatisoituun kuvausohjelmaan.

Mikroskooppiin kiinnitetyn Leica Flexacam C1 -kameran hyödyntämisessä tulisi olemaan myös omat haasteensa. Normaalisti kameraa ohjattiin Leican tietokoneohjelmalla, mutta se ei tulisi olemaan tässä projektissa toimiva vaihtoehto. Kameran asetukset voitiin kuitenkin määritellä ilman ohjelmistoa, ja kameran laukaisua varten kamerassa oli yhteensopivuus käsikytkintä varten. Käsikytkimen käyttö ei tietenkään olisi mahdollista, mutta lähdimme liikkeelle oletuksella, että voisimme laukaista kameran

ohjauspulssilla käyttäen käsikytkimelle tarkoitettua liitäntää kameran takapaneelissa. Tämä tarve ohjauspulssille tulisi lopulta myös vaikuttamaan huomattavasti automaation toteuttamiseksi hankittavien laitteiden valintaan. Kameran laukaisua tarkemmin valmistellessa tuli myös ilmi, että motorisoidun tason kontrollerilaitteen ja kameran välille tultaisiin tarvitsemaan jonkinlainen erillinen laukaisupiiri. Tämän lisälaitteen suunnittelussa ja toteutuksessa oli omat haasteensa, mutta elektroniikkaosaamisen virkistäminen ja kehittäminen oli itseni kannalta tervetullut lisä projektin sisältöön.

Laitetarpeita selvitettäessä suunnitteluvaiheen aikana alkoi myös hahmottua, millä kielellä laitteiden ohjaukseen tarvittava ohjelma tultaisiin luomaan. Kieleksi valikoitui LabVIEW, jota korostettiin käytännössä kaikkien projektiin soveltuvien motorisoitujen tasojen esittelytiedoissa yhteensopivaksi, mukaan lukien lopulta hankittavaksi päätetty laite. Yritykseltä myös löytyi entuudestaan vahvaa pitkän linjan LabVIEW-osaamista, joten ohjelmointityön haasteisiin tulisi olemaan saatavilla tukea heidän osaltaan. Tämä myös tarkoitti, että yrityksellä oli jo entuudestaan käytettävissään LabVIEW-lisenssejä, joita voitiin projektini käyttöön resursoida ilman erillistä hankintatarvetta. LabVIEW itsessään toisi projektiin myös uusia haasteita, sillä kieli ei ollut minulle entuudestaan tuttu. Tämän kautta projektiin saatiin mukaan ylimääräistä uuden oppimista.

Projektin aikana korostui jatkuvasti teollisiin menetelmiin liittyvä yrityssalaisuuksien pito, joka vaikeutti suunnattomasti minkäänlaisien esimerkkien löytämistä työn pohjaksi. Lähes kaikki projektiin liittyvä oli pääteltävä ja luonnosteltava itse, ja suuri osa projektista kului erilaisten ratkaisujen kokeilemisessa ja nopeasti iteroiduissa prototyypeissä. Tämä lähestymistapa oli omalta osaltaan kuormittavaa ja vaati paljon uskoa omaan työhön ja osaamiseen, mutta oli omalta osaltaan myös tavallista vapauttavampaa ja palkitsevampaa.

2 SUUNNITTELUVAIHE

Projektin suunnitteluvaihe käynnistyi määrittelypalaverilla asiakkaan kanssa. Määrittelypalaverin tavoitteena oli saavuttaa yhteisymmärrys projektin tavoitteista ja arvioida yleistä toteutettavuutta. Määrittelypalaverissa asiakkaan edustajat kuvasivat tarvettaan automatisoida kuvien ottaminen myöhempää kuvien kutomista, englanniksi image stitching, varten. Kuvien kutominen ei itsessään varsinaisesti kuuluisi projektin ydinsisältöön, sillä asiakas oli jo entuudestaan harkinnut käyttävänsä kaupallisia ohjelmistoja kutomisen toteuttamiseksi, joten tälle toimenpiteelle ei ollut tarvetta kehittää täysin uutta ratkaisua. Sen sijaan itse kuvien ottaminen oli työlästä, hidasta ja puuduttavaa työtä. Asiakkaan toiveena olikin koko kuvausprosessin automatisointi: kehitettävän ratkaisun tulisi voida sille annetun määritelmän mukaisesti liikuttaa kappaletta kameran alla, sekä laukaista kamera jokaisessa kuvauspisteessä.

Projekti tulisi siis vaatimaan kappaletta liikuttelevan mekaniikan toteuttamista, ohjauskoodin laatimista kyseiselle alustalle ja integraatiota asiakkaan olemassa olevan Leica Flexacam C1 -kameran laukaisuun. Tässä vaiheessa laitteen tuottamien kuvasarjojen käsittely oltiin vielä jättämässä projektin ulkopuolelle.

2.1 Mikroskooppi ja kamera

Suunnittelu oli tietysti loogista aloittaa siitä, mitä meillä oli jo olemassa, ja minkä oli asiakkaan määrittelyn mukaisesti oltava lopputuloksen pohjana. Aloitin tutustumalla asiakkaan Leica DM1750 -mikroskooppiin. DM1750 on materiaalimikroskooppi, millä oli väliä lähinnä sen suhteen, että materiaalimikroskooppi soveltuu erityisen hyvin kuvattavaksi suunniteltujen kappaleiden kuvaamiseen. Mikroskooppia oli jo aiemmin tarkoitukseen käsin kuvaamalla käytettykin, joten itse mikroskoopin soveltumisesta määriteltyyn tehtävään ei ollut huolta. Sen sijaan oli tärkeää tutustua tarkkaan mikroskoopin ja sen kameran ominaisuuksiin, jotta niiden mahdollisuudet ja rajoitteet voitaisiin riittävästi huomioida valittaessa kuvaustasoa sekä kirjoitettaessa ohjauskoodia. Itse mikroskoopin kannalta tärkeät huomioitavat ominaisuudet olivat tässä tapauksessa mikroskoopin orientaatio ja objektiivit. Orientaation kannalta mikroskooppi oli tyypillinen pystymikroskooppi, eli objektiivi sekä sen takana oleva kamera oli asetettu näytetason yläpuolelle. Tämä vaikutti siihen, minkälainen kuvaustaso mikroskoopille tultaisiin hankkimaan. Jos orientaatio olisi ollut käänteinen, eli objektiivi olisi ollut näytetason alla ja näytettä

olisi kuvattu tasossa olevan aukon lävitse, olisi myös motorisoidun kuvaustason täytynyt olla vastaavanlaiseen orientaatioon suunniteltu. Objektiiveja mikroskoopissa taas oli 5: 5-, 10-, 20-, 50- ja 100-kertaisella suurennuksella. Tilattavan motoriikan tulisi siis olla tarpeeksi tarkka 100-kertaisella suurennuksella kuvaamista varten. Ohjelman tulisi mielellään myös toimia kaikilla suurennuksilla.

Kameran kanssa huomioitavaa oli enemmän. Ensimmäinen prioriteetti olisi selvittää, miten kameraa laukaistaisiin liikkeen aikana. Kameralle oli olemassa valmistajan oma ohjelmisto. Ohjelmalla ei voinut toteuttaa kuvattavalle kappaleelle tarpeellista liikettä, mutta kameran sillä voisi laukaista. Toinen vaihtoehto kameran laukaisulle olisi etälaukaisu kamerasta löytyvällä kytkinportilla. Tässä vaiheessa ei ollut minkäänlaista tietoa, minkälaista etälaukaisinta kamerassa on tarkoitettu käyttää, mutta oletimme laukaisun edellyttävän matalaa, todennäköisesti 5 voltin pulssia kytkinporttiin lähetettynä. Lähetin tiedustelun Leicalle asian varmistamiseksi. Päätimme kuitenkin edetä kytkinportin käytöllä kameran laukaisun suhteen, sillä Leican ohjelmistoon ei ollut tiedossa minkäänlaista käypää rajapintaa, jolla sitä voitaisiin ohjelman ulkopuolelta ohjata.

Leican ohjelmistossa oli kuitenkin käteviä mittaustyökaluja, joilla sain muodostettua kuvaa siitä alasta, jonka yksittäinen kuva kattaisi. 100-kertaisella suurennuksella kuvattava ala olisi vain noin 50 µm kertaa 40 µm. Tutkiessani kappaleita suurimmilla suurennuksilla huomasin myös, että tällä suurennustasolla harva kappale oli niin tasainen, että sen yhdestä reunasta voitaisiin siirtyä kappaleen vastakkaiseen reunaan kuvaa välillä erikseen tarkentamatta. Tämän toteuttaminen tulisi vaatimaan mahdollisuutta liikuttaa kappaletta korkeussuunnassa, eli Z-akselilla. Vaadittu korkeussuuntainen liike tulisi olemaan myös mikrometrin luokkaa, mikä asettaa huomattavia vaatimuksia hankittavalle mekaniikalle.

Tiedossa oli myös, että kuvattavat kappaleet tulisivat olemaan kooltaan 2 mm-25 mm korkeudeltaan ja leveydeltään. Jos 100-kertaisella suurennuksella kuvan leveys olisi vain 50 µm, jo pienimpien kappaleiden kuvaaminen kerralla voisi edellyttää jopa tuhansia kuvia. Leican ohjelmistolla koekuvaamalla olin myös todennut, että yksittäisen, kameran ottaman 12 megapikselin kuvan tyypillinen koko oli noin 2,5 megatavua. Toisin sanottuna, pienimpienkin kappaleiden kuvaaminen kerrallaan tuottaisi useamman gigatavun edestä raakaa kuvamateriaalia. Kameran ohjekirjan mukaan etälaukaisua käytettäessä kameraan tulee kytkeä jokin massamuistilaite, kuten esimerkiksi USB-asema, kuvien tallentamista varten. Tulevan kuvamateriaalin valtava koko tulisi ottaa huomioon soveltuvaa tallennusratkaisua valitessa.

2.2 Motorisoitu kuvaustaso

Ensimmäisenä aloin selvittämään vaihtoehtoja liikkeen mekaniikaksi. Tämä oli mielestäni perusteltua, sillä käytettävä laite tulisi asettamaan omia rajoitteita ja mahdollisuuksia, jotka olisi otettava huomioon esimerkiksi ohjauskoodin kieltä valittaessa. Laitteen tulisi olla tarpeeksi tarkka, ja laitteella pitäisi mielellään olla jonkinlainen mahdollisuus pulssin lähettämiseen kameralle laukaisua varten. Aloin tutkimaan internetin avulla tarkoitukseeni soveltuvia laitteita ja metodeja. Tässä vaiheessa törmäsin kuitenkin yhteen ikävään tosiasiaan, joka tulisi vaikeuttamaan koko projektia: Tavoittelemani kaltaista kuvaamista tehdään lähinnä ainoastaan suurten yritysten sisäisessä tutkimuksessa, ja he pitävät tekniikkansa ja metodinsa liikesalaisuuksina. Tietoa kuvaamisesta mikroskoopilla toki löytyi, lähinnä joko biologian tai harrastelijoiden osalta, mutta käytännössä mikään tästä ei ollut minulle hyödyllistä: Laitteet, jotka täyttivät näiden kuvausten vaatimukset, eivät riittäisi omiini. Joutuisin määrittelemään itse laitetarpeeni mikroskooppia tutkimalla ja päättelemällä.

Mikroskooppia käyttämällä olin saanut käsitystä siitä, minkälaisia liikkeitä laitteelta vaadittaisiin, jotta liike jokaiseen vaadittuun kuvauspisteeseen sekä kuvan tarkentaminen voitaisiin toteuttaa. Vähimmäistarkkuudeksi tulisi noin mikrometri. Jo tämä välittömästi rajasi vaihtoehdot ainoastaan useamman tuhannen euron laitteisiin. Laitteen pitäisi jotenkin mahtua mikroskoopin objektiivin alle. Tämä käytännössä tarkoittaisi joko niin pientä laitetta, että se voidaan asentaa mikroskoopin omalle kuvaustasolle, tai jollekin erilliselle mikroskoopin poikki asetettavalle tasolle, joka olisi hankittava erikseen. Kolmas vaihtoehto olisi jonkinlainen varsiratkaisu, jossa laite on mikroskoopin vieressä ja ohjaa sivulta tuotavaa kuvaustasoa objektiivin alla. Kun huomioitiin vielä tarve pulssin lähettämiselle kameran laukaisua varten, järkeviä vaihtoehtoja alkoi olla enää puolen tusinaa. Näistä karsiutui vielä pari projektille kohtuuttomien, jopa puolen vuoden lähetysaikojen vuoksi.

Saatuani järkevät vaihtoehdot kartutettua, kutsuin asiakkaan kanssa uuden, ennalta sovitun palaverin käytettävän motorisoidun tason valitsemista varten. Laadin Powerpoint -esityksen löytämistäni vaihtoehdoista ja esittelin ne tärkeimpine teknisine ominaisuuksineen, sekä omat perusteluni kunkin laitteen osalta. Vaihtoehdoiksi esittämäni laitteet on tiivistetty taulukkoon 1. Tärkeimmiksi vertailtaviksi ominaisuuksiksi olin esittämissäni laitteissa valinnut tarkkuuden, toistettavuuden, liikeradan ja I/O Trigger -ominaisuuden. Näistä tarkkuudella viitataan siihen tarkkuuteen, jolla valmistaja vakuuttaa laitteen kykenevän löytämään yksittäisen pisteen liikeradallaan, ja toistettavuus taas viittaa siihen, miten tarkasti laite voi palata johonkin määriteltyyn pisteeseen liikeradalla useiden liikkeiden aikana (Zaber 2024). Liikerata itse viittaa siihen, miten pitkät kiskot laitteessa on ja kuinka pitkälle laitteen taso voi kutakin

akselia myöten liikkua. I/O Trigger, tai Input/Output Trigger, viittaa ominaisuuteen, jossa laitteessa on jonkinlainen sisäänrakennettu mahdollisuus lähettää ja vastaanottaa liipaisupulsseja.

TAULUKKO 1. Esitetyt vaihtoehdot XYZ-tasoksi

Laite	Tarkkuus	Toistettavuus	Liikerata	I/O Trigger	Toimitusaika	Hinta
HO-MEPS-5020M	5 µm	5 µm	50 mm	Kyllä	45-60 päivää	3500€
MCL-MMP3	4 µm	100 nm	25 mm	Ei	45 päivää	9900€
X-XYZ-LSM025A	15 µm	3 µm	25 mm	Ei	1-5 päivää	6000€
M30XY + M30X	8 µm	1 µm	30 mm	Kyllä	10 päivää	7000€

Lopulta valituksi tuli intialaisen Holmarc Opto-mechatronics Ltd:n laite HO-MEPS-5020M, joka erottui ensisijaisesti hintansa vuoksi. Laite oli muita esitettyjä vaihtoehtoja noin puolet halvempi. Huomautin myös, että yhtenä vaihtoehtona, tosin ei suosittelemanani, olisi yleisesti saatavilla olevien motorisoitujen kiskojen ostaminen esimerkiksi Amazonilta ja näitä ohjaamaan esimerkiksi Arduino -laite. Hinta näille olisi ollut murto-osa jopa Holmarcin hinnasta, mutta asiakkaalla ei odotetusti ollut halukkuutta ottaa riskiä kyseisten laitteiden tarkkuuksien kanssa. Koin silti vaihtoehdon esittämisen tarpeelliseksi vähintään viitekehyksen luomiseksi sekä hankintaesitysten läpinäkyvyyden kannalta.

Kompromissina kilpailijoitaan halvemman Holmarcin tekniset ominaisuudet olivat vain täpärästi riittävät täyttämään aiemmin päättelemäni vaatimukset, ja laitteen arvioitu toimitusaika olisi pahimmillaan 60 päivää, enemmän kuin muilla esitetyillä laitteilla. Tätä ei kuitenkaan koettu ongelmaksi, sillä laitteen saapumista arvioitiin voitavaksi tämän verran odottaa. Arvioin myös päättelemissäni teknisissä vaatimuksissa olevan sen verran toleranssia, että laitteen ominaisuuksien pitäisi olla riittävät. Jouduin kuitenkin varoittamaan, että laitteen tarkkuus ei välttämättä riittäisi esimerkiksi tarkennuksen toteuttamiseksi suurimmilla objektiiveilla. Totesimme kuitenkin, että 20-kertainen suurennus saattaisi olla riittävä, ja aiempien koekuvauksieni perusteella kyseisellä suurennuksella voisi olla mahdollista kuvata tarkentamatta kuvaa erikseen liikkeen aikana. Laitteessa oli myös edukseen jo valmiina I/O Trigger portit, joten pulssin lähettämiseen kameralle ei tarvittaisi erillistä laitetta.

Lähetin valmistajalle tarjouspyynnön ja asiakkaan hyväksyttyä tarjouksen tilasin Holmarcilta HO-MEPS-5020M motorisoidun XYZ tason. Laite tulisi olemaan yhteensopiva LabVIEW -ohjelmoinnin kanssa, johon Sasken voisi tarjota minulle heiltä ennalta löytyvän kehityslisenssin, sekä tarvittaessa

tarjota muuta tukea yrityksestä löytyvän mittavan LabVIEW-osaamisen ansiosta. Tämän myötä oli helppo valita LabVIEW ohjelmiston alustaksi.

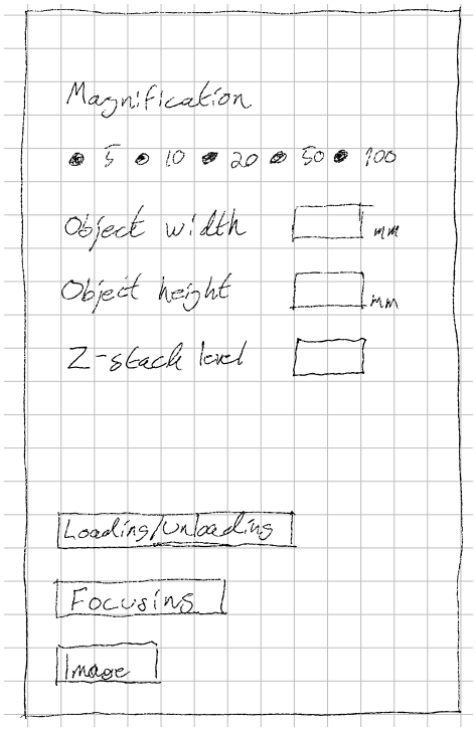
2.3 Ohjausohjelmisto

Kun ohjelmiston kehitystyökaluksi oli valittu LabVIEW, ensimmäiseksi joutuisin tutustumaan kieleen, sen mahdollisuuksiin ja oikkuihin. Mitään aiempaa kokemusta minulla ei LabVIEW'sta tässä vaiheessa ollut, mutta monilla kielillä aiemmin ohjelmoineena, en odottanut kohtaavani huomattavia vaikeuksia. Tavoitteeni oli suunnitella alustavasti ohjelmisto kuin millä tahansa muulla kielellä, ja pyrkiä sitten toistamaan rakenne LabVIEW'ssa. Aikaa LabVIEW-tutustumiseen olisi onneksi runsaasti, sillä ohjelmalla ohjattavaa laitetta jouduttaisiin odottamaan vähintään ilmoitetut kuusi viikkoa.

Periaate ohjelmalle tulisi olemaan yksinkertainen: Kuvausruudukko toteutuisi käymällä läpi kaksi sisäkkäistä For-silmukkaa. Jokaisessa pisteessä tulisi laukaista kamera, ja siirtää sitten tasoa kameran alla, kunnes kamera on seuraavan kuvauspisteen kohdalla. Liikkeen pituus pitäisi voida jotenkin syöttää ohjelmalle. Ensimmäisessä luonnoksessa suunnittelin tämän toteuttamista kuvattavan kappaleen mitoilla: kappaleen sivut tulisi siis mitata vähintään millimetrin tarkkuudella, jotka sitten syötettäisiin ohjelmaan. Ohjelma laskisi vaadittavan iteraatioiden määrän, jotta kuvausalue kattaisi kappaleen mitat. LabVIEW'ssa kehitetään aina käyttöliittymä ja koodi yhtäaikaisesti, joten aloin myös luonnostelemaan ohjelmalle käyttöliittymää, joka olisi mahdollisimman intuitiivinen ja helppokäyttöinen. Kuvassa 1 nähtävä käyttöliittymän ensimmäinen luonnos oli vielä LabVIEW -opettelua edeltävältä ajalta ja sisältää elementtejä, joita lopulliseen versioon ei koskaan tuotu. Esimerkiksi Focusing -painike jäi pois jo heti alkuunsa, koska se olisi vaatinut toimintoja, joita laitteistolla ei ollut valmiuksia toteuttaa. Toiset, kuten kuvan Loading/Unloading -painike, jonka tässä luonnoksessa olin kuvitellut tilaksi, jossa kuvaustaso tuodaan johonkin helposti ulottuvissa olevaan asentoon kuvattavan kappaleen tasolle asettamista varten, säilyivät mukana periaatteellisesti samoina, mutta toiminnallisesti hyvin erilaisina. Silti tämän ensimmäisen luonnoksen ydinrakenne ja periaate tulisi kaikumaan aina lopulliseen versioon asti.

Kun LabVIEW -opettelu alkoi tuottaa tulosta, oli aika alkaa tehdä pohjaa itse ohjausohjelmistolle. Minulla ei ollut tässä vaiheessa vielä laitetta, jota ohjata, joten ei ollut vielä mitään mieltä alkaa tekemään

rajapintaa laitteelle. Sen sijaan keskityin aiemmin hahmotelleeni ohjelman ydinperiaatteen toteuttamiseen LabVIEW'lla. Tämä tarkoittaisi siis kahden sisäkkäisen For-silmukan toteuttamista siten, että ne käyvät läpi kaksiulotteisen taulukon jokaisen indeksin oikeassa järjestyksessä.



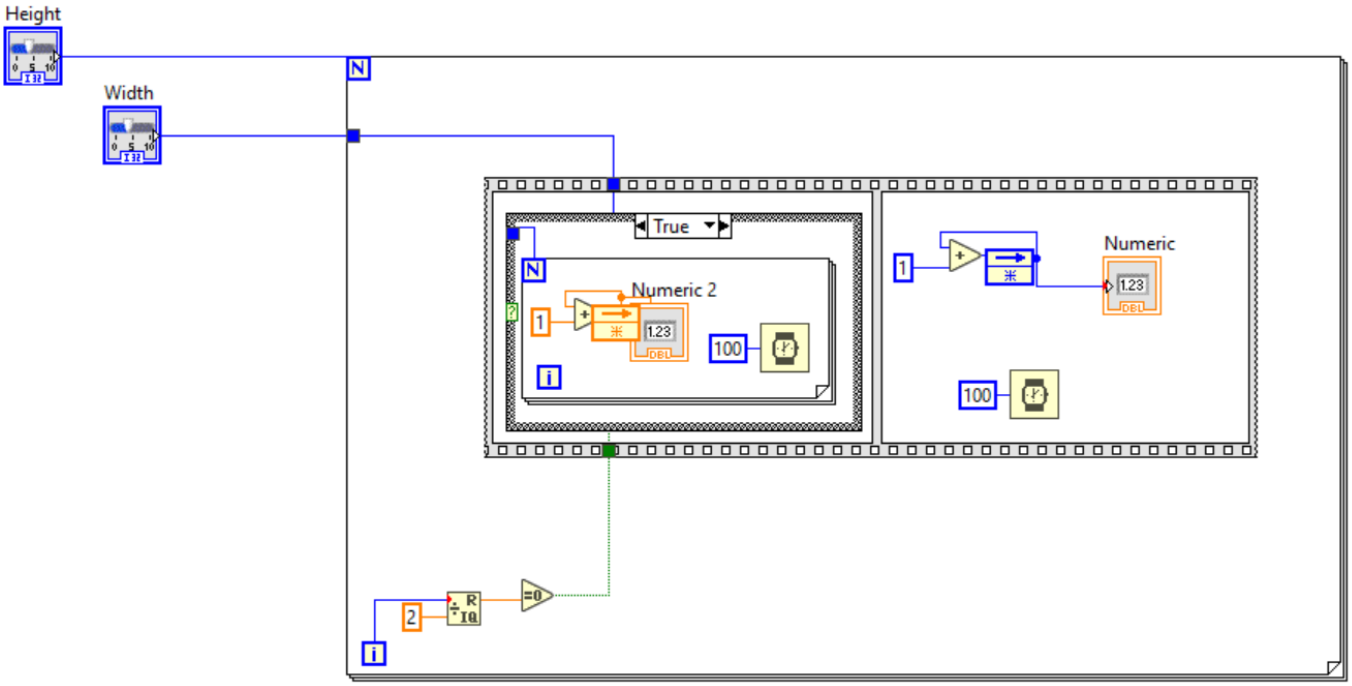
KUVA 1. Käyttöliittymän ensimmäinen luonnos

Lähdin rakentamaan LabVIEW'ssa ohjelman ensimmäistä alustavaa versiota, jonka kaavio on nähtävillä kuvassa 2. Tässä vaiheessa etupaneeli oli ainoastaan kaksi syöttöä, joilla ohjelmalle syötettiin kuvitteelliset arvot kuvitteellisen kappaleen leveydeksi ja pituudeksi, sekä kaksi näyttöä, joista voisimme seurata iteraatioiden kulkua. Kuvan 2 mukaisesti korkeus määrittäisi uloimman silmukan iteraatioiden määrän, joten tulisimme kuvaamaan vaakarivejä. Ulompi silmukka koostuu LabVIEW'n sekvenssirakenteesta, jossa ensin ajetaan sisempää silmukkaa, ja lopuksi ajetaan itse ulomman silmukan tehtävä. Tämä tehtävä tulisi jatkossa olemaan kuvan ottaminen ja rivin siirtäminen alaspäin, mutta tässä versiossa olisi riittävää vain inkrementoida korkeusakselin iteraatiolaskuria.

Sekvenssirakenne oli tarpeellinen, koska graafisena ohjelmointikielenä LabVIEW'ssa ei ole mahdollista jaksottaa koodia pohjaten esimerkiksi siihen periaatteeseen, että koodirivit suoritettaisiin aina ylhäältä alas. Sen sijaan LabVIEW ohjelma suoritetaan siinä järjestyksessä, missä data saapuu rakenteisiin ja toimintoihin (National Instruments 2024). Tämä tarkoittaisi sitä, että ilman sekvenssirakennetta,

kaksi rakennetta ajetaan saman aikaisesti, jos ne saavat tarvittavan datan samaan aikaan, joka voi aiheuttaa ohjelman suorituksessa ajoitusvirheitä.

Sekvenssirakenteen alla näkyy lisäksi pieni rakenne, joka tarkistaa korkeusakselin iteraation parillisuuden hakemalla iteraation indeksin LabVIEW'n For-silmukkarakenteen I-terminaalista. Kun tämän perusteella ohjattiin sisempää silmukkaa ajamaan positiivista tai negatiivista versiota Case-rakenteessa, voitiin toteuttaa liike, jossa joka toinen rivi kulkisi vastakkaiseen suuntaan. Näin kuvaaminen tulisi tapahtumaan tehokkaasti ilman tarvetta palata aina esimerkiksi vasempaan laitaan uuden rivin kuvausta varten. Kellon kuvat kaaviossa olivat viiveitä millisekunneissa määriteltyinä, jotka helpottivat liikkeen havainnoimista etupaneelista. Tämä kaavio oli vielä hyvin pieni ja ohjelma hyvin yksinkertainen, mutta lopputuloksena oli silti toimiva simulaatio toivotusta ohjauksesta. Tämä perusrakenne tulisi pysymään kuvaussyklien ydinlogiikkana aina viimeiseen versioon asti.



KUVA 2. Ohjelman ensimmäinen versio

3 ALUSTAVA KEHITYS

Koen suunnitteluvaiheen päättyneen ja alustavan kehityksen alkaneen, kun tilaamani laitteet viimein saapuivat pitkän odotuksen jälkeen. Tässä vaiheessa pitkittynyt suunnitteluvaihe alkoi nopeasti kantaa hedelmää, sillä LabVIEW'iin oli ollut hyvin aikaa tutustua ja kun yhteys Holmarcin motorisoituun tasoon oli saavutettu, haluttujen liikkeiden ohjelmointi oli yllättävän mutkatonta ja helppoa. Tämä vaihe oli täynnä ohjelmiston nopeita iteraatioita ja paitsi suunniteltujen ominaisuuksien toteuttamista, myös toisinaan niiden hylkäämistä tai karsimista tarpeettomina, sekä kokonaan uusien, tarpeellisten ominaisuuksien tuomista mukaan kokonaisuuteen. Vaihe kuitenkin edellisen tavoin pitkittyi huomattavasti, ehkä eniten siksi, että mikroskoopin ja kameran valmistaja Leica ei juurikaan vastannut tiedusteluihini, ja oli haluton luovuttamaan laitteiden teknisiä tietoja käyttööni.

Jouduimme luottamaan kameran suppeaan käyttöohjeeseen, sekä tekemään maltillisia ja varovaisia olettamuksia kameran toiminnasta. Tämän vaiheen suurin haaste olikin huomata, ettei kameraa voitaisi laukaista suunnitellusti suoraan lähettämällä sille 5 voltin pulssi. Leican kameran ja Holmarcin ajurin välille olisi tehtävä laukaisupiiri, joka voisi muuttaa viiden voltin pulssin kameran kytkinpiirin sulkemiseksi, jolloin kamera laukeaisi.

3.1 Motorisoitu kuvaustaso

Laitteiden vihdoin saavuttua oli syytä alkaa tutustumaan niihin välittömästi. Kuvaustaso saapui tiiviisti pakattuna ja valmiiksi kasattuna, ja kaikki tarvittavat kaapelit tulivat laitteen mukana. Laitteen ajuri tuli odotetusti RS-232 -liitännällä, ja laitteen mukana tuli myös oma RS-232 – USB-adapteri, jonka avulla laite voitiin liittää suoraan useimpiin moderneihin tietokoneisiin. Itse kuvaustason kolmessa kiskossa olevat moottorit kytkettiin ajuriin koaksiaalikaapeleilla. Trigger OUT -porttia varten ei tullut minkäänlaista kaapelia, joten ensitöiksi sellainen oli tehtävä juottamalla koaksiaaliliitäntä kelasta otettuun pätkään kaapelia. Kaapelin toinen pää olisi joka tapauksessa oltava kameralle mukautettu, joten kaapelin tekeminen kokonaan itse ei sinänsä vienyt huomattavasti ylimääräistä aikaa. Kaapelin toiseen päähän tuli 3,5 mm pistoke, mutta tämä tulisi pian paljastumaan vääräksi liittimeksi kameraa varten.

Kun laite oli kytketty, oli aika kokeilla sen toimivuutta. Valmistaja oli toimittanut laitteen mukana oman ohjausohjelmistonsa. Ohjelmistossa oli yksinkertaiset työkalut tason liikutteluun sen kiskoja pitkin, sekä alkeellinen ohjelmointi liikesarjojen tekemiseen. Nämä liikesarjat olisi pitänyt kuitenkin syöttää liike kerrallaan, joten odotetusti ohjelmalla ei ollut erityisesti arvoa projektin kannalta. Sillä oli kuitenkin arvoa testaustyökaluna, sillä ohjelmalla voitiin helposti todeta, että laite totteli odotetusti sen käskyjä ja näytti olevan moitteettomassa kunnossa. Myös Trigger OUT -liipaisupulssi testattiin mittaamalla Trigger OUT -porttia yleismittarilla, ja pulssit olivat odotetusti noin 5 volttia. Seuraava vaihe olisi saada yhteys laitteen ja LabVIEW'n välille. Tämä osoittautuisi hetkellisesti haastavaksi, sillä laitteen mukana ei tullut minkäänlaista dokumentaatiota, ja laite ei vastannut tyypillisiin standardien mukaisiin käskyihin. Tiedustelin luonnollisesti heti dokumentaatiosta laitteen valmistajalta, mutta menisi pari päivää ennen kuin saisin vastauksen. Tänä aikana jouduin varautumaan siihen, että riittävää dokumentaatiota ei välttämättä koskaan saapuisi, ja aloin itse selvittämään laitteen komentorakennetta.

Lähdin siitä periaatteesta, että koska laite toimi valmistajan ohjelmiston avulla moitteetta, ja ohjelmistolla voitiin toistaa kaikki tarvitsemani yksittäiset perustoiminnot, olisi kaikki tarvitsemani käskyt laitteen ja ohjelmiston välisessä kommunikaatiossa. Koska kaikki liikenne kulki yhden USB-portin kautta, asetin tavoitteekseni tuon portin dataliikenteen lukemisen. Tässä haasteeksi muodostui laitteen ja tietokoneen välisen yhteyden luonne, sillä vaikka USB-portteja on suhteellisen helppoa käsitellä, RS-232 yhteyksiä ei ole. RS-232 kytketään aina kiinteänä resurssina kiinni johonkin yhteen tiettyyn käyttöön. Tämä tarkoittaisi, että laite voisi keskustella vain yhden ohjelman kanssa samanaikaisesti. Löysin helposti käyttööni monia portinkuunteluohjelmia, joilla olisin dataliikennettä voinut kuunnella tai jopa lähettää viestejä, mutta millään tällä ei olisi väliä, jos en pystyisi lähettämään valmistajan ohjelmiston avulla haluamani käskyjä ja verrata niitä portin dataliikenteeseen.

Tilanteen ratkaisu löytyi lopulta omista harrastuskokemuksista, sillä olin aiemmin omaan käyttöön tutustunut laajasti tietokoneen erilaisten porttien virtualisointiin. Jos on mahdollista virtualisoida yhdenlainen portti, miksi ei myös nyt ongelmana oleva RS-232 sarjaportti? Tämän periaatteen siivittämänä löysin Electronic Team, Inc -valmistajan ohjelmat nimeltä Virtual Serial Port Driver sekä Serial Port Monitor. Ohjelmista oli saatavilla 14 päivän kokeiluversiot, joten lähdin selvittämään ongelmaani näiden avulla. Virtual Serial Port Driverin avulla onnistuinkin virtualisoimaan portteja siten, että todellinen fyysinen portti lähettäisikin kaiken dataliikenteensä virtualisoituun solmukohtaan. Kaikki tähän solmukohtaan saapuva liikenne lähtisi kahteen virtualisoituun porttiin: toisessa portissa oli Holmarcin ohjausohjelmisto, ja toisessa Serial Port Monitor. Vastaavasti kaikki Holmarcin ohjelmistosta tähän solmukohtaan lähtevä liikenne ohjattaisiin sekä itse laitteelle että Serial Port Monitorille. Metodi oli

toimiva, ja yhteys laitteelle toimi ongelmitta tässä konfiguraatiossa. Sen lisäksi laitetta ohjattaessa Holmarcin ohjelmalta, Serial Port Monitor alkoi välittömästi kaappaamaan edellä mainittujen välistä liikennettä.

Omaa ohjelmaani varten tarvitsin käskyt laitteen liikuttamiseen ja liipaisupulssin lähettämiseen, joten aloin selvittämään noita toimintoja Holmarcin ohjelmalla suoritettaessa aiheutuvaa dataliikennettä portissa. Otin ensimmäiseksi käsittelyyn laitteen liikuttamisen, ja aluksi näyttikin, että sekavaa liikennettä oli hirveä määrä Holmarcin ohjelmalta laitteelle ja takaisin. Oikeiden ohjauskäskyjen selvittäminen vaati systemaattista otetta. Otin Holmarcin ohjelman ja laitteen välisestä dataliikenteestä otannat, joissa yksi kerrallaan liikutin jokaista kolmesta kiskosta saman etäisyyden. Näistä koeajoista alkoi nousta esiin kaava. Holmarcin ohjelma lähetti ensin aina saman alustavan käskyn, johon laite vastasi aina toisella, mutta toistuvasti samalla koodilla.

Oletin, että kyseessä on komennon alustuskoodi sekä sen hyväksyntä laitteelta. Tämän jälkeen laite lähetti kolme pitkää koodisarjaa, joiden päätteeksi tuli aina yksi rivi koodia, jotka olivat keskenään identtisiä. Lopuksi tuli vielä kaksi ylimääräistä riviä, jotka olivat myös keskenään identtiset. Jokaista yksittäistä koodin pätkää seurasi myös aina sama hyväksyntäkoodi laitteelta. Tämän jälkeen laite suoritti liikkeen ja lähetti lopulta vielä yhden rivin koodia, joka oli poikkeava muista laitteelta tulleista koodeista. Oletin kyseessä olevan jonkinlainen päättökoodi, jolla laite toteaa ohjauksen omalta osaltaan päättyneen. Tärkeää oli verrata ensimmäisen alustuskoodin jälkeen lähetettyjä kolmea koodisarjaa, jotka kukin koostuivat kolmesta rivistä.

Nämä rivit olivat aina nollaa, paitsi yhdessä sarjassa, jossa ne saivatkin muita arvoja. Nämä muuttuneet arvot olivat aina samat mutta tulivat aina eri koodisarjassa jokaisella koeajolla. Toisin sanottuna nämä arvot olivat niitä, jotka muuttuivat haluamani liikkeen mukaisesti, ja se missä sarjassa ne olivat, riippui kiskosta, jonka käskin liikkua. Sarjat vastasivat järjestyksessä X-, Y- ja Z-kiskoja. Testasin teoriaani tekemällä yksinkertaisen LabVIEW-ohjelman, jonka ainoa sisältö oli yksittäinen kirjoitustapahtuma laitteelle. Kirjoitin komentokoodin käsin pohjaten löydöksiini, ja laite alkoi liikkua haluamallani tavalla.

Valitettavasti tämä oli tarpeetonta työtä, sillä vielä samana iltapäivänä sain vihdoin vastauksen Holmarcilta, täyden dokumentaation laitteen komentokäskytyksestä sekä heidän omat LabVIEW-ajurinsa, jotka jouduttaisivat työtäni mukavasti. Olin kuitenkin tyytyväinen selvitystyöhöni, sillä Holmarcin do-

kumentaatio vahvisti kaikki löydökseni, ja vaikka olisin joutunut jatkamaan tutkimusta vielä hyvän aikaa saadakseni selville kaikki tarvitsemani käskyt, olin silti onnistuneesti päätellyt toimivan metodin laitteen käskyjen selvittämiseksi haastavassa tilanteessa.

3.2 Ohjausohjelmisto

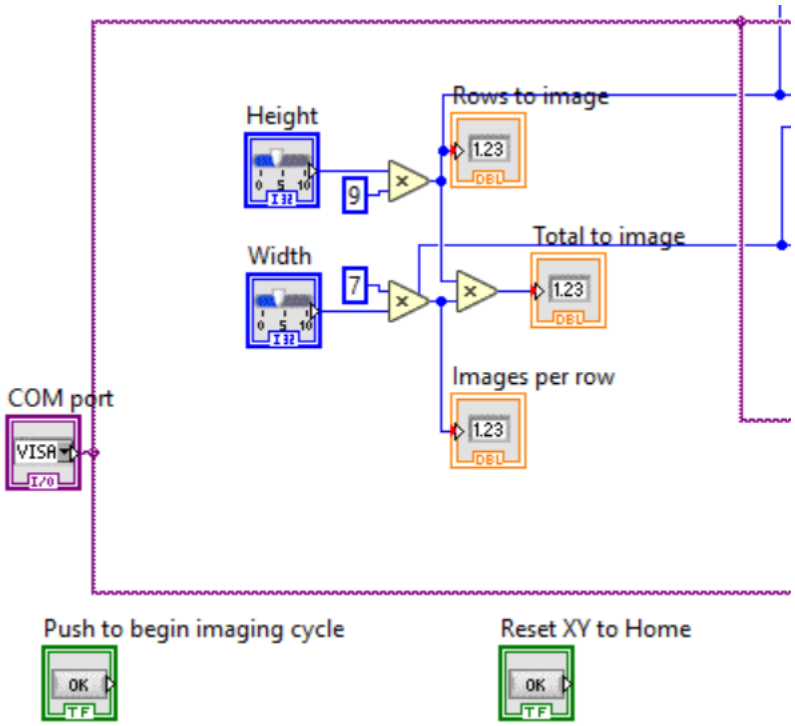
Holmarcin motorisoitu kuvaustaso oli nyt täysin hallittavissa LabVIEW'n kautta, joten oli aika siirtyä ohjelmistokehityksen pariin. Ensimmäinen tavoitteeni oli aiemmin laatimani ohjelmapohjan mukauttaminen siten, että siinä jo toteutettu rakenne saatiin täydennettyä Holmarcin LabVIEW-ajureilla. Seuraavan muutaman viikon aikana ohjelmiston kehitys kulkisi nopeasti periaatteessa toimivasta ydinjärjestelmästä käytännössä käyttövalmiiksi ohjaustyökaluksi. Merkittävien kehitysvirstanpylväiden kohdalla merkkasin ohjelman aina seuraavaan kehitysversioon. Koin versiohallinnan LabVIEW'n kanssa haastavaksi, joten pidin versioista kirjaa puhtaasti numeroimalla ja arkistoimalla vanhat versiot tarpeen varalta. Tämä olisi ollut kohtuuttoman kankeaa tehdessä yhteistyötä muiden kehittäjien kanssa, mutta tässä projektissa tekisin kaiken kehitystyön itse ja jatkokehityksen kannalta lopullisen version lähdekoodin riittävä dokumentointi tulisi riittämään.

3.2.1 Kehitysversio 2

Tässä vaiheessa ohjelma jakautui jo siististi kahteen lohkoon. Näistä ensimmäinen oli ohjauslohko (KUVA 3), joka koostuisi käyttöliittymään liitetyistä terminaaleista, joihin käyttäjä voisi asettaa haluamiaan arvoja tai painikkeita painamalla aloittaa toimintoja. Ohjauslohkoon tuli tässä versiossa useita uusia osia. COM-portin valinta oli välttämätön, sillä sen avulla voitiin osoittaa ajureille, missä tietokoneen COM-portissa yhteys Holmarcin laitteeseen olisi. Korkeus- ja leveysasetukset olivat nyt saaneet peräänsä kertoimet, joiden avulla terminaalien osoittama millimetrimäärä kääntyi laskennallisesti vaadittuun määrään rivejä ja sarakkeita kuvaussykliä varten.

Kertoimet perustuivat tässä versiossa 20-kertaiseen suurennukseen. Näiden pohjana olevat mittaukset oli tehtävä kokeilemalla, koska laitteen pienet epätarkkuudet tekivät kertoimien löytämisen laskennallisesti yllättävän vaikeaksi. Valitsin tämän suurennuksen tässä vaiheessa kehitystyön pohjaksi, koska laitteen valinnan yhteydessä oli samalla päätetty 20-kertainen suurennus suurimmaksi vaadituksi ku-

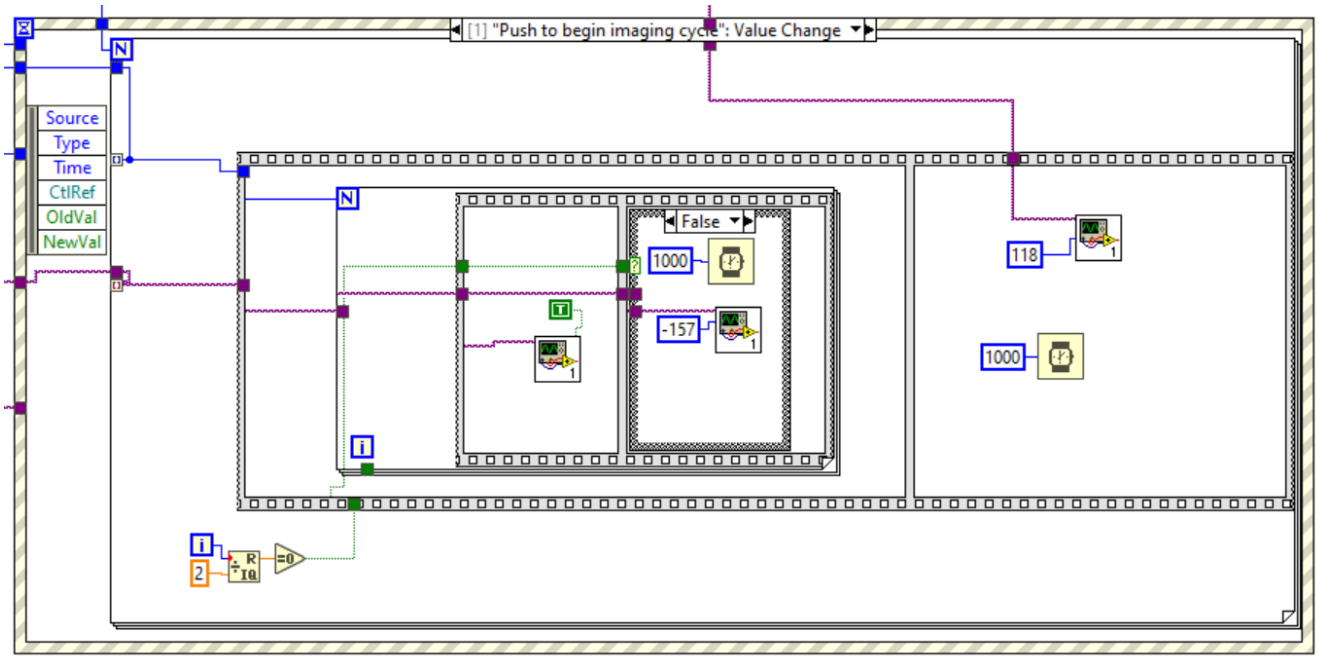
vaustarkkuudeksi. Pienempien suurennusten kuvaaminen oli oletukseni pohjalta helpompaa, joten tavoite oli kehittää ohjelmaa ensisijaisesti toimimaan vaativimmissa kuvaustapauksissa. Nämä arvot, sekä niistä johtuva yhteenlaskettu kuvattava kuvamäärä, näytettäisiin myös käyttäjälle uusien näyttöterminaalien avulla. Lopuksi lohkoon lisättiin vielä painikkeet, joilla voitaisiin aloittaa tämän version kaksi toimintoa: itse kuvaussykli, sekä palautustoiminto tason palauttamiseksi nollapisteeseen, joka oli perua alkuperäisen käyttöliittymäluonnoksen Loading/Unloading -toiminnosta.



KUVA 3. Version 2 ohjauslohko

Toiseksi lohkoksi tuli tapahtumalohko. Tapahtumalohko koostui LabVIEW'n Event-rakenteesta, joka on rakenne, joka sisältää tapahtumamääritelmien alle järjestettyjä diagrammeja. Rakenne odottaa, kunnes jokin sen määrittelemistä tapahtumista tapahtuu, ja suorittaa sitten tuota tapahtumaa vastaavan diagrammin. Event-rakenteen ensimmäisen tapahtuman tulisi aina olla Timeout, eli aikakatkaisu. Tämä johtuu siitä, että Event-rakenne varaa koko ohjelman odottaessaan tapahtumaa. Ohjelma ei voi tehdä mitään muuta, ennen kuin Event-rakenne havaitsee tapahtuman ja on ajanut haluamansa diagrammin. Jos rakenteeseen lisätään diagrammin osalta tyhjä Timeout-tapahtuma ja aikakatkaisuksi asetetaan muutama kymmentä millisekuntia, Event-rakenne ei lukitse koko ohjelmaa odottaessaan tapahtumaa. Oikeastaan näin edelleen tapahtuu, mutta käyttäjä ei sitä huomaa, koska Event-rakenne vapauttaa ohjelman muutaman kymmenen millisekunnin välein.

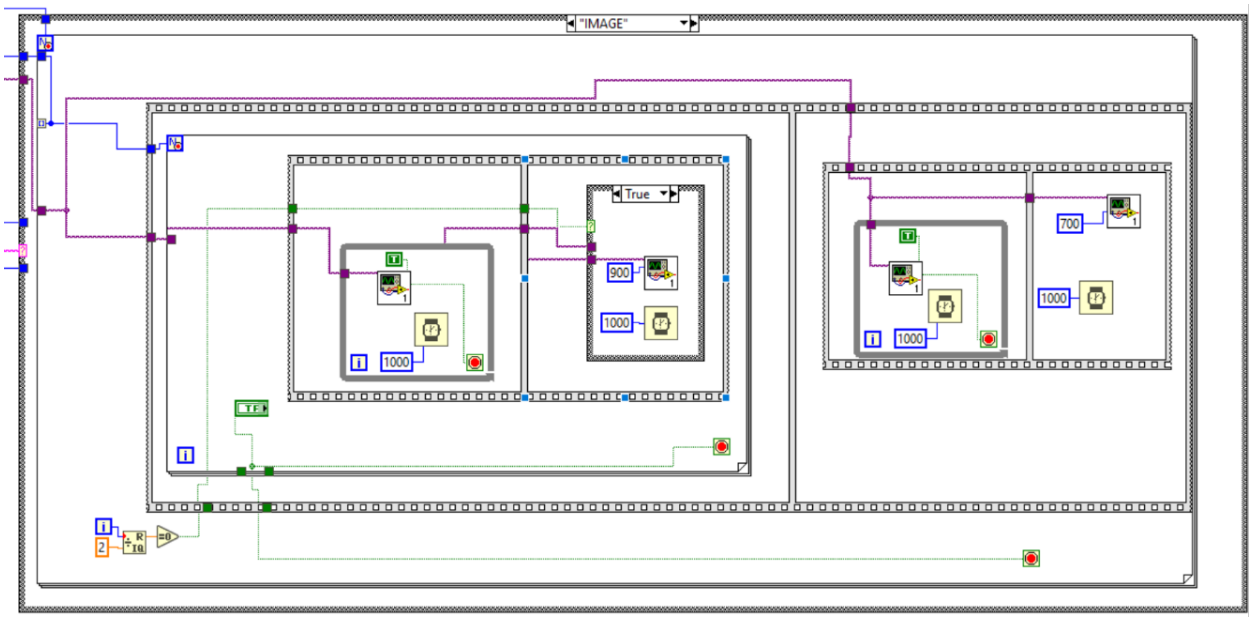
Event-rakenteen toinen tapahtuma oli ”Reset XY”, eli palautus nollapisteeseen. Tämän tapahtuman diagrammi oli yksinkertaisuudessaan vain Holmarcin ajuripalikka, jolle syötettiin COM-portin osoite sekä tarpeeksi etäisyyttä negatiivisena, jolloin tapahtuma palauttaisi korkeus- ja leveyssuunnan kiskot takaisin nollapisteeseen. Kolmas tapahtuma oli itse kuvaussykli (KUVA 4), jonka periaate periytyi ensimmäisestä versiosta. Puhtaasti teoreettiset laskimet oli nyt korvattu Holmarcin ajuripalikoilla, joille syötettiin COM-portin osoite, sekä etäisyys, joka 20-kertaisella suurennuksella siirtäisi kuvausaluetta noin kaksi kolmannesosaa sen pituudesta eteenpäin. Näin saataisiin aikaan tarvittava marginaali kuvien kutomista varten. Tämä marginaalin pituus perustui asiakasyrityksen suosituksiin, jotka puolestaan perustuivat kokemuksiin kuvien kutomisesta käsin kuvienmuokkausohjelmilla. Marginaali tarvitaan, jotta vierekkäisiä kuvia voi verrata toisiinsa. Kuvien asettelu limittäin näiden marginaalien perusteella on kuvien yhteen kutomisen perusperiaate. Tässä versiossa oli myös jo mukana liipaisupulssin lähetys sisemmässä silmukassa tulevia testejä varten. Vaikka toinen versio oli kokonaisuudessaan vielä kovin vaatimaton, koeajaessa ohjelmaa Holmarcin laitteella voitiin todeta kuvaussyklin liikkeen toimivan jo toivotulla tavalla.



KUVA 4. Version 2 tapahtumalohko ja kuvaussykli

3.2.2 Kehitysversio 3

Kolmas versio oli monin tavoin huomattava harppaus. Minimivaatimukset oli jo täytetty, joten oli aika alkaa ajattelemaan käytettävyyttä ja muita asiakkaan toivomia ominaisuuksia. Ensimmäinen huolen aihe, ja samalla perimmäinen syy uudelle versiolle, oli itse ohjelman perusrakenne. Vaikka aiemman version ohjauskoodi toimi jo ongelmitta, toiminnassa oli huomattava ongelma. Event-rakenne varasi koko ohjelman, kunnes sen sisältämä diagrammi oli ajettu loppuun. Ohjelma ei tee mitään tilan tarkistuksia tänä aikana, eikä etupaneeli reagoi mitenkään käyttäjän ohjaukseen. Tämä oli kohtuutonta ennen kaikkea siksi, että yksittäinen kuvaussykli saattaisi kestää minuutteja. Olisi täysin kohtuutonta ajatella, että käyttäjän pitäisi odottaa kuvaussyklin loppuun, jos kuvaussyklin alussa huomattaisiin jotain, joka tekisi syklin lopputuloksen käyttökelvottomaksi. Tästä esimerkkeinä voisi olla riittämätön kohdennus, tai vääränkokoinen kuvausalue. Toiminnot oli saatava Event-rakenteen ulkopuolelle. Näin syntyi ohjelmiston kolmen lohkon rakenne, joka tulisi säilymään lähdekoodin rakennekehyksenä loppukehityksen ajan. Ohjauslohkon ja tapahtumalohkon jatkoksi tuli uusi toimintolohko (KUVA 5), joka koostui ennen tapahtumalohkon sisään sijoitetuista toimintodiagrammeista.

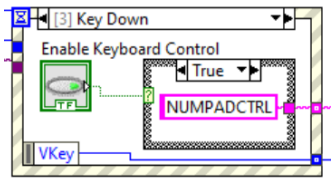


KUVA 5. Version 3 uusi toimintolohko

Toimintolohkon rakenteeksi tuli LabVIEW'n Case Structure -rakenne, jossa osa ohjelman diagrammista voitaisiin määritellä erilaiseksi halutunlaisen valintamäärityksen perusteella. Rakenne oli minulle tuttu monista muista, lähinnä C-pohjaisista ohjelmointikielistä, joissa samankaltainen rakenne kulkee nimellä switchcase. Ohjelman uudeksi toimintaperiaatteeksi tuli tapahtumalohkon käyttö ainoastaan

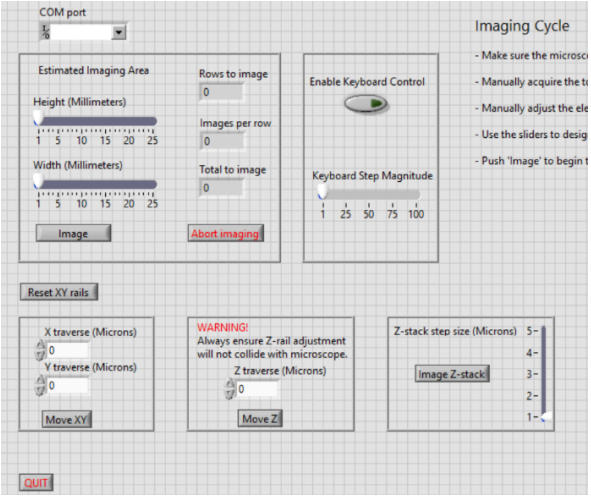
eräänlaisena käsittelijänä, joka seuraa käyttäjän toimintaa ja tiettyjen tapahtumamääritelmien toteutuessa lähettää toimintolohkolle valintamuuttujan. Toimintolohko muuttuu tämän tiedon perusteella haluttuun diagrammiin, ja toivottu toiminto toistetaan. Oli myös tarpeellista pitää mukana Timeout -tapahtuma, jonka vastaava toimintolohkon diagrammi pysyi tyhjänä.

Toinen syy tälle huomattavalle rakennemuutokselle oli asiakkaan toivoma käsiohjaus, joka helpottaisi laitteen käyttämistä huomattavasti, ja mahdollistaisi kuvaustason jättämisen paikoilleen myös mikroskoopin muun käytön aikana. Käsiohjausta oli mahdotonta toteuttaa luontevasti irrottamatta tapahtuma- ja toimintolohkoja omiksi lohkoikseen. Tämän jälkeen käsiohjauksen toteuttaminen oli kuitenkin varsin vaivatonta. Käsiohjaus tuli tapahtumalohkossa yhden tapahtuman alle (KUVA 6), joka todettaisiin aina näppäimistön painikkeita painaessa ja jossa käyttöliittymässä olevan kytkimen perusteella käsiohjauksen todettiin olevan joko pois päältä, jolloin tapahtuma lähettäisi saman tyhjään diagrammiin johtavan valintamuuttujan kuten Timeout -tapahtumassa, tai käsiohjauksen ollessa päällä uuden käsiohjauksen valintamuuttujan. Käsiohjauksen diagrammi koostui toisesta Case-rakenteesta, jonka valintamuuttujana puolestaan toimi tapahtumalohkon käsiohjaustapahtuman apumuuttuja, joka tunnistaisi painetun näppäimistön painikkeen ja valitsisi sen perusteella diagrammin, jossa Holmarcin ajurille lähetettäisiin käsky liikkeelle haluttuun suuntaan. Liikkeen pituus tuli erillisestä terminaalista ohjauslohkosta, jota voitaisiin hallita käyttöliittymästä.



KUVA 6. Version 3 tapahtumalohko ja käsiohjaustapahtuma

Uuden huomattavan toiminnon myötä tuli myös tarve asetella käyttöliittymää uusiksi. Vaikka käyttöliittymä muuttuisi vielä jokaisessa tulevassa versiossa, kolmannessa kehitysversiossa määritelty yleisasetelma (KUVA 7) pysyisi käyttöliittymän rakenteena aina lopulliseen versioon asti. Tarvetta näyttävälle käyttöliittymälle sisäiseen käyttöön tarkoitetussa työkalussa ei ollut, mutta oli silti tarpeen esittää ohjelmiston toiminnot selkeästi ja helposti ymmärrettävästi. Kaikki toiminnot lajiteltiin siististi omiin rajattuihin alueisiinsa, ja toiminnoille kuuluvat painikkeet ja kentät pyrittiin pitämään toimintojen omilla alueilla. Vain koko ohjelmaa koskevat kentät jätettiin alueiden ulkopuolelle.



KUVA 7. Version 3 käyttöliittymä

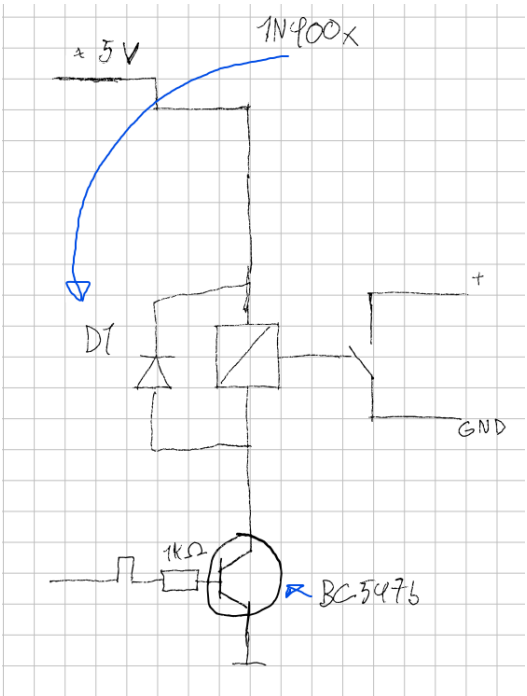
Kolmas kehitysversio oli mittava harppaus ohjelmiston kehityksessä ja samalla merkkasi alustavan kehityksen iteroivan vaiheen päättymistä. Kolmannessa versiossa ohjelmisto ohjasi jo kuvausalustan liikkeitä moitteitta ja kaikki liikkeelle asetetut tavoitteet oli saavutettu. Seuraavaksi olisi varmistettava liipaisupulssien toimivuus kameran laukaisussa ja aloitettava laitteen koeajot. Nykyisen version puutteet tulisivat helpoiten esiin, kun kokonaisuutta ajetaan varsinaista käyttötarkoitusta simuloivissa oloissa. Ennen testausvaiheeseen siirtymistä joutuisin kuitenkin saamaan ohjelmiston laukaisemat liipaisupulssit kulkemaan Holmarcin laitteesta kameraan, joka tulisi aiheuttamaan oman, odottamattoman kehitysvaiheensa.

3.3 Laukaisupiiri

Useista yrityksistä huolimatta mikroskoopissa käytettävän kameran valmistaja Leica oli ollut jo pitkään vastaamatta sähköpostitiedusteluihin tarkemmista teknisistä tiedoista kameraansa liittyen. En halunnut ottaa riskiä liipaisupulssin lähettämisestä porttiin, jonka toimintaperiaate ei ole täysin selvillä, joten aloitin asiasta oman selvitystyöni. Ensimmäiseksi mittasin kameran kytkinportin yleismittarilla, ja totesin portissa olevan 1,3 voltin jännitteen kameran ollessa päällä. Tämä oli huolestuttavaa, sillä liipaisupulssia odottavassa portissa ei mielestäni olisi pitänyt olla juurikaan jännitettä. Toinen mahdollisuus olisi, että porttiin tarkoitettu etälaukaisin olisi yksinkertaisesti vain kytkin: painiketta painamalla se sulkisi piirin kytkinportin positiivisesta navasta sen negatiiviseen napaan.

Olin jo aiemmin tehnyt kaapelin oikealla 2,5 mm liittimellä kytkinporttiin kytkemistä varten, mutta en ollut vielä tehnyt siihen Holmarcin laitteelle menevään päähän tarpeellista liitintä, jolla se voitaisiin yhdistää aiemmin tekemäni kaapelin 3,5 mm liittimeen. Tämä kaapelin toinen pää oli siis niin sanotusti auki, kahtena paljaana johtona. Kytkin kaapelin kameraan, laitoin kameran päälle ja varmistin, että kamerassa oli vaadittu massamuistiasema. Yhdistin kaapelin avoimet päät, liittäen kytkinportin navat toisiinsa, ja kamera otti kuvan. Tämä todisti epäilyni portin toimintaperiaatteesta, mutta samalla ilmeni se tosiasia, että Holmarcin laitetta ei voisi yhdistää kameraan suoraan. Olisi voitava jotenkin muuntaa Holmarcin laitteelta tuleva 5 voltin liipaisupulssi kytkimen painallukseksi, joka yhdistäisi kytkinportin navat.

Koska minulla oli vain vähän elektroniikan osaamista, pyysin tähän tukea asiakasyritykseltä, sillä heillä on runsaasti osaamista, työkaluja ja resursseja elektroniikan osalta, ja he auttoivat minua suunnittelemaan yksinkertaisen virtapiirin (KUVA 8) liipaisupulssin muuntamiseksi kytkintoiminnoksi. Periaate oli yksinkertainen transistoripiiri, jossa liipaisupulssi lähetettäisiin ohjamaan transistoria, joka puolestaan ohjaisi virtaa releelle. Releen saadessa virtaa se yhdistäisi sisäiset napansa, joihin kameran kytkinportista tuleva kaapeli yhdistettäisiin. Releen käyttövirta saataisiin pieneltä 5 voltin virtalähteeltä. Transistorin kantaan tuleva liipaisupulssi kulkisi ensin yhden kilo-ohmin etuvastuksen läpi. Virtapiikkien varalta piiriin tulisi vielä diodi rinnakkaiskytkentään releen kanssa. Piirin tarkat komponentit ovat listattuna liitteessä 2.



KUVA 8. Laukaisupiiri

Piirin valmistamisessa oli omat haasteensa, mutta nämä johtuivat lähes kokonaan omasta kokemattomuudesta elektroniikan parissa. Toisaalta piirin rakentaminen omatoimisesti minulle tehdyn piirikaavion pohjalta oli omiaan kokemuksen kehittämiseksi. Samalla tuli todetuksi myös käydyn koulutuksen arvo, sillä vaikka käymässäni tietotekniikan koulutusohjelmassa on sähkötekniikkaa vain vähän, oli se silti riittävää siihen, että ymmärsin välittömästi piirin periaatteen sekä jokaisen komponentin toiminnan. Eniten aikaa tulikin hukattua yksinkertaiseen huolimattomuusvirheeseen. Kun olin saanut piirin valmiiksi, aloin testaamaan sitä generaattorin ja yleismittarin avulla. Generaattori oli tässä tapauksessa pieni virtalähde, jolla pystyin kahden eri kanavan kautta lähettämään haluamiani jännitteitä piirille pienien pihtien avulla. Yksi kanava oli piirin käyttövirtaa varten ja toinen ohjauspulssia varten. Molempien jännite oli 5 volttia.

Kokeiden tulokset eivät kuitenkaan olleet lupaavia, ja pitkään epäilin ongelman olevan joko tinausjäljessäni tai väärässä testaustekniikassa. Lopulta tarkistettuani koko piirin uudelleen läpi löysin niin sanotun aloittelijamokani: olin asentanut transistorin väärin päin. Kun transistorin kollektori vaihdettiin piirin positiiviselle puolelle ja emitteri negatiiviselle, piiri alkoi toimia suunnitellusti ja rele sulkeutui toivotunlaisesti transistorin saadessa ohjauspulssin. Asennettuani piirin mikroskoopin kameran ja Holmarcin laitteen välille saavutettiin jälleen tärkeä virstanpylväs, kun kameran laukaisu saatiin viimein toimimaan. Seuraava kuvaussyklin koeajo tuotti pitkään odotetun ensimmäisen kuvasarjan.

4 TESTAUS JA OPTIMOINTI

Kun laitteen kaikki perustoiminnot oli saatu toimimaan, seuraava askel oli laitteen perinpohjainen testaaminen ja kehityksen jatkaminen testitulosten pohjalta. Pelkkä perustoimintojen toimivuus ei riittänyt, sillä laitteen tulisi voida toteuttaa sovitun laisia, kudontavalmiita kuvasarjoja. Ohjausohjelmistoa olisi alettava kehittämään loppukäyttäjää varten, joka tarkoittaisi selkeyttä, helppokäyttöisyyttä ja mahdollisimman monen kuvaustilanteen haasteen hoitamista loppukäyttäjän puolesta ohjelmiston puolella. Hyvin nopeasti tuli selkeäksi, että vaikka kuvien kutominen oli alun perin rajattu projektin ulkopuolelle, laitteen toimintojen riittävä optimointi tulisi vaatimaan myös tavoitellun prosessin lopputuloksen arviointia. Minun tulisi voida kutoa laitteeni tuottamat kuvasarjat käyttökelpoiseksi lopputuotteeksi.

Aloin kartoittamaan mahdollisia vaihtoehtoja kudontaohjelmaksi. Joitain näistä vaihtoehdoista oli heitelty ilmaan jo aiemmissa palavereissa, mutta valtaosa noin kymmenestä kokeilemastani ohjelmasta oli entuudestaan täysin tuntemattomia. Mukana oli sekä avoimen lähdekoodin vaihtoehtoja, että täysin kaupallisia ohjelmakokonaisuuksia. Jälleen jouduin huomaamaan, että projektini kaltainen kuvaaminen ei todellakaan ollut tyypillistä. Valtaosa löytämistäni kudontaohjelmista oli joko lähinnä maisemakuvaamiseen soveltuvia, liian hitaita tai muuten soveltumattomia tarvitsemaani valtavien kuvamäärien kutomiseen. Lopulta löysin kuitenkin yhden lupaavan vaihtoehdon: Promicra-nimisen valmistajan QuickPHOTO -ohjelman, ja tarkemmin sen erillisen, mutta pääohjelmasta riippuvaisen kuvienkudontamoduulin.

4.1 Promicra QuickPHOTO

Promicra:n QuickPHOTO-ohjelma, ja tässä tapauksessa tarkemmin sen kolmesta versiosta kevyin CAMERA versio, on kaupallinen kameramikroskoopeille tarkoitettu ohjelmisto, jonka toimintoja ovat kuvien ja videoiden ottaminen, muokkaus ja analysointi (Promicra 2024). Edes tämä ohjelma ei pystynyt käsittelemään tarvitsemiani valtavia resoluutioita, mutta sen Image Stitching -kudontamoduuli pystyi kutomaan laitteeni tuottamia kuvasarjoja, tekemään sen vieläpä nopeasti, ja tallentamaan lopputuloksen kovalevylle. Jos kudontamoduulia olisi ollut saatavilla erillisenä ilman QuickPHOTO -pääohjelmaa, olisimme pärjänneet pelkällä kudontamoduulilla.

Kudontamoduulin lopputuloksista tuli hyvin nopeasti esiin kaksi huomattavaa haastetta. Ensimmäinen oli lopullisten kudottujen kuvien valtava tiedostokoko, joka saattoi olla useita gigatavuja yksittäiselle kuvalle. Käsiteltävän materiaalin koko oli jo aiemmin huomioitu tallennuskapasiteetin kannalta, mutta tuolloin oli jäänyt huomiotta valtavien kuvatiedostojen käsittelyn vaatima välimuisti ja prosessoriteho käsittelytietokoneella, sekä tiedonsiirtonopeuksista johtuvat haasteet kuvien siirtämisessä ja avaamisessa. Tämä teki kuvien jälkikäsittelystä erittäin hidasta ja kankeaa. Kuvien kokoa saattoi jonkin verran pienentää käsittelemällä niitä kuvienkäsittelyohjelmilla, mutta joutuisin kuitenkin huomioimaan lopputuloksen valtavan koon ja siitä johtuvat tiedonsiirtohaasteet entistä enemmän jatkaessani kehitystyötä. Myös alkuperäisen kuvaustarkkuuden tiputtamista kameran puolella 12 megapikselistä esimerkiksi 4K-resoluutioon kokeiltiin, mutta pienikin tarkkuudenmenetys saattoi haastavia rakenteita sisältäviä kappaleita kuvatessa tehdä kudonnasta joko epäluotettavaa tai täysin mahdotonta.

4.2 Kehitysversio 4

Kudontaohjelman avulla löytyneet kehityskohteet johtivat ensimmäisenä kameran tarkennuksen pohtimiseen ongelmana, joka oli nyt todettava pakolliseksi ratkaista suurennuksesta riippumatta. Tämä ydinongelma, joka oltiin alun perin pyritty sivuuttamaan suurennuksen rajaamisella suurennuksiin, jotka eivät vaatisi kohdistuksen korjaamista kuvauksen aikana, oli ohjausohjelmiston neljännen version tarkoituksena ratkaista. Uuden toiminnon pohjana oli yksinkertainen ajatus: jos kohdistusta muutetaan korkeusakselia liikuttamalla, niin kohdistuksen erotus aloituspisteen ja lopetuspisteen välillä olisi se etäisyys, jolla kohdistusta pitäisi korkeusakselilla kuvauksen aikana siirtää. Laitteiston kannalta tämä ei olisi ongelma, sillä käytettävissä oli korkeusakselilla laitteen Z-kisko. Ennalta sovittujen suurennusrajoitteiden puitteissa laitteen Z-kiskon tarkkuuden uskottiin olevan riittävä.

Lisäsin ohjausohjelmiston etupaneeliin kaksi uutta numerokenttää. Näiden numerokenttien avulla käyttäjä voisi syöttää ohjelmalle tarkkuuden korjauksen Z-kiskoa pitkin. Kenttiä oli kaksi, jotta voitaisiin erikseen määritellä tarkkuuden korjaus leveys- ja pituussuunnan liikkeelle. Jakamalla tämä liike tasaisesti kunkin kuvausakselin kuvauspisteille saataisiin aikaan tarkkuuden korjausliike, joka pitäisi kuvan tarkennettuna koko kuvaussyklin ajan. Ratkaisulla olisi kuitenkin kaksi huomioitavaa rajoitetta. Ensimmäinen olisi kappaleen epätasaisuuden maksimiarvo. Tätä ei mitattu laskennallisesti, koska sille ei ollut erityistä tarvetta, mutta havainnollisesti epätasaisuuden maksimi ylittyisi, jos yksittäisen kuvan kuvausaluetta ei olisi mahdollista saada tarkaksi laidasta laitaan. Tämä tarkoittaisi kuitenkin erittäin huomattavaa vääntymää kuvattavassa kappaleessa. Näitä oli testikappaleiden joukossa muutama, mutta

nämä olivat kaikki jo silminkin nähtävästi vääntyneitä ja niitä pidettiin esikäsittelyn kannalta epäonnistuneina yksilöinä. Kappaleet, joissa esikäsittely oli onnistunut edes kohtalaisesti, ei ollut ongelmia epätasaisuuden maksimiarvon ylittymisessä.

Toinen ongelma tulisi olemaan kappaleet, joissa epätasaisuus ei olisi liian jyrkkää kuvattavaksi, mutta kappaleen esikäsittelyssä tapahtuneen komplikaation vuoksi kuvattava kappale oli kupera tai kovera. Tässä tapauksessa tarkkuudenkorjaus tulisi olla huomattavasti monimutkaisempi. Tähän en kuitenkaan aikonut käyttää kehitysaikaa, sillä nämäkin kappaleet olivat edellä mainittujen tavoin epäonnistuneen esikäsittelyn tuotoksia ja niiden onnistunut kuvaaminen ei koskaan ollut edes tavoite. Toisin kuin liian jyrkästi vääntyneet kappaleet, näitä kuperia tai koveria kappaleita olisi kuitenkin mahdollista kuvata, jos kappaleen neljännekset kuvattaisiin erikseen ja tarkkuudenkorjaus määriteltäisiin neljänneksen reunojen mukaan. Näin voitaisiin joissain tapauksissa välttyä tarpeelta uuden kappaleen esikäsittelylle.

Korjausliikkeen ohjelmointi diagrammissa oli lopulta odotettua vaivattomampaa. Uusien numerokenttien arvot vietiin kukin omaan jakolaskutoimintoonsa, joissa ne jaettiin vastaavien kiskojensa jo laskettujen kuvauspisteiden määrällä. Näin saatiin kullekin kiskolle se arvo, jolla Z-kiskoa tulisi liikuttaa aina, kun kyseinen kisko liikkuu seuraavaan kuvauspisteeseen. Nämä arvot voitiin sitten tuoda suoraan kuvaussyklin sisälle liikefunktioihin, jolloin korjausliike tapahtuisi samanaikaisesti ja samalla toiminnolla kuvaussyklin ensisijaisen liikkeen kanssa. Tämän lisäksi X-akselin paluusuuntaiselle liikkeelle oli tarpeellista lisätä kertolaskutoiminto, joka kertoi saapuvan korjausliikkeen määrän -1:llä, muuntaen sen vastaluvukseen. Näin tarkennus pysyisi oikeana X-kiskon liikkuessa edestakaisin kuvaussyklin aikana.

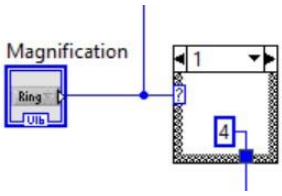
4.3 Kehitysversio 5

Neljännen version myötä tarkennusongelma oli käytännössä ratkaistu, mutta kuvausprosessin tuottamien kuvien valtava koko oli yhä huomattava ongelma. Kuvien käsittely ja siirto vei valtavasti aikaa jo pelkässä testauksessa, ja olisi huomattava rasite laitteen loppukäyttäjälle. Ratkaisu tähän löytyi huomattavasta lähestymistapamuutoksesta kehityksessä, joka jälkeenpäin katsottuna olisi pitänyt ottaa isoksi osaksi kehitystä heti alusta alkaen. Muutos ei varsinaisesti aiheuttanut mitään tarpeita jo tehdyn kehityksen hylkäämiselle tai uusimiselle, mutta sen huomioiminen viimeistään testausvaiheen alussa olisi oleellisesti nopeuttanut projektia. Tämä muutos oli kehitystyön tavoitteiden kartoittaminen asiakkaan määritelmien sijasta loppukäyttäjän prosessin pohjalta. Tätä näkökulmaa harkitessa huomattiin,

että koko kappaletta ei välttämättä tarvitsisi edes kuvata lainkaan tavoitesuurennukseksi asetetulla 20-kertaisella suurennuksella. Kaikki tarpeelliset yksityiskohdat kappaleen kartoittamiseen saataisiin näkyviin jo 5-kertaisella suurennuksella, joka vähentäisi huomattavasti kuvaussyklin kuvauspisteiden määrää ja siten lopullisen kuvan kokoa. Tarvittaessa yksittäisiä osia kappaleesta voitaisiin kuvata suuremmalla suurennuksella.

Uusi hypoteettinen kuvausprosessi edellyttäisi kuitenkin suurennusvaihtoehtojen lisäämistä ohjausohjelmistoon, ja tämä tarve oli viidennen version tavoite. Mikroskoopin objektiivia ei olisi mahdollista ohjata motoriikalla, joten sen valinta pysyisi loppukäyttäjän käsin tehtävänä. Ohjelmistoon taas olisi tarpeellista lisätä mahdollisuus valita ainakin pienin suurennus, eli 5-kertainen, sekä suurin suurennus laitteen tarkkuuden puitteissa, eli 50-kertainen. 20-kertainen suurennus olisi helppoa pitää mukana, koska työ sen lisäämiseksi oli jo tehty. Kahden muun suurennuksen lisääminen vaihtoehdoiksi alkaisi samasta paikasta kuin ensimmäisenkin, oli kokeellisesti mittailtava ja määriteltävä kuvauspisteiden pituus ja löydettävä tarpeellinen määrä kuvauspisteitä yhtä millimetriä kohti. Kun tämä oli tehty molemmille lisättäville suurennuksille, piti löytää tapa viedä nämä uudet mitat liikefunktioille ohjelman diagrammissa.

Ohjelmoinnillinen toteutus tehtiin jälleen mahdollisimman yksinkertaisesti. Kuvauspisteiden etäisyys toisistaan ja kuvauspisteiden määrä millimetriä kohti oli kovakoodattuna lukumuuttujina diagrammissa. Etupaneeliin lisättiin pieni alavetovalikko, johon sijoitettiin vaihtoehdot kolmelle halutulle suurennukselle. Tästä yhdestä alavetovalikosta saataisiin diagrammin puolelle ohjauslohkoon ohjausterminaali (KUVA 9), josta tieto valitusta suurennuksesta voitaisiin viedä haluttuihin paikkoihin diagrammissa. Aiemmin mainitut kovakoodatut lukumuuttujat ympäröitiin nyt pienillä Case-rakenteilla, joiden sisältö oli ainoastaan tuo yksittäinen muuttuja. Tämä tehtiin jokaisessa kohdassa, joissa nämä lukumuuttujat esiintyivät. Case-valinta kuhunkin rakenteeseen tuotiin suurennusvalinnan terminaalista, jolloin kullekin suurennukselle voitaisiin määritellä näille aiemmin löydetyt muuttujat. Nyt ohjelmaa ajaessa noiksi lukumuuttujiksi sijoitettaisiin Case-rakenteen avulla oikeat arvot, ja ohjelma mukautuisi valittuun suurennukseen.



KUVA 9. Suurennuksenohjaus viidennen version ohjauslohkossa

4.4 Kehitysversio 6

Viidennen version jäljiltä ohjelma oli käytännössä alkuperäisen määrittelyn mukaan valmis. Kuvaussykli toimi moitteetta kaikilla kolmella suurennuksella, ja tarkennuksenkorjaus mahdollisti epätasaistenkin kappaleiden kuvaamisen ilman ongelmia. Kuvasarjoista kudotut kuvat olivat yhä suuria, mutta jo paljon helpompia käsitellä ja laadultaan moitteettomia tarkoitukseensa. Asiakkaalla oli kuitenkin ollut toiveena lisäominaisuus, jossa kuvausalue voitaisiin määritellä kappaleen mittaamisen sijaan suoraan kuvausaluetta esikatsellessa kameraan liitetyllä näytöllä. Heidän ajatuksensa oli, että käyttäjä voisi valita suunnikkaan muotoisen kuvausalueen kulmat, joiden sisään jäävä alue kuvattaisiin. Minulla oli muun kehityksen aikana syntynyt tämän toiveen toteuttamiseksi idea, joten koin järkeväksi toteuttaa sen vielä lopulliseen versioon ennen käyttöönottoa.

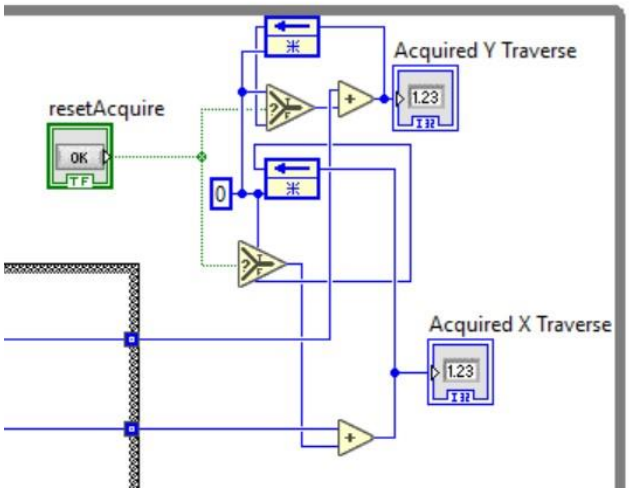
Suunnitelmani toteutukselle oli käyttää jo aikaisemmin toteuttamaani käsiohjausta. Ikävä kyllä liikkeen toteuttava laite ei tukenut ominaisuutta, jolla olisin voinut määritellä minkäänlaisia koordinaatteja valituille kuvauspisteille. Laitteella oli kyllä ominaisuus ilmoittaa tieto kunkin kiskon jäljellä olevista moottorin askeleista, mutta tämä oli osoittautunut epäluotettavaksi jo laitetta alun perin tutkiessa, enkä halunnut tehdä ominaisuudesta riippuvaista sellaisesta tekijästä, jota ei itse voitu hallita. Sen sijaan päättelin, että samaan lopputulokseen päästäisiin, jos pisteiden määrittely tehtäisiin käsiohjauksella ja käsiohjauksen kulkema etäisyys olisi mahdollista kerätä talteen. Tämä etäisyys voitaisiin sitten muuntaa laskennallisesti kuvausalueen kattaviksi kuvauspisteiksi ja toteuttaa kuvaussykli.

Ohjelmoinnillisesti tämä oli hieman aiempia toimintoja haastavampaa toteuttaa. Koko toiminnon ytimenä olisi LabVIEW-funktio, jota olin käyttänyt aiemmin ensimmäisen version simuloinnissa, nimeltään Feedback Node, eli Palautenoodi. Normaalisti LabVIEW ei muistinhallinnallisista syistä säilytä mitään tietoa ohjelmasyklien välillä. Palautenoodi on yksi tapa, jolla tietoa voidaan säilyttää syklistä toiseen. Noodin toimintaperiaate on, että se lähettää saamansa tiedon seuraavan syklin alussa eteenpäin. Jos noodi laitetaan lähettämään tieto takaisin itselleen, pysyy tieto tallessa jatkuvasti ohjelman käydessä. Noodille kulkevaa tietovirtaa voidaan sitten manipuloida, esimerkiksi laskufunktioilla, ja muuttaa näin noodissa olevaa arvoa.

Palautenoodeja tarvittiin kaksi: Yksi X-kiskolle ja toinen Y-kiskolle. Ensin palautenoodit alustetaan yhdellä muuttujalla kokonaisluvuksi 0. Palautenoodien lähdöt viedään ensimmäiseksi LabVIEW’n Select-funktioon, joka oli funktio, jota en ollut aikaisemmin käyttänyt. Selectin tarkoitus on valita yksi kahdesta tulosta ohjattavaksi yhteen lähtöönsä sillä perusteella, tuleeko funktion kolmanteen tuloon

TRUE vai FALSE, eli tosi vai epätosi. Select-funktion sijoitin tähän siltä varalta, jos loppukäyttäjä haluaa nollata tallennetut arvot. Funktion kolmas tulo, eli valinta, kytketään suoraan etupaneeliin sijoitettavaan Reset-painikkeeseen, jota painamalla Selectin valintatuloon saapuu TRUE-arvo. Tässä tapauksessa Select ohjaa takaisin palautenoodille pelkän nollan. Muissa tapauksissa palautenoodilta tullut arvo jatkaa matkaa katkeamatta Selectin läpi.

Selectiltä tuleva arvo vietiin vielä ennen palautenoodiin palauttamista yhteenlaskufunktioon. Tässä funktiossa palautenoodissa aiemmin olleeseen arvoon lisätään käsiohjauksen liikerakenteesta tuleva arvo, joka koostuu itse liikefunktiolle syötetystä etäisyydestä. Tämä arvo palautetaan sitten palautenoodille sekä lähetetään etupaneelissa oleviin numerokenttiin, jotta loppukäyttäjä voi seurata kuvausalueen mittojen kasvua reaaliajassa. Palautenoodien silmukkarakenteet on havainnoitu kuvassa 10. Näistä numerokentistä tehtiin myös paikallismuuttujat, joita voitaisiin puolestaan käyttää tuloina kuvaussykliä varten, mutta ensin ne tulisi jakaa arvolla, joka vastasi kunkin suurennuksen kuvausvälien etäisyyttä. Nämä arvot olivat kehitysversion 5 tapaan Case-rakenteissa, joita ohjattiin ohjelman suurennusvalikolla. Näistä syntyvät kuvattavien kuvauspisteiden määrät pyöristettiin ylöspäin, koska kuvauspisteiden etäisyyksien murto-osia ei ollut käytännöllistä toteuttaa ja alaspäin pyöristäminen olisi monissa tapauksissa kuvannut haluttua pienempiä kuvausalueita.



KUVA 10. Käsiohjauksen kulkeman etäisyyden talteenotto

Toiminnolle oli tehtävä kokonaan oma versionsa käsiohjausrakenteesta, mutta tämä oli vaivatonta, sillä LabVIEW’ssa kokonaisia Case-vaihtoehtoja voi suoraan monistaa. Nämä kaksi käsiohjausrakennetta eivät lopulta poikenneet toisistaan erityisen paljoa. Alun perin palautenoodien silmukat olivat uuden käsiohjausrakenteen sisällä, mutta tämä aiheutti numerokentissä päivitysongelmia, koska noodit eivät päässeet pyörähtämään kuin silloin, kun käsiohjausrakenne ajettiin. Tämä tarkoitti myös sitä, että

käyttäjän näkemä arvo olisi aina yhden päivityksen myöhässä ja täten epätarkka. Ongelma ratkesi helposti siirtämällä silmukat ulos pääohjelmarakenteeseen, jossa ohjelman pääsilmukka jatkuvasti päivitti käyttäjälle uusimmat arvot noodeista. Tämän jälkeen uuteen käsiohjausrakenteeseen ei jäänyt alkuperäiseen verrattuna muita eroja, kuin uudet noodeille menevät lähdöt.

Myös erillinen kuvaussyklirakenne oli tehtävä toimintoa varten. Jälleen pohjana toimi jo valmiin, aiemman kuvaussyklin Case-rakenne. Uusi rakenne otti silmukoidensa iteraatioiden määrät palautenoodeilta tulevien, esikäsiteltyjen kuvauspistemäärien mukaisesti. Samalla koko kuvaussyklin suunta käännettiin päinvastaiseksi, sillä kuvauksen lähtöpiste olisi se piste, mihin toinen kuvaussykli yleensä päättyisi. Tämä johtui siitä, että kuvausaluetta valitessa loppukäyttäjän tulisi viedä kuvausnäkymä kuvauksen alkupisteestä kuvauksen loppupisteeseen saadakseen haluamansa kuvausalue mitatuksi. Tarvetta palata alkupisteeseen ei ollut: kuvaussykli voitaisiin ajaa käänteisesti kääntämällä kuvaussykli-rakenteen sisällä kaikki etäisyydet vastaluvuikseen.

Näiden toimien jälkeen uusi kuvaussykli toimi moitteetta, ja oli tarpeellinen ja tervetullut työkalu etenkin suuremmilla suurennuksilla, kun haluttiin yksilöidä joku pieni alue kappaleesta kuvattavaksi erikseen. Tarvetta mittaamiselle tai arvioimiselle ei ollut, käyttäjän tarvitsi vain valita aloituspiste laittamalla kuvausalueenkaappaus-moodi päälle, liikkua toivottuun lopetuspisteeseen ja aloittaa kuvaus painamalla painiketta ohjelman etupaneelissa. Sekä minä että asiakas olimme tässä vaiheessa tyytyväisiä ohjelmaan ja sen tuottamiin lopputuloksiin, joten muut mahdolliset tulevaisuuden toiminnot jätettiin mahdollisen jatkokehityksen aiheiksi. Kehitysversio 6 tulisi pienen kosmeettisen asettelun jälkeen olemaan ohjelman valmis käyttöversio. Jäljelle jäi enää käyttöönotto kuvauslaboratoriossa ja loppukäyttäjälle tarkoitettu käyttöopas.

4.5 Käyttöönotto

Käyttöönotto sujui ongelmitta. Itse laite oli jo ollut sijoitettuna kuvauslaboratorioon useamman viikon ajan, joten sen asennus paikoilleen oli enää pientä kaapeleiden siistimistä. Käyttöönoton kannalta oleellisin osuus oli ohjausohjelmiston asennus laboratorion kuvaustietokoneelle, jolla laitetta tultaisiin jatkossa ohjaamaan. Tätä varten Kehitysversio 6 piti ensin LabVIEW’ssa kääntää ajotiedostoksi. LabVIEW’ssa on kaikkien hyvien kehitysympäristöjen tavoin helpot työkalut tähän ja ajotiedoston saa tehtyä käytännössä yhdellä painalluksella. Tässä vaiheessa annoin lopulliselle ohjelmalle nimen AutoImage. Ajotiedoston pystyi sellaisenaan siirtää kuvaustietokoneelle, mutta kuvaustietokoneelle oli

myös asennettava LabVIEW'n oma suorituspalvelu, josta LabVIEW ajotiedostot ovat riippuvaisia. Tämän jälkeen AutoImage toimi normaalisti ja laite ohjelmistoineen oli asennettu ja käyttövalmis.

Lopuksi kirjoitin vielä laitteelle ja ohjelmalle käyttöohjeet, joissa pyrin mahdollisimman yksityiskohtaisesti kuvaamaan laitteen ja ohjelman, sekä niiden vaatimat esivalmistelut, jos käyttäjä olisi esimerkiksi tilanteessa, jossa laitetta ei ole lainkaan asennettu. Tämän lisäksi käyttöohjeessa kuvattiin kaikki ohjelman etupaneelista löytyvät toiminnot ja opastettiin yksityiskohtaisesti kuvaussyklien käytössä. Asiakkaan yritys on vahvasti kaksikielinen, joten käänsin käyttöohjeet myös englanniksi. Lopulta muunnin alun perin Microsoft Wordillä tehdyt dokumentit helppokäyttöisempään PDF-muotoon ja ohjasin ne arkistoitavaksi asiakkaan verkkolevylle. Kokonaisuus tultaisiin myös koeajamaan ensimmäistä kertaa vain paria viikkoa myöhemmin, kun minun poissa ollessani kuvauslaitteiston ensisijainen käyttäjä asiakasyrityksessä käytti onnistuneesti laitetta tekemieni käyttöohjeiden pohjalta.

Laitetta käytettäessä oli huomattu kaksi vikatilannetta, joita ei ollut saatu ratkaistua. Ensimmäinen näistä liittyi COM-portin yhteyteen, joka joillain ohjausohjelman käynnistyskerroilla ei näyttänyt reagoivan. Todennäköisesti tämä johtui siitä, että kehitystyötä oli ensisijaisesti tehty kannettavalla tietokoneella, joka oli toistuvasti kiinnitetty ja irrotettu laitteesta, jolloin yhteyden alustus ei välttämättä toiminut oikein. Ongelmaa ei esiintynyt varsinaisella laboratorion kuvauskoneella, ja tapahtuessaan ongelman pystyi ratkaista käynnistämällä tietokone uudelleen. Vaikka vian lopulliseen korjaamiseen saatetaan palata tulevaisuudessa, käyttöönotettavan version osalta tähän ei käytetty aikaa. Ongelma dokumentoitiin käyttöohjeeseen loppukäyttäjälle tiedoksi.

Toinen ongelma liittyi tilanteisiin, jossa kameran laukaisu ei joskus onnistunut ja kamera ohitti kuvauspisteen. Aluksi tätä yritettiin korjata tekemällä ohjelmaan tarkistus, joka ei anna silmukan edetä, ennen kuin Holmarcin ajuri palauttaa vahvistuksen, että käsky on suoritettu. Tämä ei kuitenkaan näyttänyt ongelmaan vaikuttavan. Pohdimme asiaa asiakkaan kanssa, ja teoriamme oli, että koska laukaisuvirhe on niin johdonmukaista, tapahtuen noin joka sadannella laukaisulla, kyse oli todennäköisesti kameran tallennusnopeudesta kiinni. Oli huomioitava, että kamera yritti tallentaa satojen megatavujen edestä kuvia muutamassa minuutissa, ja tämä saattoi olla liikaa joko kameran tai käytettävän USB-massamuistilaitteen tiedonsiirtonopeuksille. Uusimman kuvausprosessin myötä tyypillinen kuvasarja oli kuitenkin pienentynyt noin 50 kuvan tienoille, emmekä asiakkaan kanssa kokenut kriittiseksi korjata puutetta ennen käyttöönottoa. Tämäkin ongelma dokumentoitiin käyttöohjeeseen tulevan varalle.

5 POHDINTA

Jo heti ensimmäisestä tapaamisesta asti tämä projekti tuntui opinnäytteeltä. En ollut koskaan ennen toteuttanut asiakkaan kuvailemaa työaseman automaatiota, mutta opintojeni ansiosta minulla oli vahva käsitys siitä, miten heidän toiveensa voisi toteuttaa, sekä vahva itsetunto sen suhteen, että minulla oli tarvittava osaaminen sen realisoimiseksi. Etenkin opinnot sulautettujen järjestelmien parissa olivat antaneet minulle tarvittavat valmiudet tuolloin vielä täysin teoreettisen kuvaustason ohjaamiseen, sillä ymmärsin laitteiden ja ohjelmien välisen tietoliikenteen periaatteen. Olin saanut opinnoistani vahvan yleiskäsityksen ohjelmoinnista, ja tiesin voivani toteuttaa ohjelmiston riippumatta siitä, millä ohjelmointikielellä tuo ohjelma tulisi toteuttaa.

Tämä itsevarmuus tuli nopeasti tarpeeseen, sillä hyvin nopeasti huomasin yhden teollisuuden tosiasian, josta en aiemmin ollut tietoinen. Kun aloin rakentamaan projektilleni vahvaa teoriapohjaa, oletin löytäväni helposti monia esimerkkejä projektini kaltaisista laiteratkaisuista. Toisin kuitenkin kävi. Oli erittäin vaikeaa löytää edes perustason tietoa halutunlaisesta laitekokonaisuudesta, tai edes sen yksittäisistä osista. On tietysti täysin ymmärrettävää, että tyypillisesti yksityissektorilla olevat teollisuuden yritykset eivät halua jakaa tietoja työkaluistaan ja metodeistaan julkiseen käyttöön. Olin kuitenkin yllättynyt, että jouduin käytännössä tekemään kaiken oman alustustyöni puhtaasti kokeellisesti, tai korkeintaan laitteista saatavilla olevien vähien teknisten tietojen perusteella. Vastaavanlaisista tapauksista löytyi kyllä tietoa taideharrastajien puolelta sekä muutamista akateemisista lähteistä, mutta ikävä kyllä jouduin toteamaan heidän laitteidensa tarkoitukset liian erilaisiksi ja heidän metodinsa omiin tarkoituksiini sopimattomiksi.

Pakostakin tämä saa miettimään, miten paljon turhaa työtä yksityinen sektori tekee jokainen päivä ratkaistessaan asioita, jotka on jo ratkaistu satoja kertoja ympäri maailmaa. Tämä on täysin ymmärrettävää kilpailuedun kannalta, mutta en voi olla ihmettelemättä, miten paljon pitemmällä globaali teknologian taso voisi olla, jos yksityinen teknologiateollisuus seuraisi akateemisessa maailmassa harjoitettavaa tiedon ja osaamisen jakamista. Positiivinen puoli tässä oli lähinnä henkilökohtainen, sillä koin mahdollisuuden selvittää projektini asiat itse erittäin palkitsevina. Ilman teknologiateollisuuden tiukkahuulisuutta projektistani ei välttämättä olisi ollut minulle edes opinnäytetyöksi asti, jos olisin voinut vain etsiä pari soveltuvaa esimerkkiä vastaavanlaisesta kuvaamisesta ja toteuttaa niiden pohjalta oman versioni. Toisaalta tuolloin työ olisi keskittynyt enemmän viemään ratkaisua yhä pidemmälle, kuin mitä aikaisemmin oli tehty, kuten akatemian puolella tehdään.

Onnistuin vaihtoehtojen kartoittamisessa projektin vaatimaksi kuvaustasoksi mielestäni hyvin. Ilman riittävän korkealaatuista kuvaustasoa projektin onnistuminen ei olisi ollut mahdollista, ja varteenotettavat vaihtoehdot maksoivat kaikki nelinumeroisia summia, joten huolellisuus oli tarpeellista. Valituksi tullut laite oli täpärästi tarpeeksi tarkka kaikkiin siltä edellytettyihin toimintoihin, ja oli samalla kilpailijoitaan huomattavasti edullisempi. Laitteen ohjauksessa oli omat haasteensa sen rajallisten ominaisuuksien vuoksi, mutta tämä oli paikattavissa omalla osaamisella ohjausohjelmistoa kehitettäessä. Projektin lopussa sekä minä että asiakas olimme tyytyväisiä laitteeseen, ja totesimme sen suurimmaksi ongelmaksi olleen pitkä toimitusaika.

Myös ohjelmistokehitys sujui erittäin hyvin ja olen päätöksiini tyytyväinen. Alkuun saattoi tuntua ehkä hieman riskialttiilta valita ohjelmointikieleksi kieli, josta minulla ei ollut mitään aiempaa kokemusta. Olen kuitenkin opintojeni myötä todennut voivani helposti oppia uusia ohjelmointikieliä vahvan ohjelmointiperustani pohjalta, joten en ollut huolissani kielen oppimisesta. Tämän lisäksi asiakkaalla oli tarjota tarvittaessa kielen kanssa tukea yrityksen oman osaamisen vuoksi, ja samalla pääsin opettelemaan ja kehittämään kokemusta kielen parissa, joka on alalla laajassa käytössä. Kielen opettelu olisi aiheuttanut oman ajallisen rasitteensa projektille, mutta tämä ei käytännössä toteutunut, sillä opettelu tapahtui kuvaustason pitkän toimitusajan aikana.

Kehitystyön aikana huomasin keksiväni pyörän uudestaan myös kehitysmetodien osalta. Alkuun tekemiseni vaikutti puhtaalta Waterfall-, eli vesiputoustyyliltä: Aikomus oli ensin suunnitella, sitten kehittää ja lopuksi testata. Kuitenkin nopeasti huomasin, että voidakseni kehittää tehokkaammin, tarvitsin enemmän tietoa jo tehdyn työn onnistuneisuudesta. Oli välissä testattava, tarvittaessa muutettava suunnitelmia löydösten pohjalta ja sitten palata kehitykseen. Tämä taas muistutti huomattavasti nykyään Agilena, eri ketteränä kehityksenä tunnettua metodiikkaa. Silti vahvasti päämääräkeskeinen lähestymistapa piti joitain vesiputousmallin piirteitä mukana, ja lopputulosta olisi voinut jo kuvata joskus Wagilena tunnettuna kehitystyylinä, joka on yhdistelmä vesiputousta ja ketterää kehitystä. Olin mielenkiintoista huomata, että kun sain itse päättää siitä, miten kehitystyöni teen, päädyin itsenäisesti samoihin johtopäätöksiin, kuin ohjelmistokehityksen valtavirta.

Tätä oman tekemiseni ohjaamista ja optimointia voisi verrata Agile-metodiikasta tunnettuun käsitteeseen itseohjautuvista työryhmistä. Näissä periaatteena on, että turha johtaminen tulee työnteon tielle ja vain hidastaa kehitystä, kun enemmän itsenäiset ryhmät järjestelevät itse työnsä siten, miten niitä on kaikista helpointa ja tehokkainta tehdä. Itse huomasin tässä itseohjautuvuudessa kuitenkin jonkinlaista puutteellisuutta. Vaikka opinnäytetyön kannalta oli erittäin hyödyllistä, että sain selvittää kaiken itse ja

pyytää apua ainoastaan silloin, kun muuta vaihtoehtoa ei ollut, oli projektin aikana monia vaiheita, jotka olisivat olleet huomattavasti nopeampia ja tehokkaampina, jos joku olisi ollut seuraamassa työtä ja ohjannut tarvittaessa projektille tukea, vaikka en itse ollut sitä vielä päätynyt harkitsemaan. Otaksun, että tämänlainen itseohjautuvuuden puute voitaisiin paikata kokemuksella, mutta mielestäni ilmiö on silti tärkeä huomioida, jos jokin työryhmä olisi kokonaisuudessaan kokematon, kuten on monesti tapana esimerkiksi startupeissa.

LÄHTEET

National Instruments 2024. *LabVIEW User Manual.* Internet-sivu. Saatavissa: https://www.ni.com/docs/en-US/bundle/labview/page/user-manual-welcome.html. Viitattu 3.5.2024.

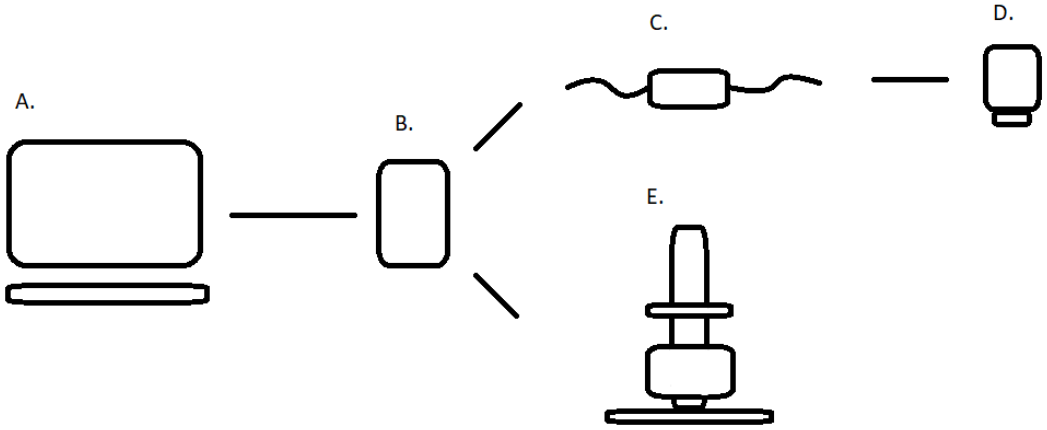
Promicra 2024. *QuickPHOTO CAMERA.* Internet-sivu. Saatavissa: https://promicra.com/quickphoto-microscope-software/quickphoto-camera/. Viitattu 14.4.2024.

Zaber 2024. *Troubleshooting.* Internet-sivu. Saatavissa: https://www.zaber.com/w/Troubleshooting. Viitattu 20.4.2024.

AutoImage User Manual

INTRODUCTION

This document is to be used as a manual in the operation of the automated imaging station. Familiarize yourself with these instructions with care before operating the station. The imaging assembly is designed to be used only with the Leica Flexacam C1 microscope camera. Compatibility with other similar cameras is unknown.
The station consists of three individual units. In addition, a computer is required for running the program and a Leica DM1750 microscope with a Leica Flexacam C1 camera is required to complete the assembly.



A) Computer
B) Holmarc Micromotion Driver
C) Triggering Circuit
D) Leica Flexacam C1
E) Holmarc HO-MEPS 5020M Motorized XYZ-taso

The camera itself requires it's own peripherals: a USB-based storage solution for storing the images, and a display for previewing and viewfinding. If the camera's settings need to be changed, a keyboard and mouse is also required.
In case of undesired operation, switch off the power from the Holmarc Micromotion Driver using the red switch in the front panel.

PREPARATION

Before imaging, the following preparations must be observed.

Computer

The station is controlled with the AutoImage program, which must be available on the computer. Note that the program will require installation of the Labview Runtime Engine to run. Additionally, the USB-Serial adapter used for interfacing with the Holmarc driver, requires it's own driver software. The adapter will attempt to install it's driver when connected.
Once all software is found on the computer, it can be connected to the Holmarc driver with the USB-Serial cable.

Holmarc Micromotion Driver

The driver's cables must be connected before operation. These include the axial cables, which connect to the motors of the motorized stage, and the power cable. Additionally, the black cable must be connected to the Trigger OUT port and it's other 3,5mm end connected to the triggering circuit.

Triggering Circuit

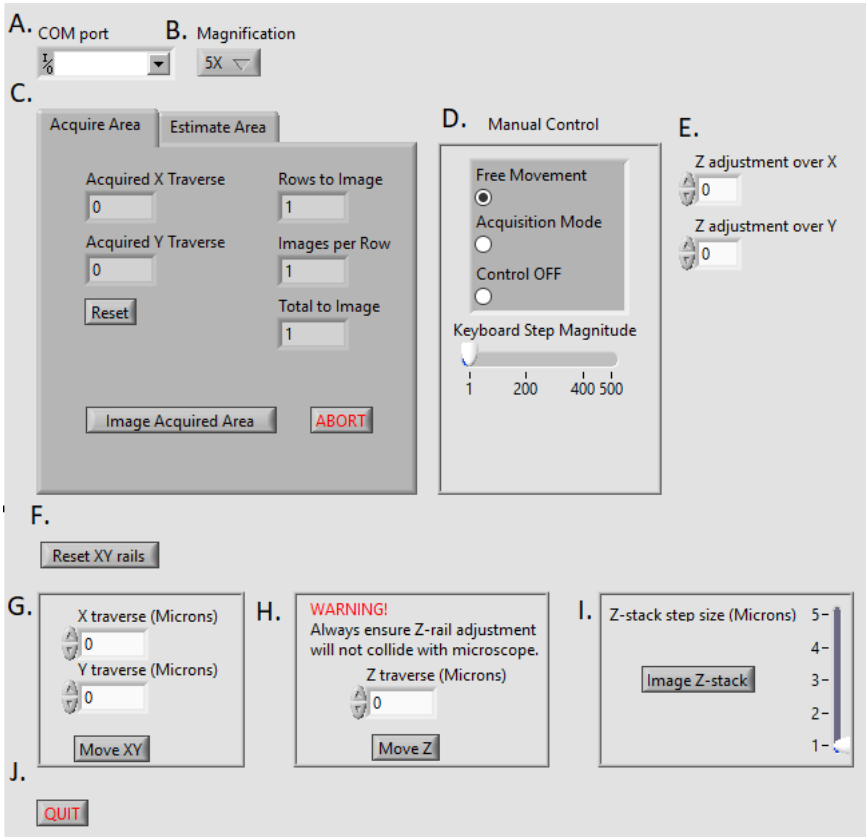
The triggering circuit has a 5V power adapter, which must be plugged in to an electrical outlet. The 2,5mm line must be connected to the switch port on the Leica Flexacam C1 camera.

DM1750 -microscope

The microscope's lights on the user's right hand side must be turned on. The objective must be chosen manually. Remember your chosen objective for the imaging program. The attached camera must also be switched on separately. A storage solution must be connected to the high speed USB port on the camera itself. The display used for the camera must be appropriately connected and turned on. If the settings of the camera need to be adjusted, a mouse and keyboard must be connected to the camera with a USB hub.
AutoImage version 1.0 supports the 5X, 20X and 50X objectives.

AUTOIMAGE

The automated imaging station is controlled by the AutoImage LabVIEW program.



AutoImage front panel.

A) COM port selection
B) Magnification
C) Imaging Cycle selection
D) Manual Control modes
E) Focus adjustment for the imaging cycles
F) Reset XY rails
G) Direct input for XY rails
H) Direct input for Z rail
I) Z-stack imaging
J) End program

COM port selection

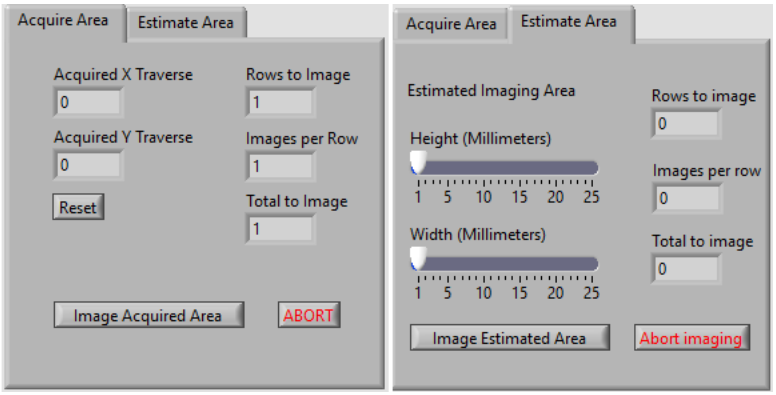
Select the COM port corresponding to the Holmarc Micromotion Driver.

Magnification

Set the desired magnification. This must be equal to the objective set on the microscope. If you change objectives, remember to also change the magnification setting.
Version 1.0 supports 5X, 20X and 50X magnification.

Imaging Cycle selection

Select by switching tabs either Acquire Area or Estimate Area. Acquire Area is best suited for faster and more convenient acquisition of imaging area for smaller specimens without need for specimen dimensions. Estimate Area is better suited for setting an imaging area for larger specimens when the specimen dimensions are known.



Imaging cycles. Acquired Area on the left, Estimate Area on the right.

Manual Control

Manual Control is divided into 3 modes, from top to bottom: Free Movement, Acquisition Mode, and Off. The sliding bar under the mode select is used to set the distance of a single motion for each key press. The XY rails respond to the arrow keys on the keyboard. Additionally, the Z rail responds to the PageUp and PageDown keys.

Focus adjustment

If the specimen is uneven, a Z adjustment can be set. The Z rail will attempt to compensate by an elevation equal to the input value over the full traverse of the X and Y rails respectively. Units of input correspond with those of the manual control.

Reset XY rails

This button will return the XY rails to their respective zero positions.

Direct input for XY rails

If the motorized stage must be moved more significantly, such as during initial positioning, it may be more practical to use this option to directly input the desired amount of traverse for the XY rails. When moving the rails towards their zero positions, negative values are used.

Direct input for Z rail

WARNING! THE RAIL DOES NOT STOP AUTOMATICALLY WHEN THE IMAGING STAGE OR A SPECIMEN UPON IT CONTACTS THE MICROSCOPE OBJECTIVE. USE WITH CAUTION.
The Z rail can be moved directly in the same was as he XY rails. Positive values will lower the stage. Negative values will raise the stage.

Z-stack imaging

Z-stack imaging can be used to take a stack of layered images from a single point for focus stacking applications. The imaging must be started from the highest point in regards to the Z rail. The imaging cycle will move the stage down the desired amount between each image.

End program

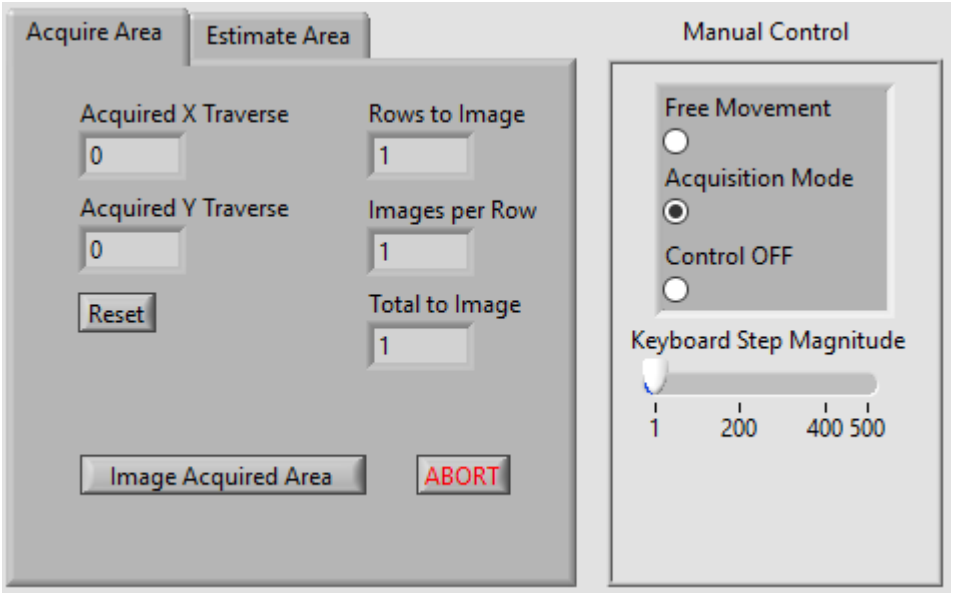
Close the AutoImage program.

IMAGE CYCLES WITH AUTOIMAGE

Imaging Cycles can be used with manual area acquisiton or estimated area. Additionally a separate imaging cycle is available for Z-stack imaging.

Acquire Area imaging cycle

Before designating an imaging area, the view must be placed on the left top corner of the specimen. To guarantee good cycle quality, the starting point should only show the specimen in the bottom right quarter. Use the Z rail to focus the image. Note, that the edges of a specimen can often be bent, and the focusing is best done on a point on the specimen slightly further in from the actual edge.
Once the starting point has been reached and the image is in focus, open the **Acquire Area** tab and set manual control to **Acquisition Mode**, as shown below.

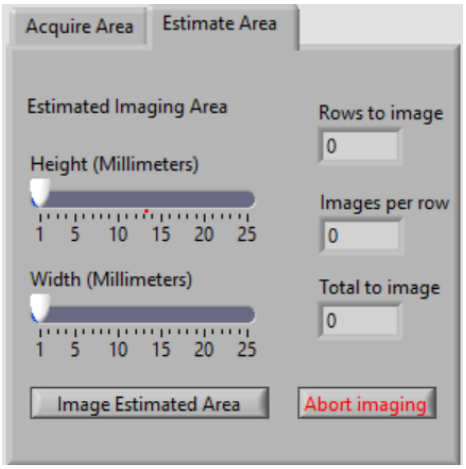


If the **Acquired Traverse** fields are not at zero, first click reset. You can also click reset if you want to start over with acquisition.
While manual control is in acquisition mode, your movement in manual control is stored in the program. You can see your total acquired traverse and the formulated imaging cycle preview in the Acquire Area window.
To designate your imaging area, while in Acquisition Mode, use manual control to move the view from the top left to the top right of the specimen. Again for good cycle quality, it is best if the specimen remains in only one quarter of the view at the furthest point. If the specimen is uneven and the image has gone out of focus, refocus the view using the Z rail. Acquisition Mode does not record the motion of the Z-rail, so input the amount of Z-rail adjustment in the **Z adjustment over X** field.
Now move the view down to the bottom right corner of the specimen. . Again for good cycle quality, it is best if the specimen remains in only one quarter of the view at the furthest point. If the specimen is uneven and the image has gone out of focus, refocus the view using the Z rail. Acquisition Mode does not record the motion of the Z-rail, so input the amount of Z-rail adjustment in the **Z adjustment over Y** field.

You are now ready to image. Begin the cycle by clicking **Image Acquired Area**. If you want to end the cycle prematurely, click on **Abort**.

Estimated Area imaging cycle

You can designate the imaging area directly in millimeters. Note that if the specimen is uneven, you will also have to estimate or manually find the necessary Z-rail adjustment.
Voit määritellä kuvausalueen myös syöttämällä suoraan haluamasi kuvausalueen millimetreinä. Huomaa, että jos kappale ei ole tarpeeksi tasainen, joudut myös arvioimaan tai käsin selvittämään tarvittavan Z-kiskon kompensoinnin.
Before designating an imaging area, the view must be placed on the left top corner of the specimen. To guarantee good cycle quality, the starting point should only show the specimen in the bottom right quarter. Use the Z rail to focus the image. Note, that the edges of a specimen can often be bent, and the focusing is best done on a point on the specimen slightly further in from the actual edge.
Open the **Estimate Area** tab, as shown below.



Set the desired imaging area in millimeters. Note the **Total to image** field, as image payloads over 100 images can be slow to stitch and the resulting image can be exceedingly large in file size and difficult to handle.
If there is need for Z rail adjustment during the cycle, set the desired amount of adjustment into the **Z-adjustment over X** and **Z-adjustment over Y** fields.
You are now ready to image. Begin imaging by clicking **Image Estimated Area**. If you want to end the cycle prematurely, click on **Abort**.

Z-stack

To image a Z-stack, first find your desired single point on the specimen. Set the focus on the Z rail in such a way, that the starting position is at the highest position of the imaging cycle. Only the furthest points on the specimen need to be in focus in the first image.
Set your desired Z-stack interval from 1-5 micrometers. The cycle will image your chosen point 20 times, lowering the stage by the designated interval for each image.
Begin imaging the stack by clicking **Image Z-stack**.

KNOWN ISSUES

- Occasionally the camera can fail to trigger.
- In some cases the driver can stop responding to the program even when correctly connected. Restarting the computer fixes the issue.

Technical documentation for Sasken Automated Imaging Station

The assembly and use of the station is documented in the Automated Imaging Station manual.

List of components

- Holmarc 3-axis rail system
- Holmarc controller unit
- Sasken Triggering Circuit
- Sasken AutoImage control program
- Leica Flexicam C1 Camera
- Leica DM1750 materials microscope

Holmarc 3-axis rail system

A free-standing rail system with 3 axis and a stepper motor driving each axis. Stepper motors are connected to the controller unit with 3 separate coaxial cables. Notable characteristics of the system as provided by the manufacturer are as follows:

- Range of motion 50mm
- Accuracy 5um
- Resolution 625nm

Holmarc controller unit

The controller unit is used to translate program commands into control of the rail system. The front panel features 2 coaxial ports for Trigger IN and Trigger OUT, a small reset button and a power switch. The back panel features coaxial ports for connecting the 3 rail motors, an RS-232 port for connecting to the control program and a power cable. The serial communication protocol used with the controller unit is documented in the separate document Command Code.

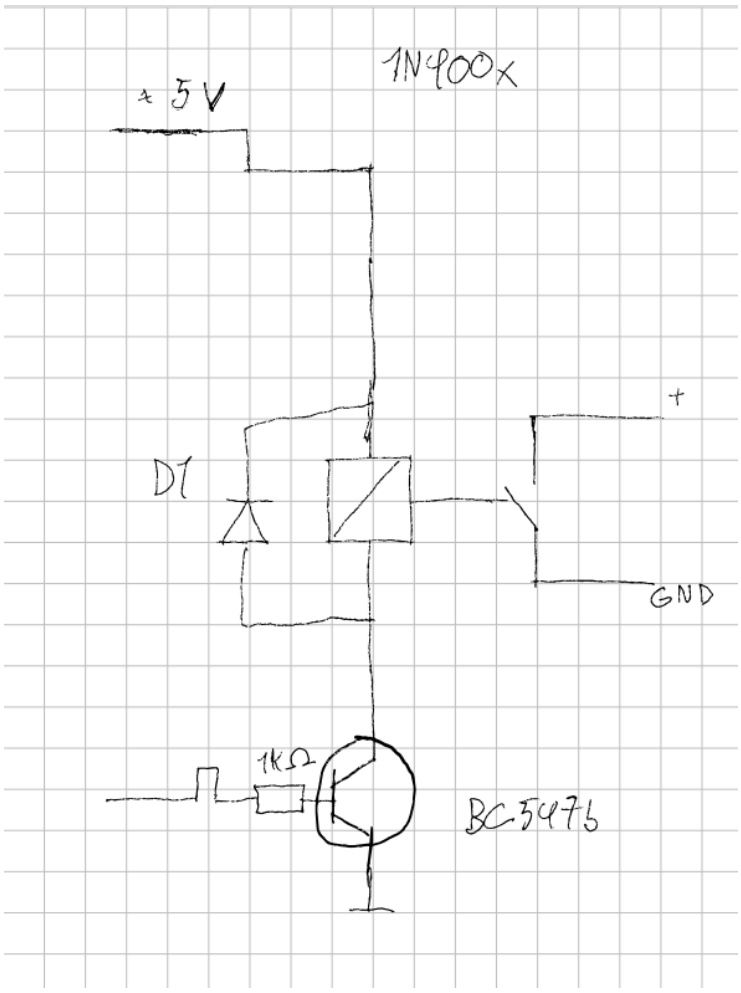
Sasken Triggering Circuit

A simple circuit where a transistor controls the ground to a relay. Once the relay has power, the relay connects the trigger pins, triggering the microscope camera. Power to the relay is provided by an attached 5V power supply. Power to the transistor gate is provided by 5V pulse from the Holmarc controller unit.

List of triggering circuit components:

- Cynergy3 S1-0504DM
- Motorola BC547B
- MCC 1N4001
- 1000 ohm resistor
- 5V power supply

See image below for diagram.



Sasken AutoImage program

A LabVIEW application. Can be used on most computers with a USB port to connect to the Holmarc control unit but requires the LabVIEW Runtime Engine to be installed. The program follows a simple handler-function structure, where events in the front panel are translated into functions to be run. The software uses modified versions of Holmarc's LabVIEW drivers. Most functions consist of sending distance or trigger commands to the controller unit using these drivers. Imaging cycles are created with nested For-loops.

Leica Flexicam C1 Camera

A camera integrated into the Leica microscope. Refer to the camera's manual for documentation.

Leica DM1750 materials microscope

A typical top-down vertical microscope with 5 objectives. Refer to the microscope's manual for documentation.