SeAMK
SEINÄJOEN AMMATTIKORKEAKOULU
SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Väinö Koski

Sovellus työstökoneiden työaikojen mittaamiseen

Opinnäytetyö
Kevät 2024
Insinööri AMK, Automaatiotekniikka

SEINÄJOEN AMMATTIKORKEAKOULU

Opinnäytetyön tiivistelmä

Tutkinto-ohjelma: Insinööri (AMK) Automaatiotekniikka

Suuntautumisvaihtoehto: Sähköautomaatiotekniikka

Tekijä: Väinö Koski

Työn nimi alaotsikoineen: Sovellus työstökoneiden työaikojen mittaamiseen

Ohjaaja: Niko Ristimäki

Vuosi:2024 Sivumäärä:41

Opinnäytetyö toteutettiin yritykselle, joka halusi kerätä työstökoneista aikatietoja. Yritys toimii uudisrakentamisen alalla. Työn tavoitteena oli tehdä sovellus, jolla pystyttäisiin lukemaan yrityksen TwinCAT-pohjaisista työstökoneista työstöaikoja. Kerätyn datan perusteella pystyttäisiin tuotannossa tekemään tarvittavia muutoksia ja saamaan maksimaalinen hyöty materiaalivirrasta.

Työssä vertailtiin monia eri ohjelmistoja ja käyttöjärjestelmiä, joiden perusteella valittiin paras mahdollinen ohjelmisto ja käyttöjärjestelmäympäristö. Koodi, jolla luettiin työstökoneita, selitettiin kuvilla. Testiversio tehtiin erilliselle tietokoneelle, jossa sitä testattiin ja ajettiin. Testiversio tehtiin tietokoneelle, jossa oli käyttöjärjestelmänä Linux. Tuloksia vertailemalla huomattiin, että aikatietojen lukeminen työstökoneilta onnistui. Sovelluksen toteutuskielenä toimi Python.

Lopputuloksena yritys sai selkeämmän käsityksen, kauanko koneet ovat päivässä päällä ja onko koneessa ollut ongelmia. Yritys sai myös tietoa, miten ohjelmistoon voitaisiin lisätä muita toimintoja.

[1]Asiasanat: sovellus, työstökone, koodi, Linux, Python

SEINÄJOKI UNIVERSITY OF APPLIED SCIENCES

Thesis abstract

Degree programme: Bachelor of Engineering, Automation Engineering

Specialisation: Electric Automation

Author: Väinö Koski

Title of thesis: Application for measuring the working times of machining centers

Supervisor: Niko Ristimäki

Year:2024 Number of pages:41

The thesis was conducted for a company that wanted to collect time data from a machining center. The company operates in the field of new construction. The aim of the work was to create an application for reading machining times from the TwinCAT-based CNC-machines of the company. Based on the collected data, necessary changes could be made in production and maximum benefit could be obtained from the material flow.

Several different pieces of software and operating systems were compared in the thesis, based on which the best possible software and operating system environment were chosen. The code for reading the machining tools was explained with illustrations. A test version was made for a separate computer, where it was tested and run. The test version was made for a computer with Linux as the operating system. When comparing the results, it was noticed that reading time data from machine tools was successful. Python was used as the implementation language for the application.

As a result, the company gained a clearer understanding of how long the machines are in operation in a day and whether there have been any issues with the machines. The company also received information on how additional functionalities could be added to the software.

[1] Keywords: software, machines, Linux, Python, program

SISÄLTÖ

Kuva- ja taulukkoluettelo

Käytetyt termit ja lyhenteet

ADS ADS (Automation Device Specification) on TwinCAT-tiedonsiirtoprotokolla, joka määrittää kahden ADS-laitteen välisen vuorovaikutuksen. Se esimerkiksi määrittelee, mitä operaatioita voidaan suorittaa toisella ADS-laitteella, mitä parametreja siihen tarvitaan ja mikä palautusarvo lähetetään suorituksen jälkeen.

AMS AMS (Automation Message Specification) määrittää ADS-tietojen vaihdon. Viestintäprotokollan tärkeä osa on AmsNetId. Tämä on määritetty lähde- ja kohdelaitteen AMS/ADS-paketissa. ADS-laite voidaan osoittaa nimenomaisesti käyttämällä AmsNetId-tunnusta.

AmsNetId AmsNetId on yhdistetyn ADS-laitteen yksilöllinen osoite, joka korvataan kohdejärjestelmän todellisella AmsNetId-osoitteella viestinnän aikana. Tämä tarkoittaa, että AmsNAT-toiminto varmistaa kaikessa ADS:n kautta tapahtuvassa tiedonsiirrossa, että kohdejärjestelmän AmsNetId korvataan.

CVS CVS (Concurrent Versions System) on ohjelma, jonka avulla koodin kehittäjä voi tallentaa ja hakea erilaisia kehitysversioita lähdekoodista. Sen avulla myös kehittäjätiimi jakaa tiedostojen eri versioiden hallinnan yhteisessä tiedostovarastossa.

GNOME Network Object Model Environment on graafinen käyttöliittymä ja joukko tietokoneen työpöytäsovelluksia Linux-käyttöjärjestelmän käyttäjille.

KDE K Desktop Environment on työpöytäympäristö Linux-pohjaiselle käyttöjärjestelmälle.

Pyads Pyads on Python-kirjasto, joka tarjoaa ohjelmoijille mahdollisuuden kommunikoida Beckhoffin automaatiojärjestelmien kanssa.

PyCharm PyCharm on ohjelmointikieli. PyCharm on hybridialusta, jonka JetBrains on kehittänyt Pythonin IDE:ksi. Sitä käytetään yleisesti Python-sovellusten kehittämiseen.

Python IDE Python IDE on Pythoniin integroitu kehitysympäristö, joka auttaa ohjelmoijia kehittämään ohjelmakoodia tehokkaasti.

TCP/IP TCP/IP (Transmission Control Protocol / Internet Protocol) on usean internetliikennöinnissä käytettävän tietoliikenneprotokollan yhdistelmä eli pino. IP-protokolla on alemman tason protokolla, joka vastaa päätelaitteiden osoitteistamisesta ja verkkopakettien reitittämisestä verkossa.

TIA Portal TIA Portal viittaa Totally Integrated Automation Portal -termiin. Se on Siemensin kehittämä ohjelmistoympäristö, joka tarjoaa integroidun ratkaisun automaatio-, ohjaus- ja käyttöliittymäteknologioihin.

TwinCAT 3 TwinCAT 3 on Beckhoff Automationin kehittämä ohjelmistoalusta, joka tarjoaa integroidun ympäristön automaatio- ja ohjausjärjestelmien kehittämiseen. Se tukee ohjelmoitavaa logiikkaa (PLC), liikkeenohjausta, HMI (Human Machine Interface) -sovelluksia ja muita automaatioon liittyviä toimintoja. TwinCAT 3 on suunniteltu erityisesti helpottamaan teollisuusautomaation kehitystä ja integrointia.

Xfce XFCE tulee sanoista XForms Common Environment. Se on ilmainen ja avoimen lähdekoodin työpöytäympäristö Unix-tyyppisille käyttöjärjestelmille, mukaan lukien Linux.

1 JOHDANTO

1.1 Työn tausta

Yritys, jolle opinnäytetyö tehtiin, toimii uudisrakentamisen ja valmisrakentamisen alalla. Yritys valmistaa alumiini- ja metalliosia, joita käytetään parveke- ja terassivalmistuksessa. Yrityksellä oli tarve sovellukselle, joka pystyisi lukemaan työstöaikaa kaikista työstökoneista. Yrityksellä ei ollut olemassa ohjelmaa, joka tarjoaisi tietoa työstökoneiden käynnissäoloajasta. Sovelluksen avulla voitaisiin parantaa töiden priorisointia, varmuutta ja seurata viikkokohtaisten töiden valmistumista.

Sovelluksen tulisi olla helppolukuinen, jotta sitä voitaisiin muokata tulevaisuudessa. Sovelluksen tulisi osata lukea työstökoneiden työstöaikaa, näin saataisiin tietää, paljonko työstökoneiden ei-käynnissä oleva aika, on päivässä.

1.2 Työn tavoite

Työn tavoitteena oli kehittää sovellus, jonka päämäärä on selkeästi määritelty. Tämä sovellus suunniteltiin ja toteutettiin siten, että se kykenisi lukemaan kaikkien yrityksen käyttöönottamien työstökoneiden tiedot, joista yritys haluaa kerätä koneiden työstöaikoja. Työstökoneista kertyvä data tallennetaan huolellisesti yrityksen palvelimelle, mahdollistaen kattavan ja helposti käytettävän tietopankin.

Sovelluksen ensisijainen tehtävänä oli lukea erilaisten työstökoneiden tilatietoja. Tämä vaati tarkkaa integrointia eri koneiden järjestelmiin, jotta tiedonkeruu sujuu saumattomasti. Työstökoneista haluttu data tallennetaan systemaattisesti, varmistaen, että tärkeä informaatio on helposti saatavilla ja jäsenneltyä, sillä kaikkien käyttäjien pitää osata lukea tietoa oikein. Yrityksen puolesta projektiin nimettiin avustajaksi yksi ohjelmistokehittäjä, jonka tehtävänä oli valvoa projektin kulkua. Tehtävään nimetty yrityksen edustaja auttoi projektin edetessä hankaliksi osoittautuneissa paikoissa, esimerkiksi käyttöjärjestelmän ja ohjelmointikielen valinnassa.

Kerätty data tarkastellaan ja analysoidaan huolellisesti. Siitä tutkitaan, miten eri tekijät vaikuttavat työstöaikadataan, ja pyritään ymmärtämään, mitkä tekijät ovat merkittäviä ja mitkä

eivät. Tämä vaihe on olennainen, jotta voidaan tunnistaa mahdolliset poikkeamat ja hienosäätää sovellusta tarvittaessa.

Kun dataa on tarkasteltu monipuolisesti, arvioidaan sen hyödyllisyyttä yritykselle. Tavoitteena on tunnistaa ne tiedot, jotka tuovat lisäarvoa päätöksentekoon ja auttavat optimoimaan työstöprosesseja. Tämä vaihe on kriittinen, sillä se varmistaa, että ohjelman kehitys johtaa todelliseen liiketoiminnalliseen hyötyyn.

1.3 Työn rakenne

Kyseessä olevan toimialan kilpailullisuuden vuoksi ja yrityksen toiveesta opinnäytetyössä ei paljasteta yritystä, jolle sovellus on kehitetty.

Opinnäytetyön teoriaosassa käydään läpi millä ympäristöllä sovellus on tehty ja miksi. Alussa kerrotaan, miksi työhön on valittu tietty ohjelmisto ja tietty kehitysympäristö. Ohjelmistoja vertaillaan keskenään, kerrotaan, miksi juuri tietty ohjelmisto tai käyttöjärjestelmä on valittu kyseiseen projektiin. Työn toisena vaiheena oli yhteyden luonti tietokoneen ja työstökoneiden välillä. Samassa yhteydessä käydään läpi, mitä AMS ja ADS tarkoittavat ja miksi ne on valittu juuri tähän työhön.

Varsinaisen sovelluksen kehityksen vaiheessa opinnäytetyössä esitellään yksityiskohtaisesti itse sovellus kuvien avulla. Kuvien avulla havainnollistetaan sovelluksen eri osa-alueita ja toiminnallisuuksia. Tämän osion tarkoituksena on antaa lukijalle selkeä käsitys siitä, miten sovellus on rakennettu ja miten se toimii käytännössä. Opinnäytetyössä on kuvia, joissa osa tiedoista on piilotettu tietoturvan vuoksi.

Viimeisenä osana opinnäytetyössä on tulosten tarkastelu ja niistä saadut hyödyt. Työstökoneiden työstöaikoja on mitattu manuaalisesti kesän 2023 aikana ja siitä saatua dataa on verrattu sovelluksen keräämään dataan. Tuloksia tavoitellaan vertailun perusteella ja pohditaan luotettavuutta ja kehitetyn sovelluksen toimintaa.

2 SOVELLUKSEEN LIITTYVÄT TEKNOLOGIAT

2.1 Työstökone

Työstökoneista käytetään lyhennettä CNC. Lyhenne CNC tulee sanoista Computer Numerical Control eli tietokoneistettu numeerinen ohjaus. Tämä tarkoittaa tekniikkaa, jossa käytetään tietokoneohjelmia ja NC-ohjelmia, ohjaamaan koneen liikettä (Longshengmfg, 2023). Kyseisellä NC-koodilla ohjataan koneen liikettä ja toimintoja.

Numeerisen ohjauksen ansiosta CNC-koneet mahdollistavat nopeamman ja laadukkaamman tuotannon ilman manuaalista koneistusta. CNC-järjestelmät tarjoavat monia etuja, kuten kustannusten vähentämisen, jätteiden vähentämisen, työntekijöiden turvallisuuden parantamisen, inhimillisten virheiden minimoimisen, ääriviivan työstön mahdollistamisen, nopeamman MCU-ohjelmoinnin, parannetun operatiivisen älykkyyden ja tuotannon pullonkaulojen vähentämisen (Yasar, 2023).

2.2 Käyttöjärjestelmän vertailu

Työssä vertailtiin Windows 10-järjestelmän ja Linuxin-käyttöjärjestelmien välisiä eroja. Linux ja Windows ovat hyvin erilaisia käyttöjärjestelmiä. Nykyään Linux-käyttöjärjestelmät ovat tulleet käytettävyydeltään ja ominaisuuksiltaan lähemmäksi Windowsin käyttäjäympäristöjä (Whitson, 2023). Linux Graduna-käyttöjärjestelmä on yksi lähimpänä oleva Windowsin käyttöjärjestelmän ympäristöä (Gangwar, 2022).

Taulukko 1. Linuxin ja Windowsin eroja (Darlington, 2023)

Ominaisuus	Linux	Windows 10
Lähdekoodi	Avoin lähdekoodi	Suljettu lähdekoodi
Kustomoitavuus	Erittäin kustomoitavissa	Rajoitetumpi käyttöliittymän muokkaus ja teemat
Suorituskyky	Usein kevyempi resurs-seissa	Raskas resurssien käyttö, erityisesti uudemmissa versioissa
Turvallisuus	Yleensä vähemmän haittaohjelmia	Usein kohde haittaohjelmille, mutta myös vahva turvallisuus
Ohjelmistotuki	Laaja valikoima avoimen lähdekoodin ohjelmia	Monipuolinen kaupallisten ohjelmien tuki
Käyttöliittymä	Käyttäjä voi valita mieluisan käyttöliittymän (esim. GNOME,KDE)	Yhteinen käyttöliittymä, joka on suunniteltu helppokäyttöiseksi.
Hinta	Ilmainen	Maksullinen käyttöjärjestelmä

Ensimmäiseksi vertailtiin Windowsin ja Linuxin käyttöjärjestelmien peruseroja. Windows-käyttöjärjestelmä on yleisesti peruskäytössä hieman tunnetumpi, sillä se on monien kotitalouksista löytyvien tietokoneiden käyttöjärjestelmänä. Windows-käyttöjärjestelmän etuihin lukeutuivat helppokäyttöisyys sekä tunnettavuus. Tämä luettiin vahvuudeksi ja ajateltiin tuovan varmuutta kehitettävän sovelluksen kehitystyössä sekä käytettävyydessä aikaisemman tietopohjan perusteella. Linux-käyttöjärjestelmän ollessa ennakolta tuntemattomampi, sen valinta epäilytti. Yrityksessä työskentelevä ohjelmistokehittäjä kuitenkin korosti käyttöjärjestelmän

etuja. Linux-käyttöjärjestelmässä erityisesti Python-koodikielen käyttö koodien kirjoittamisessa olisi helpompaa.

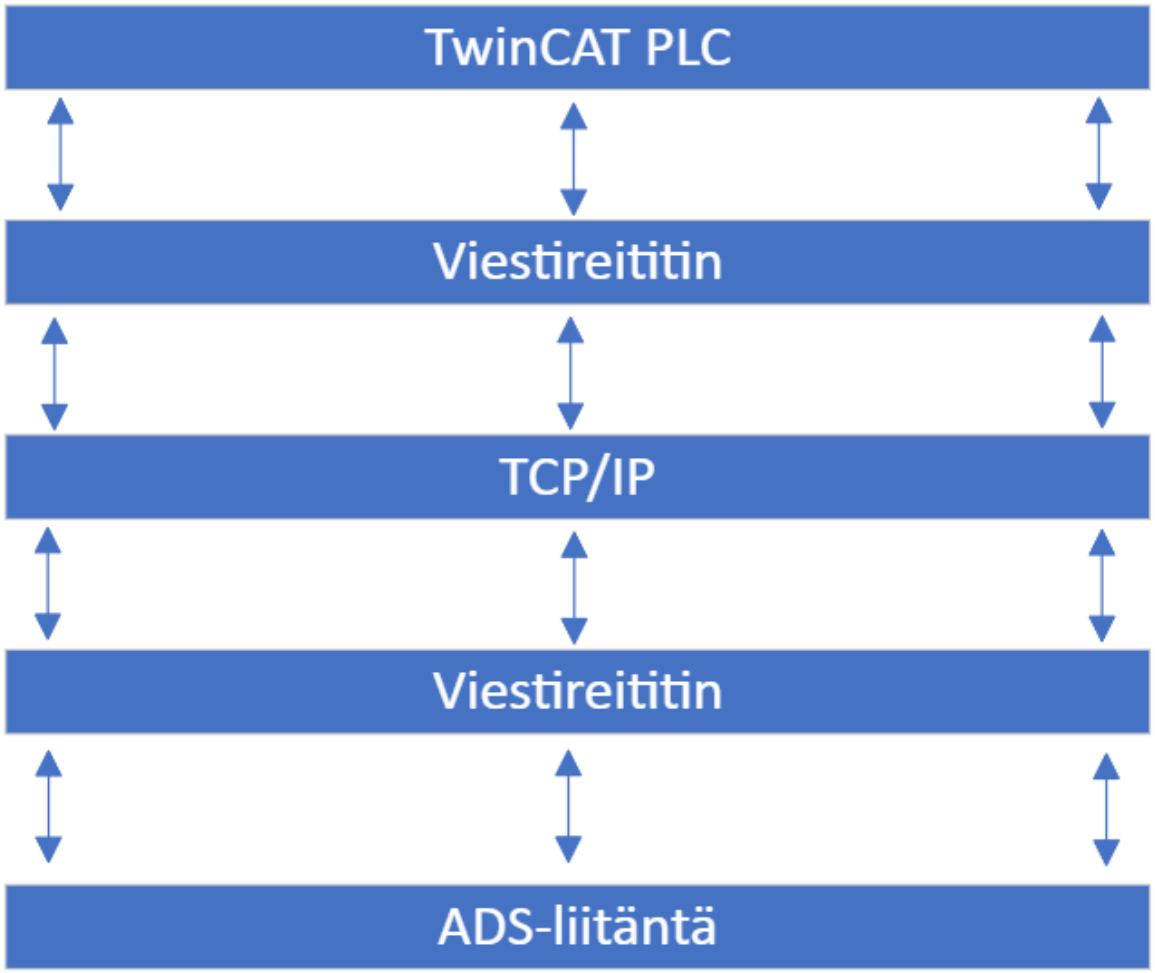
Taulukko 1 kertoo isoimmat erot käyttöjärjestelmissä. Linux-käyttöjärjestelmä on avoimen lähdekoodin käyttöjärjestelmä. Tämä tarkoittaa, että sen lähdekoodi on avoin yhteisölle (Red Hat, 2023). Käyttäjät voivat tarkastella, muokata ja jakaa lähdekoodia vapaasti. Windows 10 -käyttöjärjestelmä taas on suljetun lähdekoodin järjestelmä ja sen koodi ei ole julkisesti saatavilla. Microsoft hallinnoi ja ylläpitää Windows -käyttöjärjestelmän kehitystä.

Linux-käyttöjärjestelmillä on erilaisia työpöytäympäristöjä, kuten GNOME, KDE ja Xfce. Käyttäjä voi siis itse valita mieleisensä työnympäristön (Graduda Linux, i.a.). Windows 10 käyttöjärjestelmällä on yhtenäinen käyttöliittymä, joka on suunniteltu Microsoftin omaan tyyliin. Käyttäjät voivat muokata joitain tiettyjä asetuksia, mutta pääasiassa käyttöliittymä on määritelty Microsoftin toimesta (Darlington, 2023).

Ohjelmistotuki Linux-käyttöjärjestelmässä on laaja (Darlington, 2023). Valikoimasta löytyy ilmaisia ja avoimia ohjelmistoja. Windows 10 -käyttöjärjestelmässä on laaja valikoima maksullisia ohjelmia. Linux-käyttöjärjestelmän käyttäjä voi hallita mitä tahansa käyttöjärjestelmän osa-aluetta.

Windows -käyttöjärjestelmä vaatii enemmän tehoa verrattuna Linux -käyttöjärjestelmään (Software Testing Help, 2023). Suurin osa suurimmista supertietokoneista toimii Linux -käyttöjärjestelmällä. Linux tunnetaan suuresta kehitysnopeudestaan, kun taas Windows 10-järjestelmän tiedetään muuttuvan hitaammaksi.

Linuxilla on vahva maine tietoturvallisuuden saralla, osittain sen avoimen lähdekoodin luonteen ansiosta. Haavoittuvuudet pystytään havaitsemaan ja siten myös korjaamaan nopeasti (Memel, 2021). Windows 10 käyttöjärjestelmä on alttiimpi haittaohjelmille ja viruksille, koska se on suunniteltu toimimaan monenlaisilla laitteistoilla.



Kuva 1. TwinCAT-järjestelmän ADS yhteys

2.3 ADS ja AMS

Yhteyden luomiseksi työstökoneiden ohjaimille on välttämätöntä ottaa käyttöön AMS, jos halutaan käyttää ADS-yhteyttä. AMS (Application Management Service) on sovellusten hallintapalvelu, joka mahdollistaa turvallisen ja nopean datansiirron (Colton De Vos Managed Services, 2018). ADS-yhteys käyttää omaa osoitejärjestelmäänsä laitteiden tunnistamiseen, jota kutsutaan AmsNetId-järjestelmäksi. ADS puolestaan on lyhenne sanoista Automation Device Specification, joka tarkoittaa suomennettuna automaatiosovelluksen määritystä. TwinCAT-järjestelmäarkkitehtuuri mahdollistaa sovelluksen yksittäisten moduulien käsittelyn itsenäisinä laitteina. Viestien välityksestä huolehtii "viestireititin", joka välittää näiden objektien väliset viestit yhtenäisen ADS-rajapinnan kautta (Beckhoff New Automation Technology, i.a). Kaikkia viestejä hallitaan ja jaetaan järjestelmässä TCP/IP-yhteyksien päällä. TwinCAT-

viestireitittimet on integroitu jokaiseen TwinCAT-tietokoneeseen ja Beckhoff BCxxxx -väylä-ohjaimeen. Tämä mahdollistaa kaikkien TwinCAT-palvelin- ja asiakasohjelmien komentojen ja tietojen vaihdon. ADS-yhteyden kerrokset on kuvattu kuvassa 1.

ADS-protokolla käyttää TCP/IP-protokollaa (Klusaitė, 2022). Tietoliikenneprotokollapino TCP/IP määrittelee vaatimukset turvalliselle ja tehokkaalle tiedonsiirrolle verkossa. Tiedonsiirrosta vastaa TCP/IP-protokolla kahden eri laitteen välillä. Vaikka internetissä tietoa jaetaan monien laitteiden kesken, jokaisessa tiedonvaihdossa osallisena on vain kaksi laitetta. Nämä kerrokset kuvaavat sovellusten ja laitteiden toimintaa tiedonsiirron eri vaiheissa, alkaen alkuperäisestä verkosta lähtemisestä ja päättyen kohdeverkon saavuttamiseen.

Yhteyden luonti kehitettävässä sovelluksessa tapahtuisi AMS-yhteyden kautta, koska AMS-yhteyden avulla voidaan siirtää tietoa turvallisesti ja tehokkaasti (Colton De Vos Managed Services, 2018). Laitteiden määrittäminen AmsNetId-järjestelmälle toteutetaan reitityksen avulla (Lehmann, 2015). On tärkeää huomata, että reititysprosessi vaihtelee Windows- ja Linux-käyttöjärjestelmissä. Linux-käyttöjärjestelmässä ADS-yhteyden luomiseen tarvitaan ads-lib.so-kirjastoa, kun taas Windows-käyttöjärjestelmässä reititysprosessiin käytetään erillistä TcADSll.dll-kirjastoa.

Jotta ymmärrettäisiin reitityksen molemmat puolet, käytetään termejä "asiakas" ja "kohde". Asiakas viittaa tässä yhteydessä käyttäjän tietokoneeseen, jossa pyrkii toimimaan AMS- ja ADS-yhteydet. Kohde puolestaan tarkoittaa joko paikallista yritystä tai etätietokonetta, jonka kanssa halutaan muodostaa yhteys.

Reititysprosessi on keskeinen osa AmsNetId-järjestelmän käyttöä, ja sen ymmärtäminen on olennaista laitteiden tehokkaan tunnistamisen ja kommunikoinnin kannalta (Lehmann, 2015). Windows- ja Linux-käyttöjärjestelmissä reititys käsitellään eri tavoin, mikä vaatii käyttäjältä tietämystä kummankin järjestelmän toimintaperiaatteista. Windows-käyttöjärjestelmässä Py-adsin käyttöön täytyy käyttää TwinCAT Router UI-ohjelmaa, joka tulee TwinCAT-ohjelmiston mukana. Linux-käyttöjärjestelmässä reititys tehdään koodiin ja työstökoneelle, johon yhteys halutaan.

2.4 Käyttöjärjestelmän valinta

Linux-käyttöjärjestelmän hyvät puolet tulevat esille erityisesti sen avoimessa lähdekoodissa. Avoimen lähdekoodin hyödyt kehitettävässä sovelluksessa liittyvät esimerkiksi sovelluksen päivitystarpeeseen. Sovelluksen päivitystarpeeseen avoimen lähdekoodin etu on erityisen merkittävä. Koska koodi on avointa, kehittäjät voivat reagoida nopeasti muuttuviin tarpeisiin ja tehdä päivityksiä ilman pitkiä odotusaikoja (ESA automation, 2020). Tämä auttaa varmistamaan, että sovellus pysyy ajan tasalla ja toimii tehokkaasti erilaisten käyttäjien tarpeiden mukaisesti. Linux-käyttöjärjestelmän lähdekoodin avoimuus mahdollistaa uusia lähestymistapoja ja ajattelutapoja. Tämä tekisi koodin kirjoittamisesta sujuvampaa ja vähentäisi ulkoisten riippuvuuksien määrää. Linux-käyttöjärjestelmän turvallisuus on todella hyvä verrattuna Windows-käyttöjärjestelmän turvallisuuteen.

Näiden seikkojen pohjalta vertailtiin, kumpi käyttöjärjestelmä olisi soveltuvampi alusta kehitettävälle sovellukselle. Ohjelmoitavan ympäristön valinta kohdistui Linux Graduda versioon, sillä se oli hyvin lähellä Windowsin ohjelmoitavaa ympäristöä (Graduda Linux, i.a.-a.). Tähän valintaan vaikuttivat aikaisemmin mainittu tuntemus jo entuudestaan Windows-käyttöjärjestelmästä. Linux Gradudassa yhdistyivät kehitettävän sovelluksen kannalta parhaat ominaisuudet, mutta käyttöjärjestelmä ei olisi käytettävyydeltään liian erilainen verrattuna Windowsin käyttöjärjestelmään. Tämän vuoksi Linux-käyttöjärjestelmä ei olisi monimutkainen ja hankala oppia käyttämään. Linuxin etuna ja parempina ominaisuuksina todettiin sovelluksen kehityksen kannalta turvallisuus ja avoin lähdekoodi. Linux Graduda käyttää BTRFS-tietojärjestelmää, jonka tavoitteena on toteuttaa edistyneitä ominaisuuksia ja keskittyä samalla vikojen korjaamiseen (Graduda Linux, i.a.-a.). Vaikka Windows -käyttöjärjestelmä oli tuttu ja turvallinen, Linux tarjosi sovellukselle paremmat mahdollisuudet suoraviivaisen koodin kirjoittamiseen ja projektin vaatimusten täyttämiseen. Nämä tekivät siitä optimaalisen valinnan tähän työhön.

2.5 Ohjelmointikielen valinta

Ennen sovelluksen kirjoittamista vertailtiin ohjelmointikieliä. Vertailulla pyrittiin löytämään mahdollisimman hyvä ohjelmointikieli, joka mahdollistaisi koodin tekemisen yksinkertaiseksi ja niin, että sitä on tulevaisuudessa helppo muokata. Vertailuun valittiin kaksi eri kieltä, Java ja Python. Nämä valikoituivat vertailuun yksinomaan siitä syystä, että ne ovat käytetyimpiä ohjelmointikieliä. Java-ohjelmointikielen käyttäjäkunta kaikista ohjelmointikielten käyttäjistä on

16.14 %, kun taas Python-ohjelmointikielen käyttäjäkunta on 27.43 % (Fetisov & Birukova, 2023).

Java-ohjelmointikieli on alustariippumaton ja objektiohjelmointikieli, mikä tekee siitä helppokäyttöisen ja ylläpidettävän (Olumide, 2023). Se käyttää Java Virtual Machinea (JVM) mahdollistaen saman koodin suorittamisen eri järjestelmissä. Java-ohjelmointikieli on suunniteltu kestäväksi ja sisältää mekanismeja virheiden ja muistin hallintaan, mikä vähentää kaatumisten riskiä. Java-ohjelmointikielellä on korkea suorituskyky suurille sovelluksille. Java-ohjelmointikielen huonona puolena on koodirivien pituus. Tämän ongelman havainnollistamiseksi kehitettiin pieni pätkä koodia sekä Java- että Python-ohjelmointikielillä. Molemmilla ohjelmointikielillä haluttiin tietää käyttäjän nimi ja ikä. Esimerkit on esitetty kuvissa 2 ja 3. Kuvassa 2 on kyseinen koodi Java-ohjelmointikielellä seitsemälle komentoriville kirjoitettuna. Python-ohjelmointikielellä taasen voidaan samat tiedot selvittää käyttämällä kolmea käskyriviä, kuten kuvassa 3. Täysin funktionaalista ohjelmointia Java-ohjelmointikielellä ei voi tehdä (Staff, 2023). Python-ohjelmointikielen kirjoittaminen on siis selkeästi nopeampaa verrattuna Javaan, ja vastaava toiminto Python-koodissa ei vaadi niin montaa käskyä.

```java
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        // Luodaan Scanner-olio käyttäjän syötteiden lukemista varten
        Scanner scanner = new Scanner(System.in);

        // Kysytään käyttäjältä nimi
        System.out.print("Anna nimesi: ");
        String nimi = scanner.nextLine();

        // Kysytään käyttäjältä ikä
        System.out.print("Anna ikäsi: ");
        int ika = scanner.nextInt();

        // Tulostetaan saadut tiedot
        System.out.println("Tervehdys, " + nimi + "! Olet " + ika + " vuotta vanha.");

        // Suljetaan Scanner-olio
        scanner.close();
    }
}
```

Kuva 2. Java-ohjelmointikieliesimerkki, tekijä Väinö Koski

```python
# Kysytään käyttäjältä nimi
nimi = input("Anna nimesi: ")

# Kysytään käyttäjältä ikä
ika = input("Anna ikäsi: ")

# Tulostetaan saadut tiedot
print("Tervehdys,", nimi + "! Olet", ika, "vuotta vanha.")
```

Kuva 3. Python-ohjelmointikieliesimerkki, tekijä Väinö Koski

Yrityksen edustaja korosti, että ohjelmasta ei haluttu tehdä vaikeaa ja raskaslukuista. Yksi ratkaiseva vaikuttaja valinnassa oli hänen suosituksensa käyttää Python-ohjelmointikieltä. Yrityksen ohjelmistokehittäjä hallitsi kieltä erinomaisesti työnsä vuoksi ja tarjoutui auttamaan, mikäli sovelluksen kehityksessä ilmenisi ongelmia ohjelmointikielen suhteen. Sovelluksen kehittämiseen valittiin Python-ohjelmointikieli. Todettiin myös, että Python-ohjelmointikieli mahdollistaisi sovelluksen kehittämistä sujuvasti jatkossa.

2.6 Ohjelmointiympäristö

Seuraavaksi tutkittiin, millä ohjelmointiympäristöllä olisi helpoin tehdä koodi. Ohjelmointiympäristön tulisi olla turvallinen ja helppokäyttöinen. Testiversio tehtiin aluksi Linux-käyttöjärjestelmälle, josta se siirrettiin yrityksen omalle palvelimelle. Testisovellus oli käynnissä kokopäiväisesti yrityksen omalla palvelimella. Entuudestaan tutut ohjelmistoympäristöt olivat Visual Studio sekä PyCharm. Opintojen aikana Visual Studio ja Visual Studio Code olivat käytetyimmät ohjelmistoympäristöt.

Visual Studio Code, josta käytetään myöhemmin lyhennettä VSCode, on kevyt ja käynnistyy nopeasti. Se tarjoaa tehokkaan kehitysympäristön ilman raskasta järjestelmäresurssien käyttöä (Visual Studio Code, i.a.). VSCode on nopea lähdekoodieditori, joka tukee monia ohjelmointikieliä ja tarjoaa syntaksikorostuksen, pariliitoksen, automaattisen sisennyksen, laatikkovalinnan ja muita hyödyllisiä ominaisuuksia. Intuitiiviset pikanäppäimet ja yhteisön luomat näppäinkartat helpottavat koodissa navigointia. VSCode tarjoaa käyttäjälle ilmaisen lähdekoodieditorin, joka tukee täysin Python-ohjelmointikieltä ja hyödyllisiä ominaisuuksia, kuten

reaaliaikaista yhteistyötä. VSCodea päivitetään säännöllisesti ja siihen lisätään jatkuvasti uusia ominaisuuksia ja parannuksia.

Pycharm on Python-ohjelmointikieleen integroitava kehitysympäristö, joka tarjoaa käyttäjälle laajan valikoiman olennaisia työkaluja luodakseen käyttäjälle ympäristön, jossa käyttäjä voi kehittää omaa ohjelmaansa (Jet Brains, i.a.). PyCharm on tehokas, mutta aloittelijoille hieman rajoittunut. Esimerkiksi PyCharm voi olla uusille käyttäjille hidas ja resurssikulutusta lisäävä ohjelmisto. Pycharmin käynnistyminen voi viedä aikaa, jos ladataan suuria projekteja (Teamcode, 2023). Pro-versio ohjelmointiympäristöstä on kallis, ja ilmaisversion ominaisuudet ovat siihen verrattuna rajalliset. PyCharm ei tuo paljon lisäarvoa perusskriptien suhteen, joten se sopii paremmin kokeneille sovelluskehittäjille. PyCharm on hyvä työkalu Python-projekteihin, mutta aloittelijoille suositellaan kokeilemaan ilmaisversiota ennen pro-version hankkimista (Ramakrishnan, 2022)

Projektin kehitysympäristöstä haluttiin mahdollisimman helppo ja yksinkertainen. Yrityksen sovelluskehittäjä ehdotti työympäristöksi Linux-käyttöjärjestelmän konsolissa koodaamista (Thomas, 2007). Linux-käyttöjärjestelmän konsoli tarkoittaa ohjelmointia, jossa ei voi käyttää flash- eikä tapahtumalähtöisiä ominaisuuksia. Yleensä konsolikoodaus on tarkoitettu yksinkertaisille näytöille, joissa on rajoitettu tila. Konsolikoodaus tehdään sovelluksessa, joka suoritetaan komentokehoteikkunassa.

Visual Studio Code todettiin luontevaksi valinnaksi sovelluksen kehitykseen useista eri syistä. Tärkeimpänä huomiona oli sen käyttöliittymä käyttäjäystävällisyys ja selkeys. Työskentely Visual Studio Codessa ja sen monipuoliset ominaisuudet tekisivät koodin kirjoittamisesta sujuvaa. Visual Studio Code tarjoaa myös laajan valikoiman laajennuksia ja lisäosia, jotka tekisivät ohjelmoinnista joustavaa ja räätälöityä. (Turing, i.a.). Sovellukseen pystyttäisiin helposti integroimaan erilaisia työnkaluja ja tarvittaessa laajentamaan koodia vastaamaan tarkemmin projektin vaatimuksia. Visual Studio Code on kehitetty erityisesti web- ja pilviympäristöihin, mikä sopi hyvin suunnitellulle projektille. Sen sisäänrakennettu tuki Git-versiohallinnalle oli myös merkittävä etu, sillä projektissa oli tarve tehokkaalle yhteistyölle. Ohjelmointiympäristöksi valittiin siis Visual Studio Code, koska se tarjosi helppokäyttöisen, monipuolisen ja kehittyvän kehitysympäristön, joka vastasi parhaiten yrityksen tarpeita.

3 NYKYTILAN KARTOITUS

3.1 Sovelluksen tarpeet

Yrityksellä ei ollut käytössään mitään ohjelmistoa, joka mahdollistaisi työstökoneiden työaikojen seurannan. Yrityksessä oli todettu tarve saada käyttöön sovellus, joka kykenee tarkkailemaan kaikkien työstökoneiden käyttöastetta. Tällainen sovellus toisi selkeyttä mahdollisiin ongelmakohtiin ja mahdollistaisi tehokkaan raportoinnin yrityksen johdolle.

Tällaisen sovelluksen käyttöönotto tarjoaisi monia etuja. Sovellus mahdollistaisi tarkan seurannan kunkin työstökoneen työajasta, mikä auttaisi tunnistamaan tehokkuuteen liittyviä tekijöitä ja mahdollisia pullonkauloja tuotantoprosesseissa. Lisäksi sovellus voisi antaa arvokasta tietoa ei-käynnissä olevasta ajasta, mikä auttaisi optimoimaan koneiden käyttöä ja minimoimaan seisokkiaikoja.

Sovelluksen avulla olisi myös mahdollista havaita ennakoivasti mahdolliset ongelmat tai huoltotarpeet, mikä taas edistäisi laitteiden kunnossapitoa ja pidentäisi niiden käyttöikää. Sovelluksen avulla voitaisiin mahdollisesti vähentää odottamattomia huoltokustannuksia ja parantaa tuotantotehokkuutta pitkällä aikavälillä.

Raportointimahdollisuudet olisivat tärkeitä yritykselle, sillä ne tarjoaisivat kattavan kuvan yrityksen tuotantoprosessien tilasta. Erilaisten raporttien avulla voitaisiin analysoida työaikojen jakautumista, vertailla eri koneiden suorituskykyä ja tehdä päätöksiä tuotantotehokkuuden parantamiseksi.

Yhteenvetona voidaan todeta, että työstökoneiden työaikojen seurantajärjestelmän käyttöönotto olisi merkittävä askel kohti tehokkaampaa ja läpinäkyvämpää tuotantoa yrityksessä. Se parantaisi päätöksentekoa ja auttaisi optimoimaan resurssien käyttöä, mikä voisi lopulta vaikuttaa positiivisesti yrityksen liiketoiminnan tulokseen.

3.2 Sovelluksen merkitys

Sovelluksen käyttöönotolla olisi useita positiivisia vaikutuksia liiketoimintaan. Se auttaisi havaitsemaan mahdolliset ongelmat nopeasti ja tehokkaasti, mikä mahdollistaisi ripeän

reagoinnin ja ongelmien ratkaisemisen ennen kuin ne aiheuttavat suurempia vahinkoja tai tuotantokatkoksia. Tämä parantaisi merkittävästi työn sujuvuutta ja tehokkuutta.

Lisäksi sovellus mahdollistaisi systemaattisen datan keräämisen työstökoneiden toiminnasta. Tämä tieto olisi arvokasta yrityksen johdolle, sillä se antaisi kattavan kuvan siitä, kuinka kauan eri valmistusvaiheisiin kuluu aikaa sekä miten eri komponentit ja raaka-aineet vaikuttavat aikaan. Tämä tieto voisi toimia perustana päätöksenteolle ja auttaa tunnistamaan mahdollisia pullonkauloja tuotantoprosessissa.

Sovellus toisi merkittävän parannuksen tuotantoprosessin hallintaan. Sen avulla voidaan optimoida resurssien käyttöä, parantaa tuottavuutta ja minimoida mahdolliset häiriöt tuotantoketjuissa. Sovelluksen käyttöönotto olisi investointi, joka maksaisi itsensä takaisin kasvaneena tehokkuutena ja liiketoiminnan kilpailukyvyn kasvuna.

3.3 Yrityksen työstökoneet

Yrityksellä on käytössään kuusi työstökonetta. Työstökoneet ovat käynnissä kiireellisenä aikana ympäri vuorokauden. Työstökoneita ajetaan kolmessa työvuorossa. Kiireisimmän sesongin jälkeen työstökoneet siirretään kahteen vuoroon, aamu- ja iltavuoroon. Työstökoneiden avulla tuotetaan yrityksen tuotantoon tarvittavia materiaaleja. Työstökoneet ovat toiminnan kulmakiviä, kyseiset laitteet ohjaavat muuta tuotantoa. Jos työstökoneet eivät tuota tarpeeksi materiaalia, silloin tuotanto seisahtuu. Työstökoneiden tuottavuus täytyy olla tarpeeksi korkea, jotta turvataan riittävä materiaalivirta.

3.4 Työstökoneiden merkit ja mallit

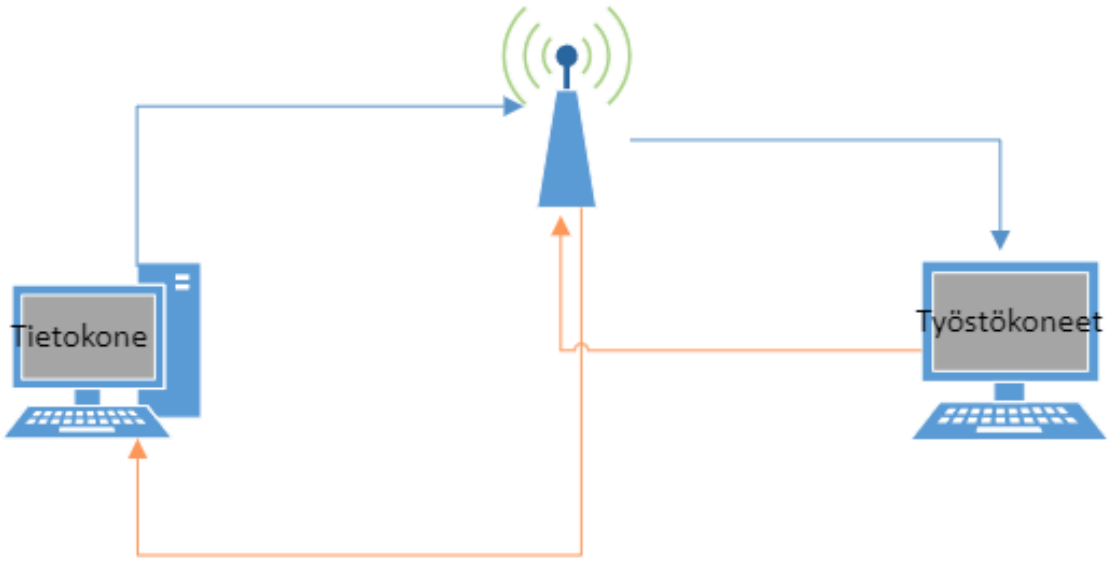
Yrityksellä on yhteensä kuusi työstökonetta joista, neljä on ohjattu TwinCAT-logiikalla. Kahdessa yrityksen koneessa on Siemensin-logiikat. Opinnäytetyössä kehitetyn sovelluksen tarkoituksena on kerätä dataa TwinCAT-logiikoilla toimivista työstökoneista. Siemensin logiikkaa ja TwinCAT-logiikkaa ei voida lukea samalla ohjelmistolla, sillä niissä on eri ohjelmointiympäristöt. Yrityksellä on kaksi Elumatec SBZ 628 XL-työstökonetta. Yrityksellä on käytössä Elumatecin vanhempi malli SBZ 628 XLR, joka käyttää TwinCAT 2-ohjelmaa. SBZ 628 XLR on hyvin samankaltainen, kuin SBZ 628 XL, isoimmat erot ovat nopeudessa ja eri TwinCAT-

järjestelmissä. Yrityksen viimeinen kone, johon sovellusta käytettäisiin, on Elumatecin SBZ 630. SBZ 630 on pienempi työstökoneyksikkö verrattuna yrityksen muihin työstökoneisiin.

4 TOTEUTUS

4.1 Ohjelmointi

Kun kehitettävän sovelluksen käyttöjärjestelmä, ohjelmointikieli ja -ympäristö oli valittu, sovelluksen kehittäminen aloitettiin luomalla yhteys tietokoneen ja työstökoneiden välille. Yhteyden luonti tapahtui Pyadsin kautta. Ensimmäinen ongelma ilmeni yhteyden luontivaiheessa. Yhteydet katosivat päivän jälkeen työstökoneilta, koska yrityksen työstökoneiden IP-osoitteet vaihtuivat joka päivä. Jotta yhteys ei katoaisi, jokaiselle työstökoneelle ja tietokoneelle luotiin kiinteät IP-osoitteet. Näin saatiin yhteydet pysymään katkeamattomina. Työstökoneista tulisi löytää muuttujia, joita voitaisiin käyttää sovelluksen koodin kutsuissa. Tämä on tarpeellista varmistaa, että sovellus toimii sujuvasti ja pystyy kommunikoimaan eri laitteiden välillä ilman ongelmia. Näiden muuttujien avulla voidaan varmistaa, että sovellus pystyy tunnistamaan, onko kyseinen työstökone käynnissä vai pysähtynyt. Työstökoneet, joille sovellus kehitettiin käyttävät Beckhoff-logiikkaa. Beckhoff-ohjesivustolta löytyi selkeät selitykset eri muuttujista ja error-koodeista (Beckhoff New Automation Technology, i.a.). Tätä ohjesivustoa lukemalla saatiin käsitys siitä, mitä muuttujia voitaisiin käyttää sovelluksessa.



Kuva 4. Yhteyden luonti

kuvan 4 avulla havainnollistetaan yksinkertaistettuna, miten tieto kulkee tietokoneen ja työstökoneiden välillä. Tietokone selvittää verkon välityksellä työstökoneilta kahden sekunnin välein, onko työstökone käynnissä vai ei. Tämän jälkeen työstökoneet ilmoittavat tietokoneelle

tiedon työstökoneen käynnissä olosta tai pysähtymisestä. Tarkoituksena oli kehittää sovelluksesta mahdollisimman yksinkertainen, mutta kuitenkin turvallinen.

Jokaiselle työstökoneelle piti tehdä muutoksia, jotta yhteys työstökoneen ja tietokoneen välillä saatiin toimimaan. Jokaisen työstökoneen ja tietokoneen IP-osoite piti vaihtaa suojattuun IP-osoitteeseen. Uusien suojattujen IP-osoitteiden käyttö mahdollisti pysyvän yhteyden muodostamisen työstökoneiden ja tietokoneen välille, kun jokaisen työstökoneen ja tietokoneen IP-osoite vaihdettiin turvallisempaan vaihtoehtoon. Tämä varmisti, että yhteydet pysyivät vakaana ja katkeamattomana, mikä oli välttämätöntä sovelluksen toiminnan kannalta. Uuden IP-osoitteen yritys osti tietoturvayritykseltä. Tämän jälkeen IP-osoitteet muutettiin työstökoneille ja tietokoneelle manuaalisesti, jotta yhteydet voitaisiin tehdä mahdollisimman turvalliseksi. Jokaiselle työstökoneelle laitettiin manuaalisesti AMS-yhteydet tietokoneeseen, mikä takasi yhteyden jatkuvan ympärivuorokautisen katkeamattoman käytön.

Tietokoneiden ja työstökoneiden yhteyksien korjaamisen jälkeen alkoi itse sovelluksen koodin kirjoittaminen. Jokaisesta työstökoneesta löytyi muuttujalistat, joista tuli selvittää, mitä muuttujaa voidaan kutsua ja mitä ei. Muuttujalla oli iso merkitys sovelluksen toiminnan kannalta. Jos sovelluksessa käytetään vääriä muuttujia, mitatun ajan tarkkuus voi olla väärä. Muuttujia testaamalla ja vertailemalla selvitettiin mahdollisimman tarkka aika.

4.2 Muuttujat

Jokaiselta työstökoneelta tuli etsiä tarvittavat muuttujat. Yrityksellä on kaksi Elumatec SBZ 682 XL-työstökonetta, yksi SBZ 628 XLR ja Elumatec SBZ 630. Työstökoneiden muuttujat olivat hyvin samankaltaisia. Erona vanhemmissa työstökoneissa, joiden ohjelmointiympäristö on TwinCAT 2 oli, että muuttujien nimet olivat hieman erilaisia. Muuttujan täytyy olla sellainen, jonka arvo muuttuu 0-tilasta 1-tilaan, kun tietty asia muuttuu työstökoneessa. Uudemmille SBZ 628 XL-koneille pystyttiin käyttämään samoja muuttujia. SBZ 628 XLR- ja SBZ 630-koneelle täytyi etsiä tarkasti muuttujat. Ongelmia ilmeni erityisesti vanhempien koneiden muuttujien kohdalla, joissa kieli aiheutti haasteita. Useimmat näiden vanhempien koneiden muuttujat olivat täysin saksankielisiä ja muuttujien selitykset olivat myös saksaksi. Tämä vaati huolellista tulkintaa ja käännöstyötä, jotta varmistuttiin, että muuttujien merkitys ja toiminta ymmärrettiin oikein. Tässä tilanteessa korostui tarve tekniselle osaamiselle ja kielitaidolle, jotta voitiin varmistaa oikeiden muuttujien löytäminen ja niiden asianmukainen käyttö.

Työstökoneiden monimuotoisuus ja ohjausjärjestelmien eroavaisuudet tekivät muuttujien hallinnasta ja integroinnista haastavaa, mutta samalla tarpeellista, osana koneiden tehokasta ja tarkkaa valvontaa.

Muuttujien asianmukainen määrittely ja niiden riittävä käyttö ovat olennaisia tekijöitä, kun pyritään saamaan tarkkaa ja perusteellista tietoa työstöajasta teollisuusprosesseissa. Morrellin (2021) mukaan yritykset pyrkivät ymmärtämään tarkemmin miksi ja miten tietyn tilanteen tapahtumat tapahtuvat. Tämä tiedon syvällinen analysointi on keskeistä pyrittäessä mahdollisimman tarkkaan ja perusteelliseen tietoon.

Tulosten selittäminen eli miksi ja miten tapahtumakulku toimii, on keskeinen osa prosessia. Nämä tiedot antavat yritykselle mahdollisuuden tehdä tarvittavia muutoksia, jotka voivat edelleen nopeuttaa tuotantoa ja optimoida prosesseja. Tieto on tärkeää, kun pyritään saavuttamaan tehokkuutta ja parantamaan liiketoiminnan suorituskykyä.

Yrityksen pyrkiessä mahdollisimman tarkkaan ajanlaskentaan työstöprosessissa, on välttämätöntä käyttää riittävästi muuttujia. Yhden muuttujan käyttö ei välttämättä anna riittävän tarkkaa tietoa.

Muuttujia tarkistamalla jokaisesta työstökoneesta täytyi valita ulostulokelkan tunnistin. Työstökoneiden ulostulokelkka on osa työstökonetta, se vastaa valmistuneiden työkappaleiden poistamisesta työstökoneen toiminnan jälkeen. Se on automatisoitu rakenne, joka siirtää valmistuneet osat pois työstökoneen työalueelta. Ulostulokelkka kuljettaa työkappaleet kuljetushihnan avulla, jotta ne voidaan jatkojalostaa tai käsitellä edelleen. Tämä osa on tärkeä työstökoneiden tehokkaassa ja sujuvassa toiminnassa, koska se mahdollistaa jatkuvan työnkulun ja estää valmistuneiden osien kerääntymisen työstöalueelle, mikä voisi häiritä koneen toimintaa. Tämä muuttuja oli erityisen merkityksellinen, sillä työstökone saattaa pyöriä joka tapauksessa, vaikka kelkka olisi täynnä ja työstökone ei suorittaisi varsinaista työstötehtävää. Toinen tärkeä muuttuja sovellukseen oli työstökoneen sahan pyörimismuuttuja, joka kertoo, onko saha käynnissä. Lisäksi työstökoneen Run-painikkeen muuttuja tulee olla päällä aina, kun kone käynnistetään. Jos Run-painike ei ole päällä työstökoneen sahat eivät pyöri. Sovellukseen on myös lisätty akselipainikemuuttuja, joka laittaa akselit päälle. Ilman tätä muuttujaa akselit eivät voi liikkua. Akseli liikuttaa profiilia koneen sisällä. Tätä akselipainikemuuttujaa ei tarvitse laittaa päälle, jos työstetään yksittäistä leikkuuta, mutta jokaisessa yrityksen kohteessa on aina monta leikkuuta, joten sen vuoksi muuttuja valittiin listalle.

Näiden muuttujien avulla saadaan madollisimman tarkka tieto työstökoneiden aikadatasta. Tarkka ajanlaskenta ja prosessien seuranta antavat yritykselle tarvittavat työkalut tehokkuuden optimointiin ja tuotannon parantamiseen. Jatkuva seuranta ja muuttujien käyttö voivat johtaa tarkempaan analyysiin ja siten parempaan päätöksentekoon liiketoiminnan kehittämisessä.

4.3 Sovelluksen kehittäminen

Sovellukseen piti tuoda tarvittavat kirjastot, jotta ohjelmisto voisi toimia. Pyads on kirjasto, joka tarjoaa Python-ohjelmoijille pääsyn Beckhoffin automaatiojärjestelmien ADS (Automation Device Specification) -liittymään. Python-sovellukselle piti toteuttaa myös yhteyden luonti, jotta yhteys pystyttäisiin muodostamaan tietokoneen ja työstökoneen välille. Tämän takia sovelluksen koodiin tuotiin myös Pyadsin Connection-moduuli. Sovelluksesta saatava tieto haluttiin kirjoittaa CVS-tiedostoon, joten sovellukseen piti tuoda aikakirjasto Time. Time on Pythonin standardikirjasto, joka tarjoaa toimintoja ajan hallintaan ja mittaukseen. Time-kirjasto mahdollistaa ajan käsittelyn, kuten odottamisen, mittaamisen ja käsittelyn. Kun viimein tarpeelliset kirjastot olivat tuotu, voitiin aloittaa sovelluksen kirjoittaminen.

```
1   import pyads
2   from pyads import Connection
3   import time
4   import os
5
6   [redacted]
7   #
8   #Ohjelman on tarkoitus laskea työstöaikaa ja luppoaikaa.
9   #kt
10  class Kone:
11      def __init__(self,nimi,ams_net_id,IP,ads_port, muuttujat = None) -> None:
12          self.nimi = nimi
13          self.ams_net_id = ams_net_id
14          self.IP = IP
15          self.ads_port = ads_port
16          self.muuttujat = muuttujat
17          self.arvot = None
18          self.running = False
19          self.running_prev = False
20          self.yhteys = False
21          self.yhteys_prev = False
22          self.muutosaika = 0
23          self.muutos_klo = 0
24
```

Kuva 5. Sovelluksen ensimmäiset koodirivit

Sovellus aloitettiin tekemällä koodiin luokka nimeltään Kone. Luokka mahdollisti kaikkien koneiden tarpeellisten tietojen ja haluttavien muuttujien yhdistämisen yhteen kokonaisuuteen. Sen avulla voitiin määritellä koneen ominaisuudet ja toiminnot kootusti yhteen paikkaan, mikä helpotti koodin ymmärtämistä ja ylläpitoa. Luokka toi sovelluksen koodiin järjestystä ja rakennetta. Luokka toimi käytännössä mallina, joka määritteli, mitä ominaisuuksia ja toimintoja koneolioilla olisi. Tämä selkeytti koodin rakennetta ja teki siitä helpommin hallittavan. Luokka tarjosi muokattavuutta sovelluksen kehityksessä. Jos koneiden tietorakenne muuttuisi tulevaisuudessa, muutokset voitaisiin tehdä helposti luokan määrittelyyn ilman, että kaikkia ohjelman osia tarvitsisi muuttaa. Tämä säästi aikaa ja vaivaa kehitystyössä.

```python
    #Tämä kohta päivittää csv tiedoston tiedot joka looppi kierroksella.kt
    def update(self):
        self.running_prev = self.running
        self.yhteys_prev = self.yhteys
        try:
            with Connection(self.ams_net_id,self.ads_port, self.IP) as plc:
                if self.muuttujat:
                    self.arvot = [plc.read_by_name(i[0], i[1]) for i in self.muuttujat]
                    self.running = all([i and j[2] for i, j in zip(self.arvot, self.muuttujat)])
                    #Tietyt muuttujat pitää olla päällä ja sit jotkut ei, että ohjelma alkaa lukemaan työstöaika tietoa koneilta.
                    self.yhteys = True

        except: #pyads.Exception.AdsException:
            #jos kone ei ole päällä se ohittaa sen ja käy kysysymässä uudestaan seuraavassa loopissa.
            self.running = False
            self.yhteys = False
```

Kuva 6. Sovelluksen päivitys

Kuva 6 sisältää koodin osan, joka pitää sisällään update-metodin päivitykseen. Tämä metodi käyttää with-lauseketta varmistaakseen, että Connection-olio vapautetaan oikein, vaikka jotain menisi väärin. Jos yhteysyritys aiheuttaa virheen, esimerkiksi jos kone ei ole päällä tai yhteys epäonnistuu jostain syystä, metodi käsittelee virheen except-lohkossa. Silloin se asettaa running- ja yhteysmuuttujat arvoon False. Tämä koodi siis päivittää koneen tilaa yrittämällä muodostaa yhteyden ja lukea sen muuttujat. Jos tämä onnistuu, se päivittää tilat sen perusteella, mitä muuttujat osoittavat. Jos yhteysyritys epäonnistuu, se merkitsee koneen tilaksi ei käynnissä ja ei yhteyttä. Koodi ei suoranaisesti päivitä CSV-tiedostoa. Sen sijaan se päivittää luokan jäsenmuuttujia self.running ja self.yhteys, joita voidaan käyttää muualla sovelluksessa päivittämään CSV-tiedosto tarpeen mukaan.

```python
def log(self, tiedostonimi):
    # Tila on muuttunut tai yhteyttä ei ole muodostettu
    if self.running_prev != self.running or self.yhteys != self.yhteys_prev:
        # jos yhteys_prev on false ja self.yhteys on True. Hakee CSV tiedostosta viimeiseksi kirjatun ajan ja jatkaa siitä ajan laskemista.
        if self.yhteys_prev == False and self.yhteys == True:
            self.muutos_klo = lue_aiempi_aika(tiedostonimi, self.nimi)

        self.muutosaika = time.time() - self.muutos_klo if self.muutos_klo else 0
        self.muutos_klo = time.time()

        #Ilmoittaa csv tiedostoon jos self.running True Ajossa
        #ilmoittaa csv tiedostoon jos self.running False. pysähtynyt
        run_str = 'Ajossa' if self.running else 'Pysähtynyt'

        #ilmoittaa csv tiedostoon jos self.yhteys True Yhdistetty
        #ilmoittaa csv tiedostoon jos self.yhteys Flase Ei yhteyttä
        yht_str = 'Yhdistetty' if self.yhteys else 'Ei yhteyttä'
        #Tallentaa muutoksen silloin kun koneen tila muuttuu# Eli jos kone muuttuu ajosta tilaan ei yhteyttä tai sammunut. Kirjaa sen csv tiedostoon.

        # Tehdään uusi tiedosto päivämäärän mukaan, jotta tiedostokoko ei kerry liian suureksi
        tiedostonimi = tiedostonimi + time.strftime('%d%m%Y', time.localtime()) + '_log.csv'
        with open (tiedostonimi,'a') as tiedosto:
            tiedosto.write(f'{time.ctime()};{round(time.time(), 1)};{self.nimi};{self.yht_str};{self.run_str};{round(self.muutosaika, 1)}\n')
```

Kuva 7.Sovelluksen kohta, jossa kirjoitetaan tiedostoon tietoa ajasta

Tässä koodiosuudessa suoritetaan tiedonkirjaustoiminto, joka tallentaa koneen tilan muutokset CSV-tiedostoon annetussa tiedostopolussa (kuva 7). Toiminto tarkkailee koneen tilan muutoksia ja tallentaa merkinnän CSV-tiedostoon aina kun tila muuttuu. Jos koneen tila on muuttunut tai yhteyttä ei ole muodostettu edellisen tarkastelun jälkeen, koodi tarkistaa muutoksen luonteen. Jos yhteys aiemmin puuttui ja on nyt muodostettu, funktio hakee aiemman ajan CSV-tiedostosta ja jatkaa siitä ajan laskemista. Jos vanhaa aikaa ei löydy CVS-tiedostosta sovelluksen koodi aloittaa laskemaan nollasta. Muutosaika lasketaan nykyhetkestä ja viimeisestä muutosajasta. Sen jälkeen ohjelma ilmoittaa CSV-tiedostoon koneen tilan (ajossa tai pysähtynyt) ja yhteyden tilan (yhdistetty tai ei yhteyttä). Näitä tietoja käytetään merkinnässä tiedostoon. Lopuksi tiedosto avataan "append" -tilassa ja merkintä tallennetaan muotoiltuna aikaleiman, koneen nimen, yhteyden tilan, käynnissä olon tilan ja muutosajan kanssa. Tämä koodi varmistaa, että kaikki koneen tilan muutokset tallennetaan ja kirjataan CSV-tiedostoon, jotta niitä voidaan myöhemmin tarkastella ja analysoida. Lisäksi se varmistaa, että tiedostokoko ei kasva liian suureksi, sillä uusi tiedosto luodaan joka päivä päivämäärän mukaan.

```
66      def __repr__(self): #ilmoittaa terminaaliin onko ajossa vai ei
67          tila = 'Ajossa' if self.running else 'Pysähtynyt'
68          return f'{self.nimi}:{tila}'
69
```

Kuva 8. Sovelluksen terminaalikohta

Kuvassa 8 koodiin lisättiin kohta, jossa koodi kertoo terminaalille, onko työstökone käynnissä vai ei. Terminaali tarkoittaa VSCodessa osaa, johon tieto tulostuu. Tämä kohta on tarkoitettu vain tarkistamaan, jos koodiin tehtiin muutoksia. Tällä pystyttiin tarkistamaan, miten Python-koodin muutokset vaikuttivat yhteyksiin ja muuttujiin. Lopullisesta sovelluksesta tämä kohta poistettiin, kun sovellus varmasti toimi halutulla tavalla.

Ennen kuin uusi aikatieto tallennetaan CVS-tiedostoon tietokone käy tarkistamassa tiedosta edellisen ajan ja jatkaa ajan laskemista vanhasta ajasta (kuva 9). Jos vanhaa CVS-tiedostoa ei löydy, koodi luo uuden CVS-tiedoston ja jatkaa siihen kirjoittamista.

```python
#csv tiedoston luonti, koska helppo laittaa exceliin ja työnjohdon/ ohjelmoinnin on helpompi lukea sitä.
def lue_aiempi_aika(tiedostonimi, kone): #lataa viimeksi tallennetun ajan ja jatkaa siitä eteenpäin.
    klo = 0
    try:
        with open(tiedostonimi, 'r') as tiedosto:
            for rivi in tiedosto.readlines():
                r = rivi.split(';')
                if r[2] == kone and float(r[1]) > klo:
                    klo = float(r[1])

    except FileNotFoundError:
        pass

    return klo
```

Kuva 9. Sovelluksen kohta vanhan ajan hakemisesta

Kuvassa 9 koodin kohta on Python-funktio, nimeltään lue_aiempi_aika, joka on suunniteltu lukemaan aikaleimoja sisältävistä tiedostoista. Funktion tarkoituksena on palauttaa viimeksi tallennettu aika tietyn koneen osalta. Toimintatapa voidaan tiivistää seuraavasti: Jos vanhaa aikaa ei löydetä, aikamuuttuja alustetaan nollaksi. Sovellus hakee kaikki tiedostot annetusta polusta. Sovellus etsii uusimman tiedoston nimen tiedostoluettelosta käyttäen tiedoston muokkausaikaa (os.path.getmtime). Sovellus yrittää avata kyseisen tiedoston ja lukea sen rivit. Jos rivillä oleva kone vastaa etsittävää konetta ja aikaleima on suurempi kuin nykyinen tallennettu aika, päivitetään kelloaika ja tilalle tulee uusi aikaleima. Tämä koodin osa auttaa seuraamaan viimeisintä tallennettua aikaa tietyn työstökoneen osalta lukemalla aikaleimoja sisältävistä tiedostoista. Aika päivitetään, jos uudempi aikaleima löytyy.

```python
LOG_TIEDOSTO = '/home/kone_logit/työstökoneidenajat.cvs'
koneet = [
    Kone('SBZ630',[redacted],pyads.[redacted][('P_TwinSafe_SBZ6xx.fbTSPanel.O_bLED_ControlOn', pyads.PLCTYPE_BOOL, True),
                                             ('P_TwinSafe_SBZ6xx.fbTSPanel.O_bLED_AxisOn', pyads.PLCTYPE_BOOL, True),
                                             ('PRG_ANLAGENTEIL3.doEinrichten.O_HGL', pyads.PLCTYPE_BOOL, True)]),
    Kone('SBZ628XLR',[redacted],pyads.[redacted][('ctrl_TwinSafe.LMControl.ObLamp', pyads.PLCTYPE_BOOL, True),
                                             ('ctrl_TwinSafe.LMAxisOn.ObLamp', pyads.PLCTYPE_BOOL, True),
                                             ('PRG_SPS_Ablauf_Anl1.prt_Infeed.fbRollerWay.ObYWorkPos', pyads.PLCTYPE_BOOL, True),
                                             ('.nOverride', pyads.PLCTYPE_INT, True)]),
    Kone('SBZ628XL Kaide',[redacted],pyads.[redacted] [('ctrl_TwinSafe.LMControl.ObLamp', pyads.PLCTYPE_BOOL, True),
                                                ('ctrl_TwinSafe.LMAxisOn.ObLamp', pyads.PLCTYPE_BOOL, True),
                                                ('PRG_SPS_Ablauf_Anl1.prt_Infeed.fbRollerWay.ObYWorkPos', pyads.PLCTYPE_BOOL, True),
                                                ('.nOverride', pyads.PLCTYPE_INT, True)]),
    Kone('SBZ628XL Kisko',[redacted],pyads.[redacted] [('ctrl_TwinSafe.LMControl.ObLamp', pyads.PLCTYPE_BOOL, True),
                                                ('ctrl_TwinSafe.LMAxisOn.ObLamp', pyads.PLCTYPE_BOOL, True),
                                                ('PRG_SPS_Ablauf_Anl1.prt_Infeed.fbRollerWay.ObYWorkPos', pyads.PLCTYPE_BOOL, True),
                                                ('.nOverride', pyads.PLCTYPE_INT, True)]),
    #Kone('Nikopress',[redacted],pyads.[redacted])
]
#Koneet sisälle kerroin minkä muuttuja pitää olla False ja minkä true, että ohjelma alkaa lukemaan työstöaikaa.
#kaikissa koneista piti ottaa kolme muuttujaa,että saa tarkan ajan työstöajasta.

#Jos muuttuja muuttaa TRUE = FALSE silloin ohjelma laskee niin sanottua luppoaikaa.

#Jotta SBZ630 näyttää ohjelmassa 'ajossa' pitää muuttujien P_TwinSafe_SBZ6xx.fbTSPanel.O_bLED_ControlOn olla True eli painettu ja

#P_TwinSafe_SBZ6xx.fbTSPanel.O_bLED_AxisOn akselien muuttuja TRUE eli painettu ja PRG_ANLAGENTEIL3.doEinrichten.O_HGL ulossyttö pöytä tyhjä muuttuja on TRUE

#ctrl_TwinSafe.LMControl.ObLamp # tämä muuttuja on elu sbz628XL koneiden ja sbz628XLR koneen startti painikkeen muuttuja. Start lähtee ajamaan konetta.
#ctrl_TwinSafe.LMAxisOn.ObLamp # tämä muuttuja on elu sbz628xl koneiden ja sbz628xlr koneen akselien painikkeen muuttuja, jos tämä muuttuja ei ole päällä, kone ei lähde ajamaan.

#fbPanel.ofbQuitt.ObLamp kuittaus painike

#PRG_SPS_Ablauf_Anl1.prt_Infeed.fbRollerWay.ObYWorkPos. Tämä muuttuja on ulostulorulla on täynnä muuttuja.
#Tämä piti lisätä koska muutten koneet näyttää ajossa tilaa vaikka koneen,
#saha, puristimet ja kelkka ei aja palikkaa. Eli kone on silloin pysähdyksissä.

#.nOverride on syöttö nopeus. Kaikista koneista löytyi sama muuttuja. Syöttönopeus
```

Kuva 10. Sovelluksen kohta muuttujat

Kuvassa 10 on koodin tärkein osuus. Tässä kohtaa sovellus selvittää, onko koneen muuttuja päällä (true) vai ei päällä (false). Jokaisen työstökoneen muuttujat piti kirjoittaa Python-koodiin, mitä sovellus käy kysymässä koneilta, jotta saataisiin tarkka aikaleima. Jokaisella työstökoneella on kolme muuttujaa, joiden kaikkien piti olla True eli päällä, jotta työstökoneista tulee tieto, että kone on päällä. Jos yksikin muuttuja on False-tilassa, työstökone on pysähtynyt.

```
127  while True:
128      for kone in koneet:
129          time.sleep(2)
130          kone.update()
131          kone.log(LOG_TIEDOSTO)
132
133  #Ohjelma lähtee päälle heti kun kun kone on päällä automaattisesti. Jos kone sammuu tiedostoon tulee tieto että siltä ajan jaksolta ei olla saatu yhteyttä koneisiin.
134
135
136  #Tulostus näkymä kertoo onko kone ollut ajossa ja kauanko ja jos koneen tila muuttuu tulosteeseen tulee tieto Ajossa->Pysähtynyt.
137  #Ohjelma kertoo myös kauanko kone on ollut pysähdyksissä. Tila muuttuu heti kun koneen tila muuttuu Ajossa tilaan.
138
139  #Ohjelma kertoo myös onko yhteyttä saatu koneeseen vai ei.
```

Kuva 11. Sovelluksen kohta viive

Kuvassa 11 esitetään, miten sovellus odottaa kaksi sekuntia jokaisen kierroksen jälkeen ja käy kysymässä työstökoneilta uudet työstöajat. Tämä on silmukka eli loop-komento. Loop-komento mahdollistaa komentojen toistamisen. Loop-komennon avulla voidaan suorittaa sama koodi useita kertoja ilman, että sitä tarvitsee kirjoittaa uudelleen jokaisen toistokerran jälkeen.

4.4 Yhteyden suojaus

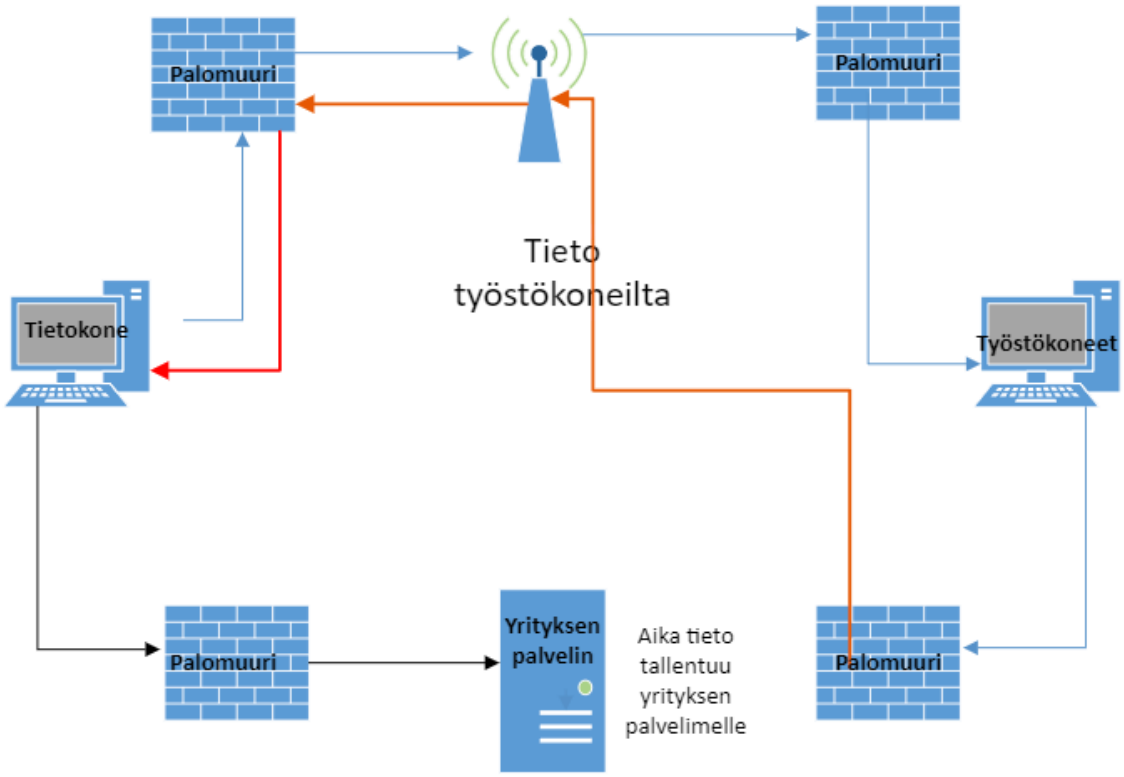
Yhteyden suojaaminen tietokoneen ja työstökoneen välillä on välttämätöntä monista syistä. Tärkein näistä on tietoturva. Suojattu yhteys estää hakkereita pääsemästä käsiksi herkkiin tietoihin, ja salaustekniikat, kuten SSL/TLS, varmistavat tietojen turvallisen siirtymisen. Kasperskyn (i.a.-a) mukaan SSL-varmenne on digitaalinen todennus, joka luo salatun yhteyden verkkosivuston ja käyttäjän välille. Suojausprotokolla SLL, jonka nimi on lyhenne sanoista Secure Sockets Layer, mahdollistaa turvalliset verkkotapahtumat ja suojaa asiakastietojen yksityisyyttä. SSL-varmenteiden käyttö on välttämätöntä yrityksille ja organisaatioille, jotka

haluavat varmistaa verkkosivustojensa turvallisuuden ja suojata asiakkaidensa tietoja. Näin estetään kolmansien osapuolten siirrettävien tietojen lukeminen tai muokkaaminen.

Tietosuojalain 1050/2018 mukaan GDR eli yleinen tietosuoja-asetus velvoittaa yrityksiä suojelemaan henkilötietoja. Tärkeimmät vaatimukset sisältävät laillisen perustelun henkilötietojen käsittelylle, tietosuojaperiaatteiden noudattamisen, tietoturva- ja riskienhallinnan, ilmoitusvelvollisuuden tietoturvaloukkauksista, tietosuojavastaavan nimeämisen tarpeen, tiukat säännöt tietojen siirroille EU:n ulkopuolelle, käyttäjien oikeudet ja vaikutusten arvioinnin. Asetuksen rikkomisesta voi seurata merkittäviä sakkoja. Luottamukselliset tiedot, kuten asiakastiedot ja liiketoimintastrategiat, säilyvät suojatussa yhteydessä salassa. Eheys on toinen tärkeä arvo, suojattu yhteys estää tietojen väärentämisen siirron aikana. Lisäksi vahva tunnistautuminen varmistaa, että vain valtuutetut käyttäjät voivat käyttää yhteyttä.

Lakisääteiset vaatimukset edellyttävät tiettyjä suojaustoimenpiteitä tietojen siirrolle sekä asianmukaiset tietoturva- ja salaustoimenpiteet. Lisäksi yrityksen maine voi kärsiä tietoturvaongelmista, jolloin suojattu yhteys on myös keskeinen osa yrityksen maineen ylläpitämistä.

Yhteyden suojaaminen on siis välttämätöntä tietojen turvallisuuden, luottamuksellisuuden ja eheyden varmistamiseksi. Suojatut yhteydet ovat olennainen osa nykypäivän tietoturvastrategiaa.



Kuva 12. Yhteyden suojaus

Kuvassa 12 havainnollistetaan tiedon suojausta yrityksen tieto- ja työstökoneilla sovellusta käytettäessä. Kuvassa näytetään esimerkki vahvasta tiedonsuojauksesta. Jokainen tieto siirtyy palomuurin läpi jokaisella tiedonsiirron kerralla, mikä varmistaa ylimääräisen suojakerroksen. Lisäksi tietokoneeseen liittyy kaksi salasanaa, jotka on tiedettävä ennen kuin tietokoneeseen edes pääsee. Näin ollen tietokoneeseen pääsy vaatii kaksi erillistä tunnistautumisvaihetta, mikä lisää tietoturvaa.

Tiedonsiirto tapahtuu internetin välityksellä, ja se on suojattu yrityksen hankkimalta tietosuojavaltuutetulta , joka vastaa yrityksen luottamuksellisten tietojen suojauksesta. Tämä varmistaa, että siirrettävät tiedot ovat suojattuja ja täyttävät tietoturvastandardit. Kun tietokone haluaa tiedustella työstökoneilta, onko työstökone käynnissä vai ei, yhteys kulkee palomuurin läpi. Tässä vaiheessa yrityksen tietosuoja tarkistaa, onko tietokoneen osoite turvallinen ja saako se pyytää tietoja työstökoneilta.

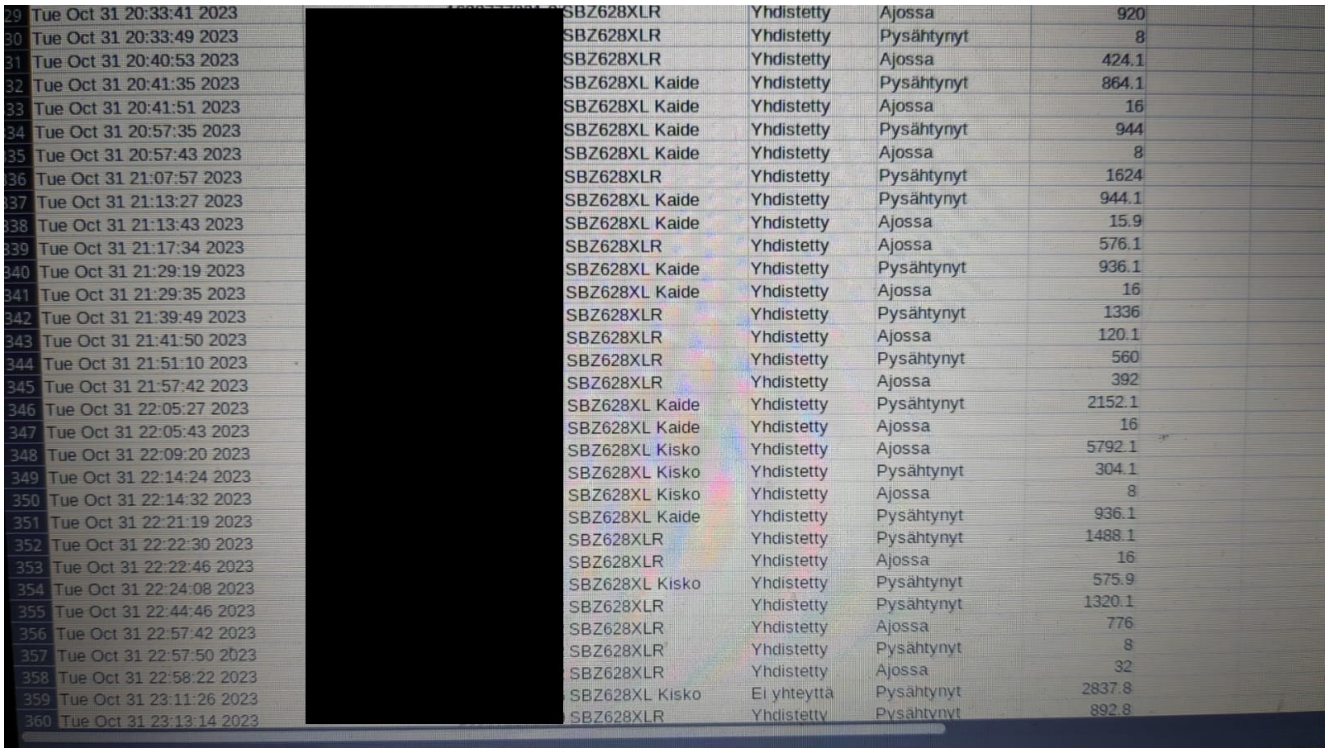
Kun työstökone haluaa välittää tietoa tietokoneelle, myös tämä tieto siirtyy palomuurin läpi. Palomuurissa tapahtuu uusi tietojen tarkistus, mikä varmistaa, että uusi yhteys vastaa turvallisuusmääräyksiä. Tällaiset turvatoimet auttavat estämään ei-toivotun pääsyn ja varmistavat, että tieto siirtyy turvallisesti ja luotettavasti järjestelmien välillä. Tämä kokonaisvaltainen tietoturvarakenne on keskeinen osa organisaation tietoturvaprosesseja ja auttaa suojaamaan arkaluontoisia tietoja mahdollisilta uhilta.

5 TULOKSET

5.1 Tulosten vertailu

Sovellusta testiajettiin monta päivää ja dataa kertyi erilliseen CVS-tiedostoon. Dataa vertailtiin manuaalisesti kellottamalla työstökoneita ja tutkimalla CVS-tiedoston aikoja. Näitä aikoja vertailtiin toisiinsa ja katsottiin, oliko sovelluksen keräämässä datassa virheitä verrattuna manuaalisesti kellotettuun aikaan. Sovelluksen keräämä data oli hyvin lähellä manuaalisesti kellotettua aikaa.

Vertailun yhteydessä huomattiin, että sovellus laskee minuutin päivässä liian vähän aikaa tiedostoon. Tämä johtui siitä, että sovellukseen laitettiin kahden sekunnin viive, joten aikadata ei ole aivan täydellinen. Tätä pohdittiin yrityksen kanssa ja he päätyivät lopputulemana siihen, että heidän tarpeisiinsa kerätty aikadata oli riittävän tarkka. Mikäli tämän epäkohdan sovelluksessa haluaisi korjata, voisi viiveeksi muuttaa 0,5 s tai 1 s. Tämä kuitenkin rasittaisi tietokonetta melko paljon, joten kyseiseen sovellukseen päädyttiin jättämään aluksi määritelty kahden sekunnin viive.



29	Tue Oct 31 20:33:41 2023	SBZ628XLR	Yhdistetty	Ajossa	920
30	Tue Oct 31 20:33:49 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	8
31	Tue Oct 31 20:40:53 2023	SBZ628XLR	Yhdistetty	Ajossa	424.1
32	Tue Oct 31 20:41:35 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	864.1
33	Tue Oct 31 20:41:51 2023	SBZ628XL Kaide	Yhdistetty	Ajossa	16
34	Tue Oct 31 20:57:35 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	944
35	Tue Oct 31 20:57:43 2023	SBZ628XL Kaide	Yhdistetty	Ajossa	8
336	Tue Oct 31 21:07:57 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	1624
337	Tue Oct 31 21:13:27 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	944.1
338	Tue Oct 31 21:13:43 2023	SBZ628XL Kaide	Yhdistetty	Ajossa	15.9
339	Tue Oct 31 21:17:34 2023	SBZ628XLR	Yhdistetty	Ajossa	576.1
340	Tue Oct 31 21:29:19 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	936.1
341	Tue Oct 31 21:29:35 2023	SBZ628XL Kaide	Yhdistetty	Ajossa	16
342	Tue Oct 31 21:39:49 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	1336
343	Tue Oct 31 21:41:50 2023	SBZ628XLR	Yhdistetty	Ajossa	120.1
344	Tue Oct 31 21:51:10 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	560
345	Tue Oct 31 21:57:42 2023	SBZ628XLR	Yhdistetty	Ajossa	392
346	Tue Oct 31 22:05:27 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	2152.1
347	Tue Oct 31 22:05:43 2023	SBZ628XL Kaide	Yhdistetty	Ajossa	16
348	Tue Oct 31 22:09:20 2023	SBZ628XL Kisko	Yhdistetty	Ajossa	5792.1
349	Tue Oct 31 22:14:24 2023	SBZ628XL Kisko	Yhdistetty	Pysähtynyt	304.1
350	Tue Oct 31 22:14:32 2023	SBZ628XL Kisko	Yhdistetty	Ajossa	8
351	Tue Oct 31 22:21:19 2023	SBZ628XL Kaide	Yhdistetty	Pysähtynyt	936.1
352	Tue Oct 31 22:22:30 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	1488.1
353	Tue Oct 31 22:22:46 2023	SBZ628XLR	Yhdistetty	Ajossa	16
354	Tue Oct 31 22:24:08 2023	SBZ628XL Kisko	Yhdistetty	Pysähtynyt	575.9
355	Tue Oct 31 22:44:46 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	1320.1
356	Tue Oct 31 22:57:42 2023	SBZ628XLR	Yhdistetty	Ajossa	776
357	Tue Oct 31 22:57:50 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	8
358	Tue Oct 31 22:58:22 2023	SBZ628XLR	Yhdistetty	Ajossa	32
359	Tue Oct 31 23:11:26 2023	SBZ628XL Kisko	Ei yhteyttä	Pysähtynyt	2837.8
360	Tue Oct 31 23:13:14 2023	SBZ628XLR	Yhdistetty	Pysähtynyt	892.8

Kuva 13. CVS-tiedosto.

Kuvassa 13 on esimerkki siitä, miltä sovelluksen keräämä aikadata näyttää CVS-tiedostossa. Ensin tiedostossa näkyy päivämäärä ja sen jälkeen aikaleima. Seuraavalle riville tulee kyseessä olevan työstökoneen nimi ja tieto työstökoneen käynnissä olosta. Seuraavaksi tiedostoon tulee tieto käynnissäoloajasta, mahdollisista pysähdyksistä ja pysähdysten kestosta. Tiedostoon aika tallentuu vain silloin, kun työstökoneen tila muuttuu. Esimerkiksi, jos työstökone on juuri lopettanut työstettävän profiilin työstön, sovellus tallentaa CVS-tiedostoon uuden ajan. Tiedostoon tallentuu tieto, kauanko kone on ollut pysähdyksissä ja ajossa. Esimerkiksi, jos tiedostossa lukee pysähtynyt 1500 tämä tarkoittaa, että kone on ollut pysähdyksissä 1500 sekuntia.

```
Wed Sep 27 11:43:05 2023; [redacted] SBZ628XL Kaide;Yhdistetty;Pysähtynyt;83002.4
Wed Sep 27 11:43:07 2023; [redacted] SBZ628XL Kisko;Yhdistetty;Pysähtynyt;83180.4
Wed Sep 27 11:44:35 2023; [redacted] SBZ628XL Kaide;Yhdistetty;Ajossa;89.1
Wed Sep 27 11:45:49 2023; [redacted] SBZ630;Yhdistetty;Ajossa;168.0
Wed Sep 27 11:45:51 2023; [redacted] SBZ628XLR;Yhdistetty;Pysähtynyt;167.4
Wed Sep 27 11:45:53 2023; [redacted] SBZ628XL Kaide;Yhdistetty;Ajossa;78.4
Wed Sep 27 11:45:55 2023; [redacted] SBZ628XL Kisko;Yhdistetty;Pysähtynyt;167.4
Wed Sep 27 11:47:36 2023; [redacted] SBZ628XLR;Yhdistetty;Ajossa;105.2
Wed Sep 27 11:48:57 2023; [redacted] SBZ628XLR;Yhdistetty;Pysähtynyt;80.9
Wed Sep 27 11:49:13 2023; [redacted] SBZ628XLR;Yhdistetty;Ajossa;16.2
Wed Sep 27 11:49:41 2023; [redacted] SBZ628XL Kisko;Yhdistetty;Ajossa;226.6
Wed Sep 27 11:50:26 2023; [redacted] SBZ628XLR;Yhdistetty;Pysähtynyt;72.8
```

Kuva 14. Palvelimen aikatieto.

Yrityksen palvelimelle tallentuva aikaleima tieto on hyvin samanlainen (kuva 14) kuin CVS-tiedostossa. Tieto tallentuu palvelimen omaan kansioon, joka on luotu sovellusta varten.

6 YHTEENVETO

6.1 Yhteenveto sovelluksesta

Tässä opinnäytetyössä keskityttiin kehittämään sovellus yritykselle, joka halusi saada aikadataa omilta työstökoneiltaan. Tämän työn tavoite oli saada mahdollisimman tarkkaa aikatieto työstökoneiden toiminnasta, jota voitaisiin hyödyntää markkinoinnissa, huolloissa ja töiden seurannassa. Työ alkoi pohdinnalla siitä, mitä teknologiaa kehitettävässä sovelluksessa olisi hyvä käyttää. Työn alussa vertailtiin eri käyttöjärjestelmien, ohjelmointikielen ja ohjelmointiympäristön etuja ja haittoja.

Tässä työssä käsitellään perusteellisesti sovelluksessa käytettyä Python-ohjelmointikieltä, joka kattaa neljä yrityksen käytössä olevaa CNC-työstökonetta. Sovellus kehitettiin työstökoneille, jotka toimivat Beckhoff-logiikalla. Sovellus on suunniteltu Linux-käyttöjärjestelmälle, joka hyödyntää Python-ohjelmointikielen monipuolisuutta ja ominaisuuksia tehokkaan toteutuksen saavuttamiseksi.

Valitsemalla Linux-käyttöjärjestelmän ja Python-ohjelmointikielen varmistetaan yritykselle järjestelmän tehokkuus ja monipuolisuus. Linux-käyttöjärjestelmä tarjoaa vakauden ja avoimen lähdekoodin edut, kun taas Python-ohjelmointikieli mahdollistaa monipuolisen ohjelmoinnin ja nopeat kehitysprosessit. Tämä yhdistelmä tarjoaa vankan perustan aikatiedon keräämiselle ja siten koko kehitetyn sovelluksen toiminnalle.

Opinnäytetyön tavoitteena oli tarjota yritykselle vankka ja monipuolinen ratkaisu, joka täyttää yrityksen monipuoliset vaatimukset. Seurantaohjelman suunnittelussa ja toteutuksessa otetaan huomioon CNC-työstökoneiden erityispiirteet ja tarpeet, ja valittu tekninen ratkaisu on suunniteltu tukemaan yrityksen toimintaa mahdollisimman tehokkaasti.

6.2 Pohdinta

Tulevaisuudessa sovellusta on mahdollista kehittää ja muokata monin eri tavoin Yksi mahdollisuus on lisätä sovellukseen kohteen nimi, tuotantonumero ja työstettävän tuotteen laji, mikä rikastuttaisi kerättävää tietoa entisestään. Yrityksellä on vuosittain monia tuhansia eri kohteita

ja niille kaikille on eri työnumerot. Jokainen kohde voi vaihdella materiaalin värin, pituuden ja paksuuden mukaan. Yrityksen työstökoneet työstävät satoja eri materiaaleja, joiden työstöaika vaihtelee sekunnista tunteihin. Näiden lisätietojen avulla voidaan saada syvällisempää käsitystä kunkin kohteen ominaisuuksista ja tuotantotiedoista.

Ohjelmaa voidaan muokata monipuolisesti parantaen sen toimivuutta eri näkökulmista. Mahdolliset muutokset voivat liittyä esimerkiksi tiedonkeruuprosessin tehostamiseen, käyttöliittymän käytettävyyden parantamiseen tai analyysityökalujen kehittämiseen. Kehitys- ja muokkausmahdollisuudet ovat lähes rajattomat, ja ohjelman jatkuva parantaminen on avain sen pitkäaikaiseen menestykseen.

On kuitenkin huomioitava, että aikarajoitteiden vuoksi koko sovelluksen toteutus ei valitettavasti tapahtunut suunnitellussa laajuudessaan. Sovelluksen alkuperäinen laajuus piti sisällään juuri mainitut töiden nimet, tuotantonumerot ja materiaalit. Mutta rajallisen ajan takia nämä jouduttiin poistamaan tästä työstä. Tämä voi vaikuttaa siihen, että osa suunnitelluista ominaisuuksista jäi toteuttamatta tai niiden toteutus jäi osittaiseksi. Tästä syystä on tärkeää priorisoida olennaisimmat muutokset ja lisäykset sovellukseen, jotta sen keskeinen tarkoitus säilyy selkeänä ja käyttökelpoisena. Vaikka sovellusta ei saatu tehtyä kokonaan loppuun asti, yritys oli tyytyväinen sovelluksen laajuuteen ja sen nykyiseen versioon.

Sovelluksen kehittämisessä onkin syytä pitää mielessä tasapaino muutosten ja ohjelman selkeyden välillä. Muutoksia tehtäessä on varmistettava, ettei sovellus käy liian monimutkaiseksi tai epäselväksi käyttäjille. Käytettävyyden ja tehokkuuden optimointiin tulee pyrkiä ilman, että sovelluksen perusominaisuudet hämärtyvät. Jos sovellukselta vaaditaan liikaa voi jopa olla, että sen toiminta hidastuu, jolloin sovellus ei välttämättä lue tarpeeksi tarkasti tarvittavaa dataa.

Mikäli yritys pyrkii hankkimaan tarkan aikaleimadatan kaikista työstökoneistaan, jotka käyttävät Siemens-logiikoita, harkittavana vaihtoehtona olisi uusien näyttöjen käyttöönotto työstökoneissa. Tämä päivitys voisi avata mahdollisuuden tehokkaampaan ja tarkempaan ajan seurantaan kaikilta työstökoneilta. On kuitenkin tärkeää huomata, että uusien näyttöjen käyttöönotto voi edellyttää muita muutoksia, kuten ohjelmistopäivityksiä, ja mahdollisesti lisääntyneitä koulutustarpeita käyttäjille.

Ennen päätöksen tekemistä olisi suositeltavaa suorittaa huolellinen arviointi, jotta varmistutaan siitä, että valitut näytöt ovat täysin yhteensopivia nykyisten Siemens -logiikoiden kanssa. Tämä varmistaa, että päivitys sujuu saumattomasti ilman yhteensopivuusongelmia ja mahdollistaa sujuvan aikaleimadatan keräämisen kaikilta työstökoneilta. Samalla on tärkeää ottaa huomioon, että päivitysprosessi saattaa vaatia ylimääräistä resurssien käyttöä, ja siksi on suositeltavaa punnita hyödyt suhteessa kustannuksiin.

Lisäksi yrityksen tulisi huomioida, että uusien näyttöjen käyttöönotto työstökoneisiin voi vaikuttaa myös käyttäjien päivittäiseen työhön. Mahdolliset lisääntyneet koulutustarpeet on otettava huomioon päätöstä tehtäessä, ja varmistaa, että työntekijät ovat valmiita ja kykeneviä hyödyntämään uusien näyttöjen tarjoamia ominaisuuksia. Näin varmistetaan, että päivitys tuottaa odotetut hyödyt ilman merkittäviä haittavaikutuksia työprosessiin.

Voidaan todeta, että vaikka ohjelman toteutusprosessi oli aikarajoitteiden vuoksi lyhyt, sen tulevaisuuden kehittäminen tarjoaa lukuisia mahdollisuuksia. Jatkokehityksen tulisi keskittyä kriittisiin parannuksiin ja lisäyksiin, joilla voidaan edelleen tehostaa ohjelman käyttöä ja antaa arvokasta tietoa yrityksen päätöksenteon tueksi.

LÄHTEET

Beckhoff New Automation Technology. (i.a.). *Beckhoff Information System.* https://infosys.beckhoff.com/

Brihadiswaran, G. (2020). *A Performance Comparison Between C, Java, and Python.* https://medium.com/swlh/a-performance-comparison-between-c-java-and-python-df3890545f6d

Colton De Vos Managed Services. *(5.3.2018). Application Mangement Services (AMS).* https://www.resolutets.com/what-are-application-management-services-ams/

Daelington, D. (2023). *Linux vs.Windows – What are the differences between Linux and Window*. https://recoverit.wondershare.com/computer-tips/linux-vs-windows.html

ESA automation. (1.12.2020). *Linux Operating System in Automation and Robotics.* https://www.esa-automation.com/en/linux-operating-system-in-automation-and-robotics/

Fetisov, E & Birukova, D. (2023). *Python vs Java 2023: The Ultimate Showdown for Business Applications*. https://jaydevs.com/python-vs-java/

Gangwar, M. (3.8.2022). *Top 7 Best Linux Distros For Laptops.* https://www.digitalocean.com/community/tutorials/top-best-linux-distros-for-laptops

Graduda Linux. (i.a.-a.). *Garuda Linux*. https://garudalinux.org/

Kaspersky. *(i.a.-a.). What is an SSL certificate-Definition and Explanation.* https://www.kaspersky.com/resource-center/definitions/what-is-a-ssl-certificate

Klusaitė, L. (9.3.2022). *TCP IP -mikä se on, mihin sitä tarvitaan ja mitä se tekee?* https://nordvpn.com/fi/blog/tcp-ip-protokolla/

Lehmann, S. (2015). pyads. https://pyads.readthedocs.io/en/latest/installation.html

Longshengmfg. (2023). *How long does CNC machining take.* https://medium.com/@longshengmfg/how-long-does-cnc-machining-take-5d173ef3d113

Memel, T. (2021). *5 Reasons Why Linux is More Secure Than Windows*. https://medium.com/codex/5-reasons-why-linux-is-more-secure-than-windows-1d036c3d3324

Morell, J. (2021). *Does More Data EEqual Better Analytics?* https://www.datameer.com/blog/does-more-data-equal-better-analytics/

Neubert, J. (5.3.2019). *What is a PLC? And how do I talk Python to it* [video]. https://speakerdeck.com/jonemo/what-is-a-plc-and-how-do-i-talk-python-to-it

Olumide, S. (25.4.2023). *What is Java Used For in 2023? The Java Programming Language and Java Platform Strengths*. https://www.freecodecamp.org/news/what-is-java-used-for/

Puri, P. (2014). *Why use Linux for Python Development?* https://parbhatpuri.com/why-use-linux-for-python-development.html

Jet Brains. (i.a.). *The Python IDE for Data science and web development.* https://www.jetbrains.com/pycharm/promo/?source=google&medium=cpc&campaign=EMEA_en_WEST_PyCharm_Branded&term=pycharm&content=536947779984&gad_source=1&gclid=Cj0KCQiArrCvBhCNARIsAOkAGcUhP5d7EggrIUcOSXpNH1EiRYwCs7z41e_Cmr6VhXGJ2tiDSpGi-poaAuMXEALw_wcB

Python Software Foundation. (2020). *Python*. https://www.python.org/

Ramakrishnan, M. (21.12.2022). *How PyCharm Can Help You Code Faster and Better Using Python*. https://emeritus.org/blog/coding-what-is-pycharm/

Red Hat. (3.1.2023). *What is Linux.* https://www.redhat.com/en/topics/linux/what-is-linux

Software Testing Help. (2023). *Linux Vs Windows Difference: Which Is The Best Operating System?* Haettu 28.12.2023. https://www.softwaretestinghelp.com/linux-vs-windows/

Staff, C. (21.11.2023). *Python vs. Java: Which Should I Learn.* https://www.coursera.org/articles/python-vs-java

Teamcode. (16.6.2023). *Comparing PyCharm and VSCode for Python Development.* *https://medium.com/@teamcode20233/comparing-pycharm-and-vscode-for-python-development-6de85180d994*

Thomas, M. (2007). *Console Programming.* https://homepages.uc.edu/~thomam/Intro_OOP_Text/Console_Prog.html

Tietosuojalaki 1050/2018. https://finlex.fi/fi/laki/alkup/2018/20181050

Turing. (i.a.). *Your Ultimate Guide To Visual Studio vs Visual Studio Code.* https://www.turing.com/kb/ultimate-guide-visual-studio-vs-visual-studio-code

Visual Studio Code. (i.a.). *Python in Visual Studio Code*. Haettu 28.11.2023 https://code.visualstudio.com/

Whitson, G. (23.5.2023). *Sick of Microsoft? How to make the Switch from Windows to Linux.* *https://uk.pcmag.com/linux/124238/how-to-make-the-switch-from-windows-to-linux*

Yasar, K. (10.3.2023). *Computer numerical control (CNC).* https://www.techtarget.com/searcherp/definition/computer-numerical-control-CNC