SAVONIA

OPINNÄYTETYÖ - **AMMATTIKORKEAKOULUTUTKINTO**
TEKNIIKAN JA LIIKENTEEN ALA

TUOTEJÄLJITETTÄVYYS JA TUNNISTEKOODINLUKU AUTOMAATIOJÄRJESTELMÄÄN

Opinnäytetyö

TEKIJÄ/T: Joonas Heikkinen

SAVONIA-AMMATTIKORKEAKOULU

OPINNÄYTETYÖ
Tiivistelmä

Koulutusala Tekniikan ja liikenteen ala	
Koulutusohjelma/Tutkinto-ohjelma Sähkötekniikan tutkinto-ohjelma	
Työn tekijä(t) Joonas Heikkinen	
Työn nimi Tuotejäljitettävyys ja tunnistekoodinluku automaatiojärjestelmään	
Päiväys 14.12.2020	Sivumäärä/Liitteet 43
Ohjaaja(t) Jari Ijäs, Pasi Lepistö	
Toimeksiantaja/Yhteistyökumppani(t) Insta Automation Oy	

Tiivistelmä

Tuotteiden jäljitettävyys määritellään osana logistiikan hallintaan, joka pyydystää, tallentaa ja välittää tarvittavat tiedot tuotteista toimitusketjun kaikissa vaiheissa, niin että tuote voidaan tarkistaa turvallisuuden ja laadunvalvonnan varmistamiseksi.

Opinnäytetyön toimeksiantajana toimi Insta Automation Oy. Tavoitteena oli luoda Siemens TIA Portal -ympäristössä GS1-tunnistekoodien muodostamiseen sekä jollekin viivakoodilaitteelle viivakoodienluentaan tarvittavat ohjelmalohkot. Ohjelmalohkojen rakenteet ja periaatteet kuvaillaan lohkojen yhteydessä. Teoriassa perehdyttiin tuotteiden jäljitettävyyteen, erilaisiin tunnistamismenetelmiin sekä niihin kuuluviin standardeihin ja ehtoihin.

Opinnäytetyö suoritettiin suunnittelemalla ohjelmalohkot, jotka muodostavat tunnistekoodin. Tunnistekoodit valittiin tähän työhön GS1-standardin yleisimpien käytetyistä sovellustunnuksista. Jotta nämä tunnistekoodit voidaan lukea, niin oli myös luotava viivakoodinluentaan tarkoitettu ohjelmalohko sekä luetulle datalle datalohko, johon tiedot tallennetaan.

Opinnäytetyön lopputuloksena saatiin toteutettua toimivat ohjelmalohkot GS1-tunnistekoodien muodostamista ja viivakoodinluentaa varten sekä lohkojen toiminnan rakenteet ja periaatteet kuvailtiin kommenttien muodossa. Ohjelmalohkot testattiin lopuksi mahdollisimman kattavasti eri ehdoilla, jolloin varmistetaan ohjelman toimivuus.

Avainsanat
tuotejäljitettävyys, GS1-standardi

SAVONIA UNIVERSITY OF APPLIED SCIENCES

THESIS
Abstract

Field of Study Technology, Communication and Transport			
Degree Programme Degree Programme in Electrical Engineering			
Author(s) Joonas Heikkinen			
Title of Thesis Product traceability and reading of an identification code for an automation system			
Date	14 December 2020	Pages/Appendices	43
Supervisor(s)			
Client Organisation /Partners Insta Automation Oy			

Abstract

Product traceability is defined as part of the logistics management that captures, stores and transmits the necessary information about products at all stages of the supply chain so that the product can be inspected to ensure safety and quality control.

The thesis was commissioned by Insta Automation Oy. The purpose of the thesis was to create program blocks in the Siemens TIA Portal environment for generating GS1 identification codes and for some barcode device for barcode reading. The structures and principles of the program blocks are described in the connection of the blocks.

First, the traceability of products, various capture methods and the associated standards and conditions were studied. Then, the thesis was completed by designing the program blocks that form the identification code. Identification codes were selected for this thesis from the most common application identifiers used in the GS1 standard. After that, in order to read these identification codes, a program block intended for barcode reading had to be created. There was also a need to create a data block for the read data in which the data would be stored. Finally, the program blocks were tested as comprehensively as possible under different conditions, ensuring the functionality of the program.

As a result of the thesis, functional program blocks for generating GS1 identification codes and barcode reading were implemented. The structures and principles of the operation of the program blocks were also described in the form of comments directly to the program.

Keywords
product traceability, GS1 standard

ESIPUHE

Haluan kiittää Insta Automation Oy:tä opinnäytetyön aiheesta. Opinnäytetyön aikana Siemens TIA Portal -ohjelmointiympäristössä työskentely sekä SCL-kielen toiminta ja periaatteet tuli tutuiksi. Erityisesti tahdon kiittää työn ohjaajani Markku Toiviaista tärkeistä neuvoista työn etenemisen kannalta sekä tahdon myös kiittää ohjaavaa opettajaani Jari Ijästä.

Kuopiossa 14.12.2020
Joonas Heikkinen

SISÄLTÖ

1 JOHDANTO

Opinnäytetyön tarkoituksena on luoda yleiskäyttöinen ohjelma GS1-tunnistekoodin muodostamiseen ja viivakoodinluentaa varten Siemens TIA Portal -ympäristöön sekä kuvailla lohkojen rakenteet ja periaatteet ohjelmaan. Tarkoituksena on myös tutustua erilaisiin tuotteiden yksilöinti- ja tunnistamistapoihin sekä tuotteiden jäljitettävyyteen lähinnä elintarviketeollisuudessa.

Tuotteiden tunnistaminen ja jäljitettävyys ovat tärkeä osa teollisuuden prosesseja, koska yrityksen on oltava vastuussa valmistamiensa, maahantuomiensa sekä myymiensä tuotteiden turvallisuudesta. Tuotteiden toimitusketjut kasvavat edelleen laajuudeltaan ja monimutkaisuudeltaan sekä yritykset toimivat maailmanlaajuisesti enemmän. Kauppakumppanit, sääntelyviranomaiset ja kuluttajat tarvitsevat ja vaativat lisää tietoa hankkimistaan ja käyttämistään tuotteistaan. Kyky jäljittää raaka-aineiden lähde tuotteen lopulliseen määränpäähän tulee yhä tärkeämmäksi. Jäljitettävyys mahdollistaa tuotteiden näkyvyyden koko toimitusketjussa ja luo peruspalikoita laadun, riskienhallinnan ja muuntyyppisten järjestelmien puitteissa. (GS1)

1.1. Toimeksiantaja

Insta Group Oy on vuonna 1960 perustettu suomalainen perheyritys, joka muodostuu kuudesta toimialayhtiöstä:

- Insta DefSec Oy, joka on puolustus- ja turvallisuusteknologiayhtiö.
- Insta ILS Oy, joka on ilmailulaitteiden ja järjestelmien kunnossapitopalveluiden toimittaja.
- Insta Response Oy, joka on hätä- ja hälytyskeskusratkaisujen ja kansallisen turvallisuuden erityisosaaja.
- Insta Digital Oy, joka on erikoistunut turvalliseen digilisaatioon sekä automaatioon.
- Insta Secrays Oy, joka on erikoistunut kyber-, tietoturva- ja tietosuojakonsultointiin.
- Insta Automation Oy, joka on erikoistunut teollisuus- ja prosessiautomaatioon.

Installa työskentelee yhteensä 1000 ammattilaista. Opinnäytetyöni toimeksiantajana toimi Insta Automation Oy, joka tarjoaa automaatio- ja sähkösuunnittelun, keskusvalmistuksen, asennuksen sekä kunnospito- ja elinkaaripalveluiden osaamista. Yhtiön toimialoina toimivat vesihuolto, elintarvike-, kemian-, energia-, kaivannais-, kappaletavara-, meri-, metsä- sekä prosessiteollisuus. Toimipisteet sijaitsevat eri puolella Suomea. (Insta)

1.2. Opinnäytetyössä käytetyt lyhenteet

CPU	Central Processing Unit. Prosessori.
DCS	Distributed Control System. Hajautettu ohjausjärjestelmä.
EAN	European Article Number. Kansainvälinen tavaranumero.
FC	Function. Funktio.
GIAI	Global Individual Asset Identifier. Käyttöomaisuuden yksilöintitunnus.
GLN	Global Location Number. Yksilöidään yritys, sen sisäisiä toimintoja tai toimipisteitä.
GRAI	Global Returnable Asset Identifier. Kiertävien kuormankantajien tunnus.
GSRN	Global Service Relation Number. Palvelusuhteen yksilöintitunniste.
GTIN	Global Trade Item Number. Numerosarja tuotteen yksilöintiin.
HMI	Human Machine Interface. Ihmisen ja ohjelmoitavan logiikan välisessä kommunikaatiossa käytettävä käyttöliittymä.
I/O	Input/Output. Rajapintoja, joita käytetään tiedonsiirtoon.
PLC	Programmable logic controller. Ohjelmoitava logiikka.
QR	Quick Response. Ruutukoodi.
RFID	Radio Frequency Identification. Radiotaajuinen etätunniste.
RTU	Remote Terminal Unit. Etäpääte yksikkö.
SCADA	Supervisory Control and Data Acquistion. Valvomo-ohjelmisto.
SCL	Structured Control Language. Ohjelmointikieli, joka on rakenteellista tekstiä.
SSCC	Serial Shipping Container Code. Sarjatoimitusyksikkökoodi.
UPC	Universal Product Code. Kansainvälinen tavaranumero.

2 TUOTTEIDEN JÄLJITETTÄVYYS

Jäljitettävyys on liiketoimintaprosessi, jonka avulla jokainen jäljitettävyyskumppani voi tunnistaa jäljitettävissä olevan kohteen toimittajan sekä vastaanottajan. Jäljitettävissä oleva kohde voi olla tuote tai logistiikkayksikkö. Jäljitettävyysjärjestelmän toteuttaminen toimitusketjussa edellyttää, että kaikki osapuolet yhdistävät tuotteiden fyysisen virtauksen niitä koskevien tietojen kulkuun. Jäljitettävyysprosesseja koskevien GS1-standardien hyväksymisellä varmistetaan, että jäljitettävissä olevien kohteiden tunnistamisesta päästään yhteisymmärrykseen. Toimitusketjun jäljitettävyys on kahden toisiaan täydentävän liiketoimintaprosessin nettotulos, jota kutsutaan ulkoiseksi ja sisäiseksi jäljitettävyydeksi. (Bracken;ym., 2015)

2.1. Ulkoinen jäljitettävyys

Kaikki jäljitettävät kohteet on yksilöitävä ja nämä tiedot jaetaan kaikkien asianomaisten toimitusketjukumppaneiden kesken. Kaikkien hierarkitasojen tuotteiden ja lähetyksen tuotetason välinen yhteys on säilytettävä. Näin ollen kaikkien tuotteiden tunnistaminen jäljitettävyyttä varten edellyttää vähintään seuraavaa:

- Yksilöllisen GTIN-koodin (Global Trade Item Number) määrittäminen ja yksittäisen arvon (tuotantopäivän/sarjanumeron) tai eränumeron määrittäminen.
- Yksilöllisen SSCC-koodin (Serial Shipping Container Code) määrittäminen, jos jäljitettävä kohde on logistiikkayksikkö. SSCC-koodi menetelmää voidaan käyttää myös tunnistamaan jäljitettävää nimikettä tuotantoprosessin syötteeksi.

Ulkoisen jäljitettävyyden säilyttämiseksi jäljitettävät nimikkeen tunnistenumerot on ilmoitettava kauppakumppaneille tuoteselosteissa ja niihin liittyvissä papereissa tai sähköisissä liiketoiminta-asiakirjoissa. Tämä yhdistää fyysiset tuotteet jäljitettävyyden kannalta välttämättömiin tietovaatimuksiin. (Bracken;ym., 2015)

2.2. Sisäinen jäljitettävyys

Prosessit, jotka mahdollistavat sisäisen jäljitettävyyden, joita osapuolet ylläpitävät organisaatiossaan, jotta raaka-aineiden tunnistetiedot voidaan liittää valmiisiin tuotteisiin. Kun tuotetta yhdistetään muihin, käsitellään, konfiguroidaan uudelleen tai pakataan uudelleen. Uudella tuotteella on oltava oma yksilöllinen tuotetunnus (GTIN-koodi). Jäljitettävyyden ylläpitämiseksi on säilytettävä tämän uuden tuotteen ja sen alkuperäisten syöttöaineiden välinen yhteys. Kun jäljitettävää tuotetta luodaan, se on liitettävä tiettyyn tuotantoerään. Eränumero on säilyttävä tuotteessa, kunnes tuote on kulutettu. Tätä periaatetta sovelletaan myös silloin, kun jäljitettävä tuote on osa suurempaa pakkaushierarkiaa. (Bracken;ym., 2015)

2.3. Sisäinen ja ulkoinen jäljitettävyys

Päästä päähän -jäljitettävyys edellyttää, että sisäisen ja ulkoisen jäljitettävyyden prosessit toteutetaan tehokkaasti. Jokaisen jäljitettävyyskumppanin olisi pystyttävä tunnistamaan jäljitettävissä olevien tuotteiden lähde sekä vastaanottaja. Tätä kutsutaan ”yksi askel ylös, yksi askel alas” -periaatteeksi. Tämä edellyttää, että toimitusketjun kumppanit keräävät, tallentavat, ja jakavat jäljitetävyyttä varten vähimmäistiedot.

Tehokas jäljitetävyysjärjestelmä koko toimitusketjussa edellyttää:

- Kaikki tuotteet, jotka on jäljitettävä eteen- tai taaksepäin, on tunnistettava maailmanlaajuisesti ja yksilöllisesti.
- Kaikkien toimitusketjun osapuolten olisi toteutettava sekä sisäisiä että ulkoisia jäljitettävyyskäytäntöjä. (Bracken;ym., 2015)

3 GS1-STANDARDI

GS1 on voittoa tavoittelematon kansainvälinen organisaatio, joka kehittää ja ylläpitää maailmanlaajuisesti standardeja liiketoiminnalle. Tunnetuin näistä standardeista on viivakoodi, joka on painettu tuotteiden pinnalle, jota voidaan skannata optisesti.

GS1-standardit on suunniteltu parantamaan toimitusketjujen tehokkuutta, turvallisuutta ja näkyvyyttä. Ne muodostavat yrityskielen, joka yksilöi, tallentaa ja jakaa keskeisiä tietoja tuotteista, sijanneista, resursseista sekä monesta muusta informaatiosta. GS1 avulla yritykset pystyvät tunnistamaan, keräämään ja jakamaan tietoja keskenään, mikä on ratkaiseva osa liiketoimintaa. (GS1)

3.1. GS1-standardijärjestelmä

GS1-standardijärjestelmän tavoitteena on parantaa liiketoimintaprosessien tehokkuutta ja tuottaa kustannussäästöjä automaation avulla, mikä perustuu maailmanlaajuisesti ainutlaatuiseen tunnistamiseen sekä digitaaliseen informaatioon. GS1-standardit voidaan jakaa kolmeen eri ryhmään: yksilöintiin, tunnistamiseen ja tietojen jakamiseen. (The Global Language of Business)

3.1.1 Yksilöinti

Yksilöintiä koskevat GS1-standardit tarjoavat keinot yksilöidä reaalimaailman kokonaisuudet, jotta ne voivat olla loppukäyttäjien tallentamien ja/tai välittämien sähköisten tietojen kohteena. GS1-tunnistestandardeihin kuuluvat yksilölliset tunnistimet, joita voidaan käyttää tietojärjestelmässä, jolla viitataan yksiselitteisesti reaalimaailman yksikköön, kuten kauppanimikkeeseen, logistiikkayksikköön, fyysiseen sijaintiin, asiakirjaan, palvelusuhteeseen tai muuhun yksikköön. (The Global Language of Business)

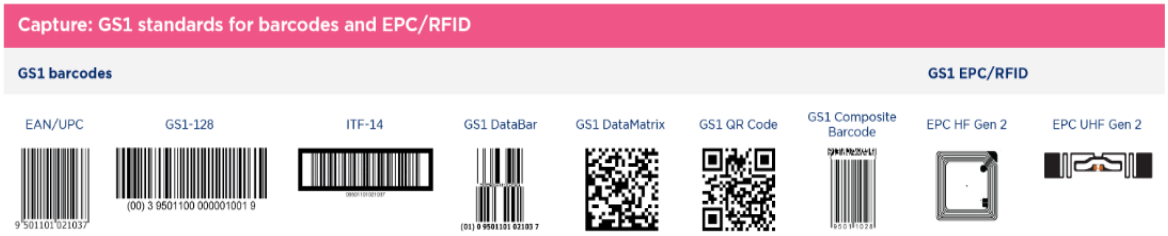
KUVA 1. GS1-standardit yksilöintiä varten (gs1.org)

Jokaiselle kauppatavaralle annetaan yksilöllinen GTIN-koodi (Global Trade Item Number), jotta jälleenmyyjille taataan ainutlaatuinen tapa viitata tiettyyn kauppatavaraan tietojärjestelmiinsä. Jokaisen tuotemerkin omistajan on annettava vain yksi tunniste kauppatavaralleen. (The Global Language of Business)

3.1.2 Tunnistaminen

GS1-standardit tarjoavat keräämistä varten talteenottaa tietoja, jotka liitetään suoraan tuotteisiin tai lähetyksiin. GS1-tietojen tunnistamismenetelmiin kuuluvat:

- Viivakoodi- ja RFID-tunnisteiden määritelmät, joiden avulla GS1-tunnisteavaimet ja lisätiedot voidaan kiinnittää suoraan fyysiseen kohteeseen.
- Standardit, jotka määrittävät yhdenmukaiset rajapinnat lukijoihin, tulostimiin, ja muihin laitteisto- ja ohjelmistokomponentteihin, jotka yhdistävät tietokannat yrityssovelluksiin.



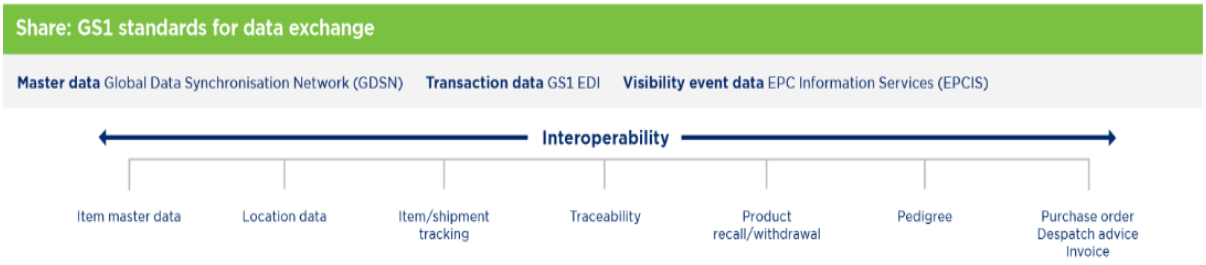
KUVA 2. GS1-standardit viivakoodeille ja radiotaajuisille tunnistimeille (gs1.org)

Jokainen kauppatavara kantaa mukanaan GTIN-koodia tuotepakkauksessa, joka noudattaa GS1-viivakoodistandardeja. GS1-standardien mukaisia tiedonkeruujärjestelmiä käytetään kauppatavaroiden automaattiseen ja luotettavaan tallentamiseen kuljetuksessa toimitusketjun läpi, tuotteen toimituksesta myyntipisteen vastaanottamiseen. (The Global Language of Business)

3.1.3 Tietojen jakaminen

Tiedonvaihtoa koskevat GS1-standardit tarjoavat keinot tietojen jakamiseen sekä kauppakumppaneiden kesken, että sisäisesti, mikä luo perustan sähköisille liiketoimille, fyysisen ja digitaalisen maailman sähköiselle näkyvyydelle sekä muille informaatiosovelluksille. GS1-standardit tietojen jakamiseen ovat:

- Perustietojen, liiketoimintatapahtumatietojen ja fyysisten tapahtumatietojen määritelmät.
- Työkalut online-tuotehaun optimointiin.
- Viestistandardit tietojen jakamiseksi sovellusten ja kauppakumppaneiden välillä.
- Etsintästandardit, jotka auttavat paikantamaan, missä asiaankuuluvat tiedot sijaitsevat eri toimitusketjussa.
- Luottamusstandardit, jotka auttavat määrittämään edellytykset tietojen jakamiselle turvallisella tavalla.



KUVA 3. GS1-standardit tiedonsiirtoa varten (gs1.org)

Jälleenmyyjä saa tuotteen tuotemerkin omistajalta perustiedot, jotka täyttävät GS1-perustietostandardit. Tälläistä tietoa käytetään eri tavoin, esimerkiksi tuotteen kuvauksen näyttämiseen, kun se skannataan myyntipisteessä. Jälleenmyyjä voi käyttää GS1 EDI-standardeja tuotteen uudelleen tilaamiseen valmistajalta, kun tavarat ovat loppumaisillaan.

GS1-näkyvyystapahtumatietojen standardeja voidaan käyttää antamaan yksityiskohtaisia tietoja tapahtumista, kuten siitä, mitä tuotteita kuhunkin myymälään on toimitettu sekä mitä tuotteita poistettu.

GS1-SmartSearchin avulla kuluttajat voivat hakea tarkkoja tuotetietoja verkkohakutulosten ja älypuhelinten skannaussovellusten kautta. Perustietoihin, sähköisiin tapahtumatietoihin ja näkyvyystapahtumatietoihin sovelletaan kaikkia GS1-tietostandardeja ja niissä käytetään GTIN-koodia tai muuta GS1-tunnistusstandardia viitatakseen tarkalleen asianmukaisiin kauppatuotteisiin tai muihin reaalimaailman kokonaisuuksiin. (The Global Language of Business)

4 TALTEENOTTOMENETELMÄT

Teollisuudessa yleisesti käytetään tuotteiden talteenottoon koneellisia tai käsillä luettavia tunnistamistapoja, kuten esimerkiksi QR-koodia, RFID-tunnistetta tai perinteistä viivakoodia. Tämä tekee tuotteiden käsittelystä luotettavampaa sekä nopeuttaa prosessia toisin, kuin manuaalisesti kirjoitetut tunnisteet.

4.1. Viivakoodi

Viivakoodi on koneellisesti luettavissa oleva informaatiomuoto visuaalisella pinnalla. Yksiulotteiset tai lineaariset viivakoodit edustavat systemaattisesti dataa vaihtelemalla rinnakkaisviivojen leveyksiä sekä välejä. Perinteisempiä ja tunnettuja viivakoodityyppejä ovat:

- UPC (Universal Product Code), jota käytetään kulutustavaroiden merkitsemiseen ja skannaamiseen myyntipisteissä ympäri maailmaa – pääasiassa Yhdysvalloissa, Britanniassa, Australiassa, Uudessa-Seelannissa sekä monissa muissa maissa.
- EAN (European Article Number), jota myös käytetään kulutustavaroiden merkitsemiseen maailmanlaajuisesti, mutta pääasiassa Euroopassa. (Scandit Products/Solutions, 2015)

KUVA 4. EAN-viivakoodi (asiakas.gs1.fi)

Yksiulotteisen viivakoodin pituus on suoraan sidottu siihen, kuinka paljon informaatiota se sisältää. Näin ollen käyttäjien on rajoitettava kunkin koodin merkkien määrää. Lineaarisia viivakoodeja käytetään yleisesti yrityksien eri toiminnoissa ajan säästämiseksi sekä varaston työnkulkujen tehostamiseksi. (Scandit Products/Solutions, 2015)

4.2. QR-koodi

QR-koodi eli ruutukoodi, joka tulee sanoista Quick Response (nopea vaste). QR-koodit toimivat samalla tavalla kuin viivakoodit, mutta kaksiulotteisuuden ansiosta tätä voidaan koodata vaaka- että pystysuunnassa, jolloin voidaan tallentaa olennaisempia tietoja moninkertaisesti viivakoodia enemmän. Tämän ansiosta QR-koodiin voidaan tallentaa runsaasti yksityiskohtia, kuten tuotteen kunto,

tuotantopäivä, toimitusaikatiedot, jotka helpottavat yritystä tuotteen jäljittämiseen. Viivakoodin nähden QR-koodin etuna on myös skannauksen nopeus sekä kestävyys. QR-koodeja voidaan edelleen lukea, vaikka koodista olisikin osa tuhoutunut tai sotkeutunut. (Woodford, 2019)

KUVA 5. QR-koodi (qr-koodit.fi)

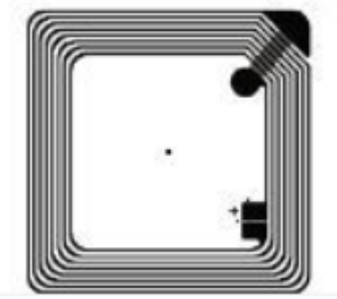
Esimerkiksi QR-koodin varastojärjestelmien avulla voi seurata varastotasoja, rakentaa kokoonpanoja materiaaliluettelon mukaan, seurata valmistusprosesseja ja kerätä reaaliaikaisia tietoja varastoihin mobiililaitteilla. Työntekijät voivat käyttää kameralla varustettua tietolaitetta QR-koodin skannaukseen, lomakkeiden täyttämiseen, valmistusprosessien seurantaan eri vaiheissa ja tietojen etsimiseen. Tietolaitteilla kerätyt tiedot lähetetään turvalliseen, keskitettyyn pilvipaikkaan, ja ne ovat välittömästi kaikkien valtuutettujen käyttäjien saatavilla. (Burkert)

4.2.1 Staattinen ja dynaaminen QR-koodi

QR-koodeja on kahdenlaisia, staattisia sekä dynaamisia. Staattisiin koodeihin tallentamaa sisältöä ei voida jälkeenpäin muokata, sillä se pysyy aina samana. Dynaamisiin koodeihin voidaan tallenttetua sisältöä muokata jälkeenpäin, joten tämä ratkaisu on yleensä pitkäikäisempi sekä antaa enemmän joustavuutta. (Stein, 2020)

4.3. RFID-tunniste

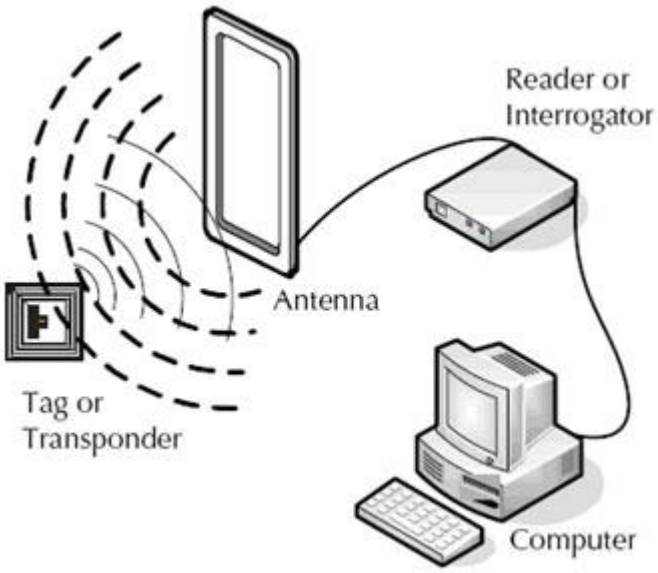
RFID-tunniste eli radiotaajuinen etätunnisus (Radio Frequency Identification) hyöty moniin muihin tunnistemenetelmiin nähden on objektin lukukelpoisuus etänä sekä samalla säilyttäen tietosuojan. Koteloitu tunniste kestää karkeaa käsittelyä, mikä tarkoittaa sitä, että ne voivat säilyttää niiden käytettävyyden jopa kymmeniä vuosia. RFID-tunniste voidaan integroida tuotteeseen joko sen valmistusvaiheessa tai lisätä se myöhemmin kohteeseen teipillä tai tarralla. Perinteiseen viivakoodin verrattuna, RFID-tunniste ei tarvitse suoraa näkökohdetta tuotteen tunnistamiseen esim. pakettien tai laatikoiden kautta sekä RFID-tunnisteita on mahdollista lukea jopa kymmeniä samanaikaisesti. Lisäksi viivakoodiin nähden RFID-tunnistimet kestävät likaisia teollisuusolosuhteita paremmin. RFID-tunnisteiden lukuetäisyydet vaihtelevat sentistä jopa 20 metrin etäisyyksiin. (Atlas RFID Store)



KUVA 6. RFID-tunniste (asiakas.gs1.fi)

4.3.1 RFID:n toiminta

RFID-tunnisteen toiminta perustuu tietojen langattomaan uudelleenkäsittelyyn radioaaltoja käyttävän RFID-lukijan avulla. RFID-tunniste yksinkertaisimmassa muodossa koostuu antennista signaalien lähettämiseen ja vastaanottamiseen sekä sirusta, johon tallennettu tunnisteen tiedot. RFID-tunnisteiden avulla lähetetään tietoja kohteesta radioaaltojen kautta antenni-/lukijayhdistelmään. Energiansa RFID-tunnisteet saavat lukijan tuottamasta radioaalloista esimerkiksi, kun tunniste vastaanottaa lähetyksen lukijasta, energia kulkee sisäisen antennin kautta tunnisteen siruun, jolloin energia aktivoi sirun, joka moduloi energiaa halutulla informaatiolla ja lähettää signaalin takaisin kohti antennia. (Atlas RFID Store)



KUVA 7. RFID-järjestelmän toiminnan periaate
(https://www.autodesk.com/products/eagle/blog/rfid-works-antenna-design/)

4.3.2 Passiiviset, puolipassiiviset ja aktiiviset tunnistimet

RFID-tunnisteet voidaan jakaa kolmeen eri kategoriaan:

- Passiiviset tunnisteet
- Puolipassiiviset tunnisteet
- Aktiiviset tunnisteet

Passiiviset RFID-järjestelmät käyttävät tunnisteita, joilla ei ole sisäistä virtalähdettä, vaan sen sijaan toimivat RFID-lukijalta lähetetyllä sähkömagneettisella energialla. Passiivisia tunnisteita käytetään muun muassa kulunvalvontaan, tiedostojen seurantaan, toimitusketjujen hallintaan, älykkäisiin tarroihin sekä moneen muuhun. Passiivisten RFID-järjestelmien käyttö on edullista monille teollisuuden aloille sen alhaisen hinnan vuoksi. (Smiley, 2019) (Miller, 2019)

Puolipassiiviset tunnistimet sisältävät integroidun piirin, antennin ja akun, mutta ne eivät rajoitu näihin kolmeen osaan. Akun sisällyttäminen puolipassiivisiin tunnisteisiin mahdollistaa tunnisteiden käyttöön lisäominaisuuksia, kuten antureita, reaaliaikaista seurantaa sekä ääni-ilmoituksia. Aktiiviset ja puolipassiiviset tunnisteet eroavat toisistaan siitä, että puolipassiivissa tunnistimista ei löydy lähetintä, jolloin tunnistimen lukualue on rajallinen. (Smiley, 2019) (Miller, 2019)

Aktiiviset tunnistimet koostuvat integroidusta piiristä, antennista, akusta sekä lähettimestä. Lähettimen tarkoituksena on lähettää energiaa suoraan lukijalle, mikä lisää lukualuetta. Lisääntyneen lukualueen lisäksi aktiivisilla tunnistimilla voidaan ottaa käyttöön lisäominaisuuksia, kuten integroituja antureita, lisää muistia sekä enemmän logiikkaa. Tämän takia aktiiviset tunnistimet ovat kalliimpia sekä fyysiseltä kooltaan suurempia kuin passiiviset ja puolipassiiviset tunnistimet. Lisäksi aktiivisenten tunnistimien käyttöikä on lyhyempi, koska ovat akkukäyttöisiä sekä teho on rajallinen. (Smiley, 2019) (Miller, 2019)

5 AUTOMAATIOJÄRJESTELMÄT JA TOIMINNOT

Teollisuusautomaatiolla tarkoitetaan teknologiaa, jossa käytetään ohjausjärjestelmiä sekä -laitteita, kuten tietokoneohjelmistoja ja robotiikkaa, joilla mahdollistetaan teollisten prosessien ja koneiden automaattista toimintaa sekä hallintaa ilman merkittävää ihmisen puuttumista. Teollisuusautomaatio muun muassa vähentää kustannuksia, säästää aikaa, parantaa tuotantonopeutta sekä poistaa inhimillisen virheiden mahdollisuuden. (Unitronics)

Teollisuuden automaatiojärjestelmät voivat olla luonteeltaan hyvin monimutkaisia ja niillä on suuri määrä laitteita, jotka toimivat synkronoituina automaatioteknologioiden kanssa. Automaatiojärjestelmän voi lajitella neljään eri tasoon, jotka ovat kenttälaitteet, ohjausjärjestelmä, tuotannonohjausjärjestelmä ja toiminnanohjausjärjestelmä. (Hibbard, 2019)

5.1. Kenttälaitteet

Automaatiohierarkian alin taso, johon kuuluvat anturit, kamerat, moottorit ja muut toimilaitteet. Kenttälaitteiden päätehtävänä on siirtää prosessien ja koneiden tiedot ohjausjärjestelmään seurantaan sekä analysointia varten. Toimilaitteita käytetään prosessiparametrien ohjaamiseen. (Hibbard, 2019) (Electrical Technology)

Jäljitettävyyden kannalta tuotteen merkitseminen tehdään yleensä hyvin varhaisessa vaiheessa tuotantoprosessissa, jotta kaikkia myöhempiä vaiheita voidaan ohjata tuotteen tunnistetietojen avulla. Tämä mahdollistaa tuotannon tuotteiden tunnistamisen ja jäljittämisen toimituksen jälkeen. Tarkastusjärjestelmät takaavat merkkien luettavuuden koko tuotantoprosessin ajan riippumatta kontaminaatiosta tai erilaisten lukulaitteiden käytöstä. Lisäksi merkin taataan säilyvän koko yrityksen luettavana tuotantoprosessin jälkeen ja tuotteen koko elinkaaren ajan. (Siemens AG, 2019)

5.2. Ohjausjärjestelmä

Ohjausjärjestelmät koostuvat erilaisista automaatiolaitteista, kuten CNC-koneista, PLC:stä, jotka hankkivat prosessiparametrit eri antureista ja toimilaitteista. Automaattiset ohjaimet ohjaavat toimilaitteita käsiteltyjen anturisignaalien ja ohjelma- tai ohjaustekniikan perusteella. Ohjelmoitavat logiikkaohjaimet (PLC) ovat yleisimmin käytetyt teollisuusohjaimia, jotka pystyvät tuottamaan automaattisesti anturin tuloihin perustavia ohjaustoimintoja. Logiikkaohjaimet koostuvat erilaisista moduuleista, kuten CPU:sta, analogisista sekä digitaalisista I/O: ista ja viestintämoduuleista. Ohjausjärjestelmässä on aina myös jonkinlainen paikallinen ja konekohtainen käyttöliittymä, jolla voidaan vaikuttaa suoraan koneen toimintaan. (Hibbard, 2019) (Electrical Technology)

Lukulaitteen ja prosessinohjauksen välinen viestintä suoritetaan vakioliitäntöjen, kuten PROFINETin, Ethernetin ja RS232:n kautta sekä digitaalisten tulojen ja lähtöjen kautta. Lukijat voivat myös käyttää viestintämoduuleja. Tämä mahdollistaa nopean ja turvallisen tiedonsiirron muiden kenttäväylä-protokollien, kuten PROFIBUS ja EtherNet kautta prosessin ohjaukseen liittämistä varten. (Siemens AG, 2019)

5.3. Tuotannonohjausjärjestelmä

Tuotannonohjausjärjestelmään kuuluvat automaattiset laitteet ja valvontajärjestelmät, jotka helpottavat ohjaus- ja säätötoimintoja. Näitä ovat muun muassa HMI (Human Machine Interfaces), DCS (Distribution Control Systems) ja SCADA (Supervisory Control and Data Acquisition) -laitteet erilaisten parametrien seurantaan, tuotantotavotteiden asettamiseen, datahistorian arkistointiin sekä koneiden käynnistykseen ja sammuttamiseen. (Hibbard, 2019) (Electrical Technology)

5.3.1 HMI (Human Machine Interface)

HMI on käyttöliittymä tai kojelauta, joka yhdistää henkilön koneeseen, järjestelmään tai laitteeseen. HMI:tä voidaan teknisesti soveltaa mihin tahansa prosessiin, jonka avulla käyttäjä voi olla vuorovaikutuksessa laitteen kanssa. HMI:tä käytetään yleisimmin teollisen prosessin yhteydessä. Teollisessa ympäristössä hyödynettävät HMI:t ovat useimmiten näyttöjä tai kosketuspaneeleja, jotka yhdistävät käyttäjiä koneisiin, järjestelmiin tai laiteisiin. Tehdasoperaattorit käyttävät HMI-laitteita koneiden sekä niiden tuotantolinjojen ohjaamiseen.

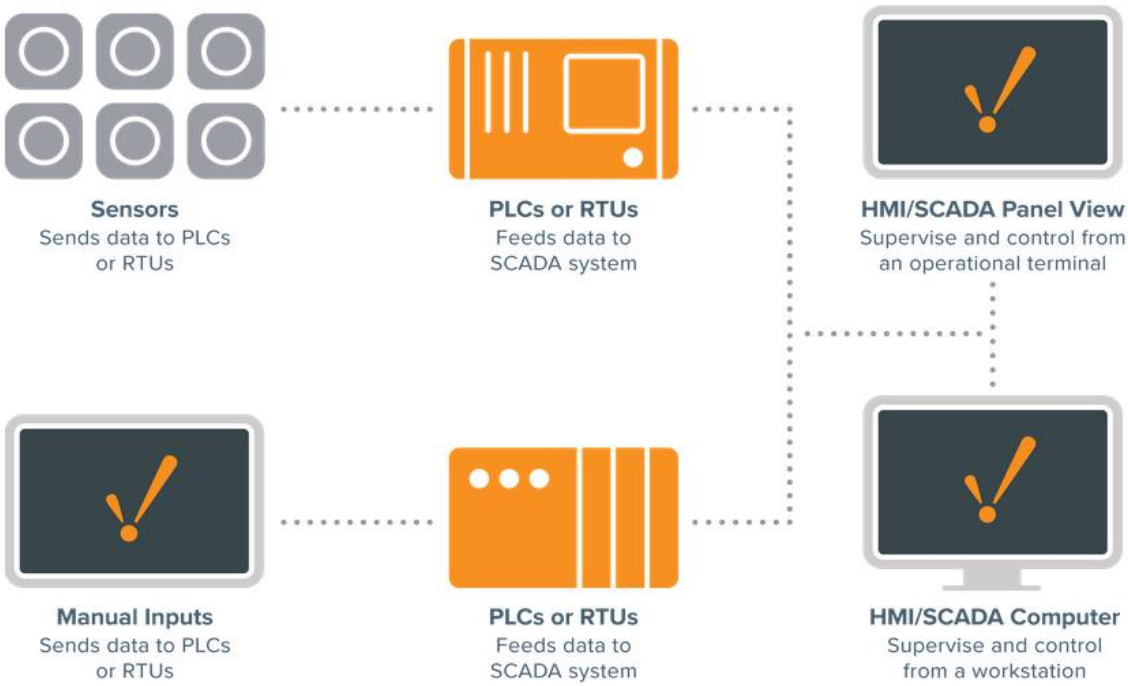
HMI:tä käytetään optimoimaan teollista prosessia digitoimalla ja keskittämällä tietoja katsojalle. Tätä hyödyntämällä operaattorit voivat nähdä tärkeät tiedot, jotka näkyvät kaavioissa, taulukoissa tai digitaalisissa koontinäytöissä, tarkastella ja hallita hälytyksiä sekä muodostaa yhteyden SCADA- ja MES-järjestelmiin yhdellä konsolilla. (Exor, 2019) (Inductive Automation, 2018)

5.3.2 SCADA (Supervisory Control and Data Acquisition)

SCADA on ohjelmisto ja laitteistoelementtien järjestelmä, jonka avulla teolliset organisaatiot voivat:

- Hallita teollisia prosesseja paikallisesti tai etänä
- Seurata, kerätä tai käsitellä reaaliaikaisia tietoja
- Olla suoraan vuorovaikutuksessa laitteiden, kuten antureiden, venttiilien, pumppujen, moottoreiden ja muiden laitteiden kanssa HMI-ohjelmiston avulla
- Tallentaa tapahtumat lokitiedostoon

SCADA-järjestelmät ovat keskeisiä teollisille organisaatioille, koska ne auttavat ylläpitämään tehokkuutta, käsittelemään tietoja älykkäämpiä päätöksiä varten ja kommunikoimaan järjestelmäongelmien kanssa seisokkien lieventämiseksi. SCADA-perusarkkitehtuuri alkaa ohjelmoitavilla logiikkaohjaimilla (PLC) tai etäpääteyksiköillä (RTU). PLC- ja RTU-laitteet ovat mikrotietokoneita, jotka kommunikoivat eri kohteiden, kuten tehdaskoneiden, HMI-laitteiden, antureiden ja päälaitteiden kanssa ja reitittävät näiden objektien tiedot tietokoneisiin, joissa on SCADA-ohjelmisto. SCADA-ohjelmisto käsittelee, jakaa ja näyttää tiedot, auttaa operaattoreita ja muita työntekijöitä analysoimaan tietoja ja tekemään tärkeitä päätöksiä. (Inductive Automation, 2018)



KUVA 8. SCADA-järjestelmän yksinkertaistettu kaavio (inductiveautomation.com)

5.3.3 DCS (Distributed Control System)

Hajautettu ohjausjärjestelmä on erityisesti suunniteltu automaattinen ohjausjärjestelmä, joka koostuu maantieteellisesti hajautetuista ohjauselementeistä laitoksen tai valvonta-alueen yli. DCS:ssä jokaista prosessielementtiä tai konetta tai koneryhmää ohjaa erillinen ohjain. DCS koostuu suuresta määrästä paikkallisia ohjaimia laitoksen valvonta-alueen eri osissa, ja ne on kytketty nopean tietoliikenneverkon kautta.

DCS-valvontajärjestelmässä tiedonkeruu- ja valvontatoiminnot suoritetaan useiden DCS-ohjainten kautta, jotka ovat mikroprosessoripohjaisia yksiköitä, jotka jakautuvat toiminnallisesti ja maantieteellisesti laitoksen yli ja sijaitsevat lähellä aluetta, jolla suoritetaan valvonta- tai tiedonkeruutoimintoja. Nämä ohjaimet pystyvät kommunikoimaan keskenään ja myös muiden ohjaimien, kuten muun muassa valvontapäätteiden ja operaattorien päälaitteiden kanssa.
Hajautetut yksittäiset automaattiohjaimet on kytketty kenttälaitteisiin, kuten antureihin ja toimilaitteisiin. Nämä ohjaimet varmistavat kerättyjen tietojen jakamisen muille hierarkaalisille ohjaimille eri kenttäväylien kautta, kuten muun muassa Profibuksen, HART:in, Profinetin, Modbusin kautta.

DCS soveltuu parhaiten suuriin käsittely- tai valmistuslaitoksiin, joissa on seurattava ja valvottava useita jatkuvia ohjaussilmukoita. Hajautetujen ohjainten ohjaustehtävien jakamisen tärkein etu on se, että jos jokin osa DCS:ssä vikaantuu, laitos voi jatkaa toimintaansa vikaantuneesta osasta riippumatta. (Electrical Technology)

5.4. Valmistuksenohjausjärjestelmä

Valmistuksenohjausjärjestelmä eli MES (Manufacturing Execution System) on tietojärjestelmä, joka tarkkailee ja seuraa valmistettujen tuotteiden tuotantoprosessia tehtaalla. MES:in yleisenä tavoitteena on varmistaa, että valmistustoimenpiteet toteutetaan tehokkaasti tuotannon parantamiseksi. Tämä tavoite saavutetaan seuraamalla ja keräämällä reaaliaikaisia ja tarkkoja tietoja koko tuotannon elinkaaresta.

MES:llä sekä ERP:llä on kyky työskenellä yhdessä, koska molemmat ohjelmistot tuovat erilaisia ominaisuuksia. Niiden käyttäminen yhdessä voi auttaa yritystä saamaan enemmän tasapainoisia tuloksia. Molemmat ohjelmat voidaan integroida, mikä voi lisätä toiminnan selkeyttä ja antaa organisaatioille mahdollisuuden seurata ja mukauttaa suorituskykyä liiketoimintasuunnitelmiin nähden. ERP tietää, miksi päätöksiä pitää tehdä, kun taas MES osaa tehdä ne päätökset. (WorkWise)

5.5. Toiminnanohjausjärjestelmä

Toiminnanohjausjärjestelmä eli ERP (Enterprise Resource Planning) on teollisuuden automaatiohierarkitaso, jonka tehtävänä on hallita koko teollisuuden automaatiojärjestelmää. Toiminnanohjausjärjestelmä käsittelee tuotantosuunnitelua, asiakas- ja markkina-analyysiä sekä tilauksia ja myyntiä. Kyse on siis enemmän kaupallisen toiminnan tukemista kuin teknistä näkökulmaa. Toiminta perustuu tiedonkulkuun eri tasojen väliltä. Kun tiedot etenevät, ne kootaan yhteen, ja kun tiedot laskevat, ne antavat yksityiskohtaista informaatiota prosessista. Toiminnanohjausjärjestelmän integrointi edistää tehokkuutta sekä avoimuutta yrityksessä pitämällä kaikki samalla tietokannalla.

Elintarviketeollisuudessa rakennetulla ERP-ratkaisulla tarkoitetaan tietoa, mistä materiaaleista, mistä työvaiheista, millä koneella, kenen toimesta ja milloin tuote on valmistettu ja mihin se toimitettiin. Jäljitystiedot kerätään sarjanumerolla tai erätasolla vaatimusten mukaan. Tietojen perusteella on mahdollista tunnistaa tuotteet, joissa on käytetty tiettyä materiaalia tai puolivalmistetta, tai materiaalierää, josta tuote on valmistettu. Jäljitystietoja kerätään varastohallinnassa ja valmistuksessa. Tiedonkeruu voidaan rekisteröidä automaattisesti myös tuotantolinjoille. Jäljitettävyys tukee muita liiketoiminnan tarpeita, kuten materiaalihallintaa, jakelua sekä laadun seurantaa. Jos jäljitystietojen keräämistä ei ole integroitu ERP-järjestelmään, materiaali- ja tuotetiedot on poikkeuksetta syötettävä ja haettava eri lähteistä, mikä lisää huomattavasti rutiinityön määrää ja sovellusten välisten rajapintojen tarvetta. (QAD)

6 GS1-TUNNISTEKOODIEN MUODOSTAMINEN JA VIIVAKOODINLUENTA

GS1-tunnistekoodien muodostaminen ja viivakoodinluenta suoritettiin Siemens TIA Portal -ohjelmointiympäristössä, joka oli esiasennettuna virtuaalitietokoneeseen. TIA Portal tarkoittaa täysin integroitua automaatioportaalia, joka on Siemensin kehittämä ohjelmointityökalu teollisuusautomaation suunnitteluun.

Ohjelmalohkot luotiin käyttämällä funktiota eli FC:tä, joilla ei ole erillistä muistia sekä käytettiin myös toimilohkoa viivakoodinluennan toteutukseen. Ohjelmoinnissa käytettiin SCL-kieltä eli Structured Control Language, joka on tekstipohjainen ohjelmointimuoto. SCL-ohjelmointi soveltuu tähän työhön hyvin, koska käsitellään paljon dataa, merkkijonoja sekä matemaattisia operaatioita. Ohjelmalohkojen rakenteitten ja periaatteiden kuvaus tehtiin suoraan ohjelmaan kommenttien muodossa. Lohkojen oikein toimivuuden testaaminen suoritettiin simuloimalla Siemens TIA PLCSim:llä.

Esimerkkisovelluksen tarkoituksena oli saada peruslohkot tunnistekoodien muodostamiseen sekä niiden luentaan jollekin viivakoodinlukijalle. Laitteina työssä esimerkkinä käytettiin S7-1500-perheen logiikkaa sekä Sick -viivakoodinlukijaa, jotka ovat osa Profinet IO-järjestelmää. Kuvassa 9 on esitetty laitteina käytetyt logiikka ja viivakoodinlukija.

KUVA 9. Laitteistomäärittelyn verkkonäkymä työssä käytetyistä laitteista

6.1. Tunnistekoodiohjelmien luominen SCL-kielellä

GS1-tunnistekoodien muodostamisen tarkoituksena on saada jollekin tuotteelle/lähetykselle oma yksilöintitunniste, jotta se voidaan tunnistaa sekä seurata sähköisten sanomien välityksellä. Tunnistekoodien muodostamisessa on otettava GS1-standardien merkkien lukumäärä huomioon. Osalla tunnistekoodilla on tietty määrätty pituus, sekä toisilla voi olla vapaavalintainen lukumäärä 1–30 väliltä. Data tuodaan ohjelmalohkosta ulos sovellustunnuksen kanssa sekä ilman sovellustunnusta ja nämä ovat datatyypiltään merkkijonoja (string). Työhön valittiin yleisimmin käytetyt GS1-standardin mukaiset tunnistekoodit, jotka ovat esitetty kuvassa 10.

Seuraavissa vaiheessa käydään SSCC-koodin, eränumeron, viimeisen käyttöpäivämäärän sekä nettopainon ohjelmalohkojen muodostamiset läpi. Näillä esimerkeillä voidaan loput GS1-tunnistekoodit muodostaa ehtoja ja rakennetta muokkaamalla.

AI/ST	Koko nimi	Muoto*	FNC1 vaaditaan	Datan nimi
00	Sarjatoimitusyksikkökoodi	n2+n18	ei	SSCC
01	Maailmanlaajuinen tuotenumero	n2+n14	ei	GTIN
02	Toimitusyksikköön sisältyvien tuotteiden GTIN numero	n2+n14	ei	CONTENT
10	Eränumero	n2+X..20	ei	BATCH/LOT
11	Valmistuspäivämäärä (VVKKPP)	n2+n6	ei	PROD DATE
13	Pakkauspäivämäärä (VVKKPP)	n2+n6	ei	PACK DATE
15	Parasta ennen päivämäärä (VVKKPP)	n2+n6	ei	BEST BEFORE or BEST BY
17	Viimeinen käyttöpäivä (VVKKPP)	n2+n6	ei	USE BY or EXPIRY
21	Sarjanumero	n2+X..20	kyllä	SERIAL
310n	Nettopaino	n4+n6	ei	NET WEIGHT (KG)
330n	Bruttopaino	n4+n6	ei	GROSS WEIGHT (KG)
37	Toimitusyksikköön sisältyvien tuotteiden määrä	n2+n..8	kyllä	COUNT
400	Asiakkaan ostotilausnumero	n3+X..30	kyllä	ORDER NUMBER
410	Toimitusosoite - GLN (osapuolitunniste)	n3+n13	ei	SHIP TO LOC
414	Fyysisen sijainnin yksilöinti, Global Location Number - osapuolitunniste	n3+n13	ei	LOC No
8003	Maailmanlaajuinen palautettavien pakkausten tunniste	n4+n14+X..16	kyllä	GRAI
8004	Maailmanlaajinen kappaleiden, yksiköiden tunniste	n4+X..30	kyllä	GIAI
8018	Maailmanlaajuinen palvelusuhdenumero	n4+n18	kyllä	GSRN

KUVA 10. GS1-standardin yleisimmät käytetyt sovellustunnukset (GS1 AISBL, 2020)

6.1.1 SSCC-koodin rakenne ja periaatteet

SSCC-koodi (Serial Shipping Container Code) eli sarjatoimitusyksikkökoodi, jota käytetään logistiikkayksiköiden tunnistamiseen ja jonka avulla yritykset voivat tätä seurata tehokkaasti läpi koko tilaus-toimitusketjun. SSCC-koodi tarjoaa maailmanlaajuisesti toimivan tunnistenumeron logististen yksiköiden yksilöimiseen, sekä antaa yrityksille mahdollisuuden kuvata sähköisten sanomien välityksellä sisällön yksityiskohtaisesti ja jakaa tiedot muille toimijoille. SSCC-koodi voidaan luoda GS1-128 viivakoodille tai RFID-tunnisteelle.

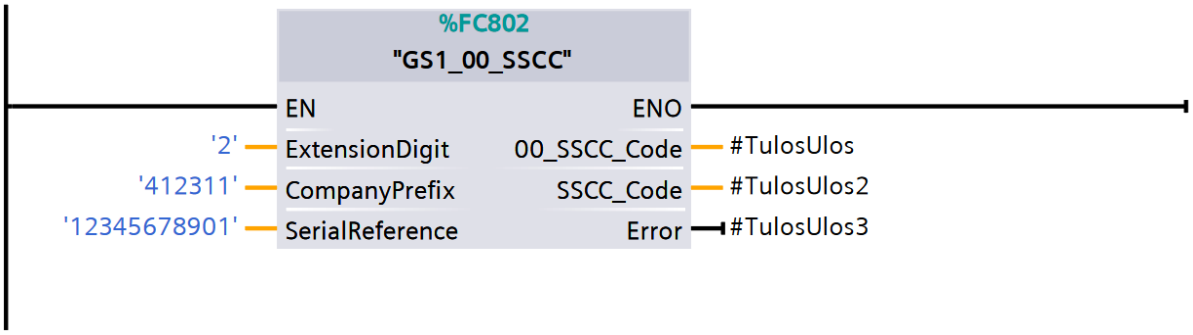
SSCC-koodi muodostuu yhteensä 18 merkistä. Viivakoodia käytettäessä tai SSCC-koodia välittäessä tarvitaan koodin alkuun sovellustunnus, joka on 00. Sovellustunnus kertoo, että kyseessä on sarjatoimitusyksikkö. Seuraavana on laajennustunnus, joka on vapaavalintainen luku arvoltaan 0–9 vä-

liltä. Laajennustuksen jälkeen tulee yrityksen GS1-tunniste, joka voi olla 6-, 7-, 9-, 10- tai 11-numeroa pitkä. Seuraavaksi tulee sarjanumero, jonka yritys itse valitsee. Sarjanumeron merkkien määrä riippuu yritystunnisteen pituudesta. Lopuksi tulee tarkistusnumero, joka lasketaan modulo 10 laskentasäännön mukaan. (GS1 AISBL, 2020)

GS1 Application Identifier	SSCC (Serial Shipping Container Code)																
	Extension digit	GS1 Company Prefix →									Serial reference ←						Check digit
0 0	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}	N_{15}	N_{16}	N_{17}	N_{18}

KUVA 11. SSCC-koodin rakenne (GS1 AISBL, 2020)

Kuvan 12 esimerkissä SSCC-koodin tuloina toimivat laajennustunnus, yritystunnistus, sarjanumero sekä tarkistusnumero, joka laskettiin modulo 10 laskentasäännöllä. Lähtöinä on SSCC-koodi sovellustunnuksen kanssa sekä ilman ja virhe, jos tunnistekoodi on vääränpituinen. Tulojen ja lähtöjen datot ovat tyypiltään merkkijonoja (string), paitsi error, joka on bool. Virheilmoitus nousee ylös, kun havaitaan virhe tunnistekoodin muodostamisessa.



KUVA 12. Ohjelmalohko SSCC-koodin muodostamisesta

6.1.2 SSCC-koodin luominen SCL-kielellä

SSCC-koodin luominen aloitetaan lukemalla annettujen yritystunnisteen sekä sarjanumeron pituus muuttujiin, joiden avulla voidaan vertailla ehtojen toteutumista IF-komennolla.

Kuvista 13 ja 14 huomaamme, kuinka tässä tapauksessa IF-komennolla toteutetaan yritystunnisteen sekä sarjanumeron pituus GS1-standardin ehtojen mukaisesti. Esimerkkinä käydään ensimmäinen ehto, jos yritystunnisteen pituus on yhdeksän merkkiä ja sarjanumeron pituus on seitsemän merkkiä niin ehto toteutuu. Kootaan nämä apumuuttujaan, joka kerää koodin yhteen, jotta myöhemmässä vaiheessa saadaan kokonainen tunnistekoodi ulos. Tämä muuttuja on tyypiltään string. Muuttujaan lisätään ensimmäisenä laajennustunnus, joka oli luku 0–9 väliltä, jonka jälkeen voidaan lisätä yritystunniste sekä sarjanumero.

```
//Määritellään yritystunnisteen pituus ja sen mukaan määritellään jäljelle jäävän sarjanumeron pituus
//Jos yritystunnisteen pituus on yhdeksän merkkiä niin sarjanumero on oltava seitsemän merkkiä pitkä
IF #StringLenght = 9 AND #StringLenght2 = 7 THEN

    //Lisätään laajennustunnus ja yritystunniste
    #SSCC_koonti := CONCAT(IN1 := #ExtensionDigit, IN2 := #CompanyPrefix);

    //Lisätään sarjanumero
    #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #SerialReference);

    //Jos yritystunnisteen pituus on seitsämän merkkiä niin sarjanumeron on oltava yhdeksän merkkiä pitkä
ELSIF #StringLenght = 7 AND #StringLenght2 = 9 THEN

    //Lisätään laajennustunnus ja yritystunniste
    #SSCC_koonti := CONCAT(IN1 := #ExtensionDigit, IN2 := #CompanyPrefix);

    //Lisätään sarjanumero
    #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #SerialReference);

    //Jos yritystunnisteen pituus on 10 merkkiä niin sarjanumeron on oltava kuusi merkkiä pitkä
ELSIF #StringLenght = 10 AND #StringLenght2 = 6 THEN

    //Lisätään laajennustunnus ja yritystunniste
    #SSCC_koonti := CONCAT(IN1 := #ExtensionDigit, IN2 := #CompanyPrefix);

    //Lisätään sarjanumero
    #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #SerialReference);
```

KUVA 13. Yritystunnisteen ja sarjanumeron pituuden määrittäminen

Kuvasta 14 näemme, jos yritystunnisteen ja sarjanumeron pituuden ehto ei toteudu, niin annetaan tunnistekoodin epäonnistumisen johdoksi virhe.

```
    //Jos yritystunnisteen pituus on 11 merkkiä niin sarjanumeron on oltava viisi merkkiä pitkä
ELSIF #StringLenght = 11 AND #StringLenght2 = 5 THEN

    //Lisätään laajennustunnus ja yritystunniste
    #SSCC_koonti := CONCAT(IN1 := #ExtensionDigit, IN2 := #CompanyPrefix);

    //Lisätään sarjanumero
    #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #SerialReference);

    //Jos yritystunnisteen pituus on 6 merkkiä niin sarjanumeron on oltava 10 merkkiä pitkä
ELSIF #StringLenght = 6 AND #StringLenght2 = 10 THEN

    //Lisätään laajennustunnus ja yritystunniste
    #SSCC_koonti := CONCAT(IN1 := #ExtensionDigit, IN2 := #CompanyPrefix);

    //Lisätään sarjanumero
    #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #SerialReference);

ELSE
    //Jos edelliset ehdot eivät toteudu niin tulos on virheellinen
    #Error := TRUE;
END_IF;
```

KUVA 14. Yritystunnisteen ja sarjanumeron pituuden määrittäminen

Tunnistekoodin loppuun tulee tarkistusnumero, joka lasketaan edellisten arvojen mukaan. Tämä tarkistusnumero lasketaan modulo 10-laskentasäännöllä. Kuvassa 15 tarkistusnumeron muodostaminen aloitetaan hakemalla muut SSCC-koodiin kuuluvat arvot, jotka käydään yksi kerrallaan läpi FOR/NEXT-komennon avulla. FOR/NEXT-silmukka tuottaa toistuvan ohjelmointisilmukan, kunnes silmukan muuttuja on määritellyt asetetut arvot. Tässä tapauksessa haetaan SSCC-koodin ensimmäiset 17 numeroa ja IF-komennolla tarkistetaan, ettei löydy tyhjiä merkkejä välistä. Tyhjien merkkien tilalle lisätään nolla, jos niitä löytyy.

Seuraavaksi muunnetaan merkkijono lukuarvoksi, jotta voidaan suorittaa laskutoimenpiteet. Lukuarvot kerrotaan vuorotellen ensimmäisestä lähtien kolmosella ja joka toinen ykkösellä ja tämä saadaan suoritettua IF-komennolla. Tästä saadut tulokset summataan keskenään ja saadusta summasta vähennetään lähimmästä tätä suuremmasta kymmenellä jaollisesta luvusta, jolloin saadaan tarkistusnumero. Jos tarkistusnumerosta saadaan 10, niin muunnetaan se IF-komennolla nollaksi.

```
75  //Lisätään tarkistusnumero
76  //Määritellään muuttujien arvot
77  #Summa := 0;
78  #Tulos := 0;
79  #Kerroin := 1;
80
81  //Tarkistusnumeron laskeminen modulo 10 laskentasäännöllä
82  //Haetaan indexit 1-17
83 FOR #LukuOsoitin := 1 TO 17 DO
84
85      //Jos löytyy tyhjiä merkkejä niin lisätään niiden tilalle nolla
86      IF #SSCC_koonti[#LukuOsoitin] = ' ' THEN
87          #SSCC_koonti[#LukuOsoitin] := '0';
88      END_IF;
89
90      //Muunnetaan merkkijono lukuarvoksi
91      STRG_VAL(IN:= #SSCC_koonti[#LukuOsoitin],
92               FORMAT:= 0000,
93               P:= 1,
94               OUT=> #ArvoLukuna);
95
96      //Kerrotaan lukuarvot vuorotellen kolmosella ja ykkösellä -> 3/1/3/1..
97      IF #Kerroin = 3 THEN
98          #Kerroin := 1;
99      ELSE
100         #Kerroin := 3;
101     END_IF;
102
103     #Tulos := #ArvoLukuna * #Kerroin;
104
105     //Lasketaan yhteen
106     #Summa := #Summa + #Tulos;
107
108 END_FOR;
109
110 //Lasketaan tarkistusnumero
111 #TarkistusnumeroInt := 10 - (#Summa - ((#Summa / 10) * 10));
112
113 //Jos tarkistusnumero on 10 niin muutetaan se nollaksi
114 IF #TarkistusnumeroInt = 10 THEN
115     #TarkistusnumeroInt := 0;
116 END_IF;
```

KUVA 15. Modulo 10 laskentasäännön luominen

Tämän jälkeen muunnetaan tarkistusnumero string muotoon, jolloin se voidaan lisätä tunnistekoodin loppuun. Lopuksi muodostetaan SSCC-koodi ilman sovellustunnusta sekä sovellustunnuksen kanssa.

```
118  //Muutetaan tarkistusnumero stringiksi
119 VAL_STRG(IN:=#TarkistusnumeroInt,
120          SIZE:=1,
121          PREC:= 0,
122          FORMAT:= 0000,
123          P:= 1,
124          OUT=> #TarkistusnumeroString);
125
126  //Jos tila on error niin tyhjennetään tarkistusnumero
127 IF #Error = TRUE THEN
128     #TarkistusnumeroString := '';
129 END_IF;
130
131  //Lisätään tarkistusnumero SSCC-koontiin
132  #SSCC_koonti := CONCAT(IN1 := #SSCC_koonti, IN2 := #TarkistusnumeroString);
133
134  //Muodostetaan SSCC-koodi ilman sovellustunnusta
135  //Tulos ulos
136  #SSCC_Code := #SSCC_koonti;
137
138  //Muodostetaan SSCC-koodi sovellustunnuksella
139  //Jos tila on error niin tyhjennetään sovellustunnus
140 IF #Error = TRUE THEN
141     #"00_SSCC_Code" := '';
142 ELSE
143     //Lisätään sovellustunnus 00, jolloin saadaan tulos ulos
144     #"00_SSCC_Code" := CONCAT(IN1 := '00', IN2 := #SSCC_koonti);
145 END_IF;
```

KUVA 16. SSCC-koodin luominen

Ehtoja ja rakennetta muokkaamalla, luodaan samalla periaatteella tässä työssä myös GTIN-, Content-, GLN- sekä GSRN-koodien muodostamisohjelmat, koska näillä on tietty määrätty pituus ja ne ovat rakenteeltaan samantapaiset.

6.1.3 Eränumeron rakenne ja periaatteet

Eränumero tarkoittaa myyntiyksikköjoukkoa. Eränumeron yhteydessä on tuotteen tiedot, joita valmistaja pitää merkityksellisinä sen kauppaerän jäljitettävyyden kannalta, johon merkkijonoa käytetään. Tiedot voivat viitata itse kauppanimikkeeseen tai sen sisältämiin eriin. Numero voi olla esimerkiksi tuotantoerän numero, vuoronumero, koneen numero, aika tai sisäinen tuotantokoodi.

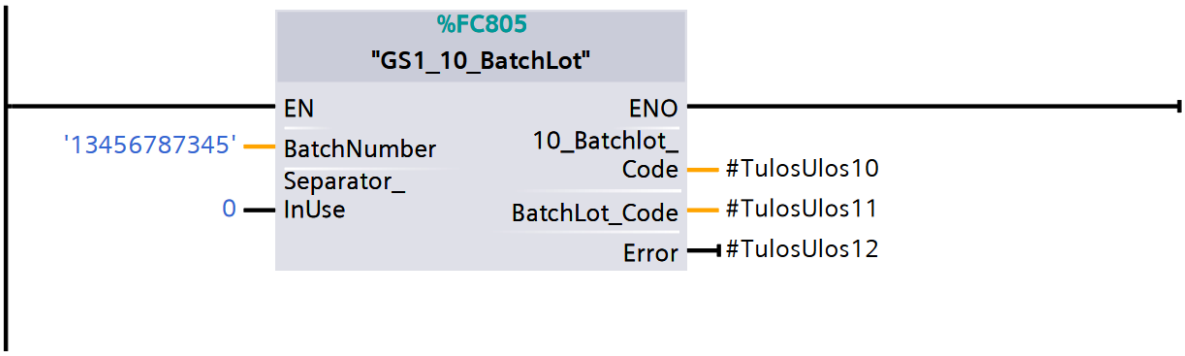
Eränumero koostuu sovellustunnuksesta 10 ja 1-20 vapaavalintaisesta merkistä. (GS1 AISBL, 2020)

GS1 Application Identifier	Batch or lot number
1 0	X_1 ——————→ variable length ——————→ X_{20}

KUVA 17. Eränumeron rakenne (GS1 AISBL, 2020)

Kuvassa 18 tulona on eränumero, joka on 1-20 merkkiä pitkä. Lähtöinä on eränumero sovellustunnuksen kanssa sekä ilman ja virhe, jos tunnistekoodi on yli 20 merkkiä pitkä. Jos eränumeron merkkien lukumäärä on pienempi kuin 20 niin käytetään erotinmerkkiä eränumeron perään, jolloin koodinlukija osaa tulkita merkkijonon päättyvän. Tulojen ja lähtöjen datat ovat tyypiltään merkkijonoja

(string), paitsi erotinmerkki sekä error, jotka ovat tyypiltään bool. Erotinmerkin käytön ilmoitus nostetaan ylös, kun halutaan käyttää tätä tunnistekoodin loppuun ja virheilmoitus nousee ylös, kun havaitaan virhe tunnistekoodin muodostamisessa.



KUVA 18. Ohjelmalohko eränumeron muodostamisesta

6.1.4 Eränumeron luominen SCL-kielellä

Jos eränumeron tai minkä tahansa yksilöinnin, jonka tunnistekoodin lukumäärä on vapaavalintainen, niin lisätään erotinmerkki perään. Erotinmerkki on tunnistekoodissa näkymätön merkki, joka erottelee vapaavalintaisten pituuksien omaavat tunnistekoodit. Kuvassa 19 erotinmerkki muodostetaan IF-komennolla, jos tunnistekoodi valitaan pienemmäksi kuin 20 merkkiä. Jos halutaan erotinmerkkiä käyttää, IF-komennolla nostetaan erotinmerkin käytön ilmoitus ylös, jolloin tämä kirjoitetaan apumuuttujaan. Apumuuttuja lisätään tunnistekoodin loppuun, kun muodostetaan eränumero, joka on pienempi kuin 20 merkkiä. Seuraavaksi luetaan tunnistekoodin pituus ja IF-komennon avulla käydään ehdot läpi. Ehtojen toteutuessa tuodaan valmis eränumero ulos tai annetaan virhe, jos ehto ei toteutunut.

```
17  //Jos sarjanumeron merkkien lukumäärä on vähemmän kuin 20 niin käytetään separaattoria, joka "katkaisee" koodin
18 IF #Separator_InUse = 1 THEN
19     //Group Separator
20     #Separator := '$1D';
21 END_IF;
22
23  //Luetaan eränumeron pituus
24  #StringLenght := LEN(#BatchNumber);
25
26  //Muodostetaan eränumero ilman sovellustunnusta
27  //Jos merkkien lukumäärä on suurempi kuin 20 niin virhe
28 IF #StringLenght > 20 THEN
29     #Error := TRUE;
30     #BatchLot_Code := '';
31
32     //Jos eränumero on pienempi kuin 20 ja separaattori päällä
33 ELSIF #StringLenght < 20 AND #Separator_InUse = 1 THEN
34
35     //Tulos ulos
36     #BatchLot_Code := CONCAT(IN1 := #BatchNumber, IN2 := #Separator);
37
38     //Jos eränumero on pienempi kuin 20 ja separaattori pois päältä
39 ELSIF #StringLenght < 20 AND #Separator_InUse = 0 THEN
40
41     //Virhe
42     #Error := TRUE;
43
44     //Jos eränumero on 20 ja separaattori pois päältä
45 ELSIF #StringLenght = 20 AND #Separator_InUse = 0 THEN
46
47     //Tulos ulos
48     #BatchLot_Code := #BatchNumber;
49
50 END_IF;
```

KUVA 19. Eränumeron luominen

Ehtoja ja rakennetta muokkaamalla, tällä samalla periaatteella luodaan tässä työssä myös määrän, tilausnumeron, sarjanumeron, GRAI- tai GIAI-koodien muodostamisohjelmat, koska näillä on vaihteleva koodin pituus ja ne ovat rakenteeltaan samantapaiset.

6.1.5 Viimeisen käyttöpäivämäärän rakenne ja periaatteet

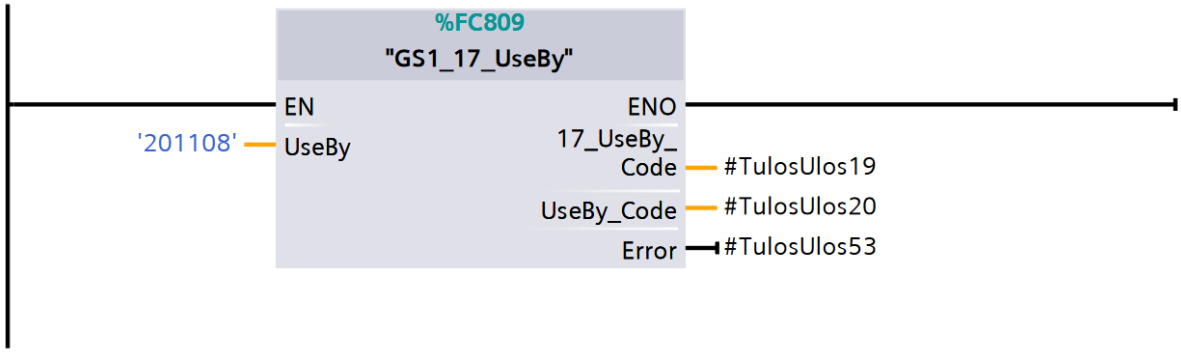
Viimeinen käyttöpäivämäärä on päivä, joka määrittää kulutuksen tai käytön rajan tuotteeseen. Sen merkitys määritetään kauppatavaran kontekstin perusteella esimerkiksi elintarvikkeille päivämäärä osoittaa mahdollisuuden välittömään terveysriskiin, joka aiheutuu tuotteen käytöstä tuotteen jälkeen. Tätä kutsutaan usein "käyttöpäivämääräksi" tai "viimeiseksi säilytyspäivämääräksi".

Viimeinen käyttöpäivämäärä muodostuu sovellustunnuksella, joka on tässä tapauksessa 17 sekä kuudesta merkistä, jotka viittaavat vuoteen, kuukauteen ja päivään. (GS1 AISBL, 2020)

GS1 Application Identifier	Expiration date		
	Year	Month	Day
1 7	N_1 N_2	N_3 N_4	N_5 N_6

KUVA 20. Viimeisen käyttöpäivämäärän rakenne (GS1 AISBL, 2020)

Kuvassa 21 tulona on viimeinen käyttöpäivämäärä, joka muodostuu vuodesta, kuukaudesta ja päivästä ja jokainen näistä on kahden merkin pituinen. Lähtöinä on päivämäärä sovellustunnuksen kanssa sekä ilman ja virhe, jos tunnistekoodi on vääränpituinen. Tulojen ja lähtöjen datat ovat typiltään merkkijonoja (string), paitsi error, joka on tyypiltään bool. Virheilmoitus nousee ylös, kun havaitaan virhe tunnistekoodin muodostamisessa.



KUVA 21. Ohjelmalohko viimeisen käyttöpäivämäärän muodostamisesta

6.1.6 Viimeisen käyttöpäivämäärän luominen SCL-kielellä

Luetaan syötetyn käyttöpäivämäärän pituus ja verrataan tätä ehtoihin IF-komennolla. Jos merkkien määrä on pienempi tai suurempi kuin kuusi, niin annetaan virhe epäonnistuneen päivämäärän muodostamiseksi. Jos päivämäärän merkkien määrä on kuusi, niin tuodaan viimeinen käyttöpäivämäärä ulos ilman sovellustunnusta sekä sovellustunnuksen kanssa.

```
19  //Luetaan viimeisen käyttöpvm pituus
20  #StringLenght := LEN(#UseBy);
21
22  //Muodostetaan viimeinen käyttöpvm ilman sovellustunnusta
23  //Jos merkkien lukumäärä on pienempi tai suurempi kuin kuusi niin virhe
24 IF #StringLenght <> 6 THEN
25      #Error := TRUE;
26      #UseBy_Code := '';
27  ELSE
28      //Jos merkkien lukumäärä on kuusi, saadaan tulos ulos
29      #UseBy_Code := #UseBy;
30  END_IF;
31
32  //Muodostetaan viimeinen käyttöpvm sovellustunnuksella
33  //Jos merkkien lukumäärä on pienempi tai suurempi kuin kuusi niin virhe
34 IF #StringLenght <> 6 THEN
35      #Error := TRUE;
36      #"17_UseBy_Code" := '';
37  ELSE
38      //Jos merkkien lukumäärä on kuusi niin lisätään sovellustunnus 17 ja saadaan tulos ulos
39      #"17_UseBy_Code" := CONCAT(IN1 := '17', IN2 := #UseBy);
40  END_IF;
```

KUVA 22. Käyttöpäivämäärän luominen

Päivämäärän nimeä muokkaamalla, tällä samalla periaatteella luodaan tässä työssä myös tuotanto-, pakkaus- sekä parasta ennen päivämäärien muodostamisohjelmat, koska näillä on syötetyn päivämäärän pituus sekä rakenne samanlaiset.

6.1.7 Nettopainon rakenne ja periaatteet

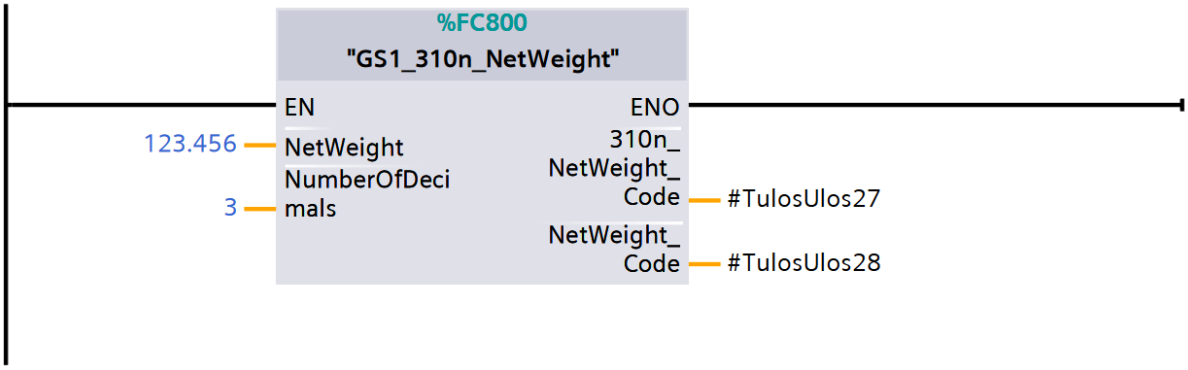
Nettopaino kertoo toimitusyksikköön sisältyvien tuotteiden nettopainon.

Nettopainon rakenne koostuu sovellustunnuksesta 310n ja kuudesta merkistä. Jos nettopainon merkitsee alle kuudella merkillä, niin tyhjät paikat korvataan nollilla. Sovellustunnuksen viimeinen numero ilmaisee oletetun desimaalipilkun sijainnin esimerkiksi luku nolla tarkoittaa, että desimaalipilkkua ei ole, ja numero 1 tarkoittaa, että desimaalin pilkku sijaitsee merkkien viiden ja kuuden välillä. (GS1 AISBL, 2020)

GS1 Application Identifier	Applicable value
A_1 A_2 A_3 A_4	N_1 N_2 N_3 N_4 N_5 N_6

KUVA 23. Nettopainon rakenne (GS1 AISBL, 2020)

Kuvassa 24 tuloina on nettopaino, joka ilmoitetaan kilogrammoina sekä desimaalipisteen sijainnin merkitseminen. Nettopaino merkitään tuloon reaalisena ja desimaalin paikka merkitään kokonaislukuna eli integerinä. Lähtöinä on nettopaino sovellustunnuksen kanssa sekä ilman ja virheestä, jos tunnistekoodi on vääränpituinen. Nettopaino lisätään numeraalisena, joka muunnetaan string muotoon, kun tuodaan tulos ulos. Error on tyypiltään bool ja tämä nousee ylös, kun havaitaan virhe tunnistekoodin muodostamisessa.



KUVA 24. Ohjelmalohko nettopainon muodostamisesta

6.1.8 Nettopainon luominen SCL-kielellä

Kuvassa 25 poistetaan nettopainon reaaliarvosta mahdollinen desimaalin paikka. Laskentaohjelman avulla poistetaan tuloksesta desimaalin paikka, jotta saamme tuloksen merkkijonona ulos. Ensiksi FOR/NEXT-laskentasilmukalla haetaan osoittojan ja desimaalien lukumäärän avulla mahdollinen desimaalipilkun paikka ja tästä saatu tulos saadaan kertoimeen. Esimerkiksi, jos reaaliarvo on 123,456 niin merkataan desimaaleja kolme kappaletta. Kertoimen arvoksi tästä saadaan 1000, joka siirtää desimaalin paikkaa siten, että nettopainoksi saadaan 123456,0. Tämä arvo katkaistaan TRUNC funktiolla DINT-muotoon, jotta nettopaino saadaan kokonaisluvuksi. Tämän jälkeen muutetaan kokonaisluku merkkijonoksi eli stringiksi, jolloin saamme tuloksen ulos ilman desimaaleja. Jos nettopainon merkkijonnon arvoksi jää alle kuusi merkkiä, niin viimeisen FOR/NEXT-silmukan avulla tyhjien merkkien tilalle lisätään nollia, jotka sijoittuisivat vasemmalta lähtien arvon eteen.

```
18  #Multiplier := 1.0;
19 FOR #Index := 1 TO #NumberOfDecimals DO
20      // Nolla ?
21      #Multiplier := #Multiplier * 10.0;
22  END_FOR;
23
24  //Kerrotaan nettopaino kertoimella
25  #NetWeight_Aux := #NetWeight * #Multiplier;
26
27  //Leikataan numeraallinen arvo desimaallista
28  #NetWeightDINT := TRUNC(#NetWeight_Aux);
29
30  //Muunnetaan numeraalinen arvo merkkijonoksi
31 VAL_STRG(IN := #NetWeightDINT,
32          SIZE := 6,
33          PREC := 0,
34          FORMAT := 0000,
35          P := 1,
36          OUT => #ApuString);
37
38  //Lisätään indexien 1-6 tyhjien merkkien tilalle 0
39 FOR #Index := 1 TO 6 DO
40      IF #ApuString[#Index] = ' ' THEN
41          #ApuString[#Index] := '0';
42      END_IF;
43
44  END_FOR;
```

KUVA 25. Nettopainosta desimaalien poistaminen

Kuvassa 26 tuodaan nettopaino ulos ilman sovellustunnusta sekä sovellustunnuksen kanssa. Kun nettopainoa muodostetaan sovellustunnuksen kanssa, niin desimaalipilkun apumuuttuja muutetaan string muotoon, jolloin se voidaan lisätä sovellustunnuksen perään. Lopuksi kokonainen sovellustunnus ja nettopaino tuodaan liitettyinä CONCAT-funktiolla ulos.

```
46  //Muodostetaan nettopaino ilman sovellustunnusta
47  //Nollataan TulosString
48  #TulosString := '';
49
50  //Yhdistetään ApuString data TuloStringiin
51  #TulosString := CONCAT(IN1 := #TulosString, IN2 := #ApuString);
52
53  //Tulos ulos = Nettopaino ilman sovellustunnusta
54  #NetWeight_Code := #TulosString;
55
56  //Muodostetaan nettopaino sovellustunnuksella
57  //Nollataan ApuString
58  #ApuString := '';
59
60  //Muunnetaan desimaalien lukumäärä merkkijonoksi ApuStringiin
61 VAL_STRG(IN := #NumberOfDecimals,
62           SIZE := 1,
63           PREC := 0,
64           FORMAT := 0000,
65           P := 1,
66           OUT => #ApuString);
67
68  //Yhdistetään sovellustunnus 310 sekä ApuStringin data, joka lisää desimaalien määrän
69  #ApuString := CONCAT(IN1 := '310', IN2 := #ApuString);
70
71  //Tulos ulos = Yhdistetään ApuStringin data sekä TulosStringin data, jolloin saadaan nettopaino sovellustunnuksella
72  #"310n_NetWeight_Code" := CONCAT(IN1 := #ApuString, IN2 := #TulosString);
```

KUVA 26. Nettopainon muodostaminen

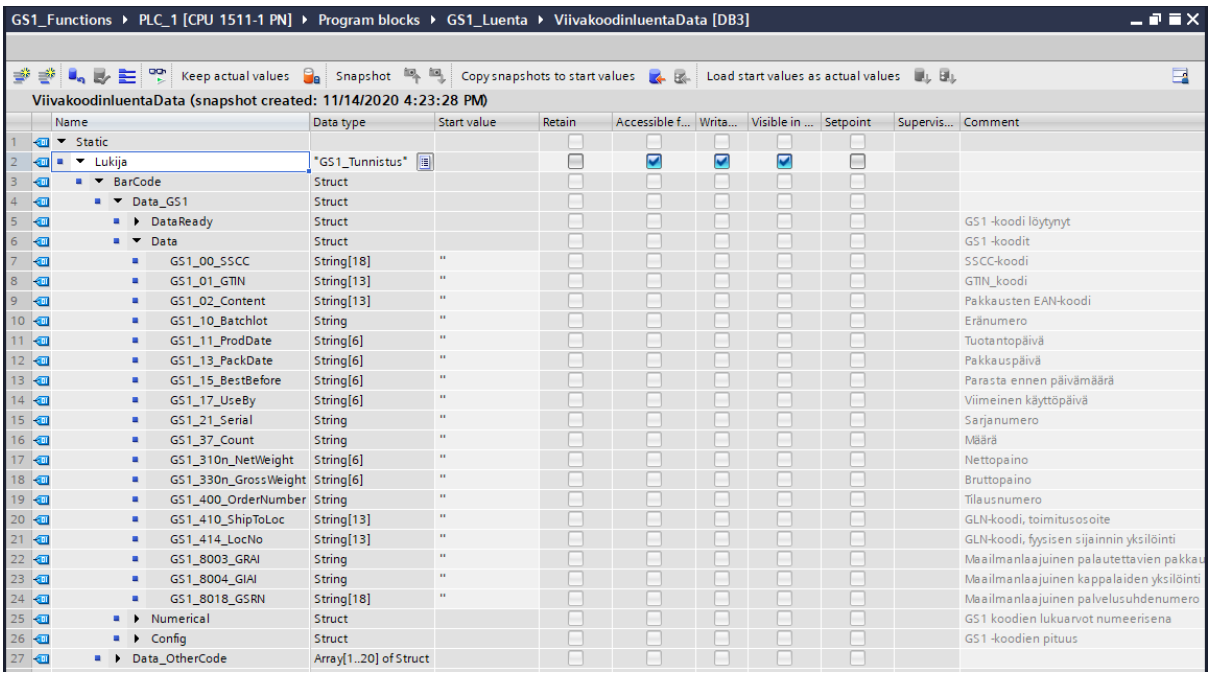
Nettopainon nimeä muokkaamalla, tällä samalla periaatteella luodaan tässä työssä myös bruttopainon muodostamisohjelmat, koska näillä on syötetyn painon pituus sekä rakenne samanlaiset.

6.2. Viivakoodinluentaohjelman rakenne ja periaatteet

Viivakoodinluentaohjelman tarkoituksena on lukea viivakoodinlukijalta luettua GS1-tunnistekoodeja ja hakea näistä ehtojen mukaiset sovellustunnukset ja tallentaa sovellustunnuksien takana olevat datat määrättyyn tietokantaan.

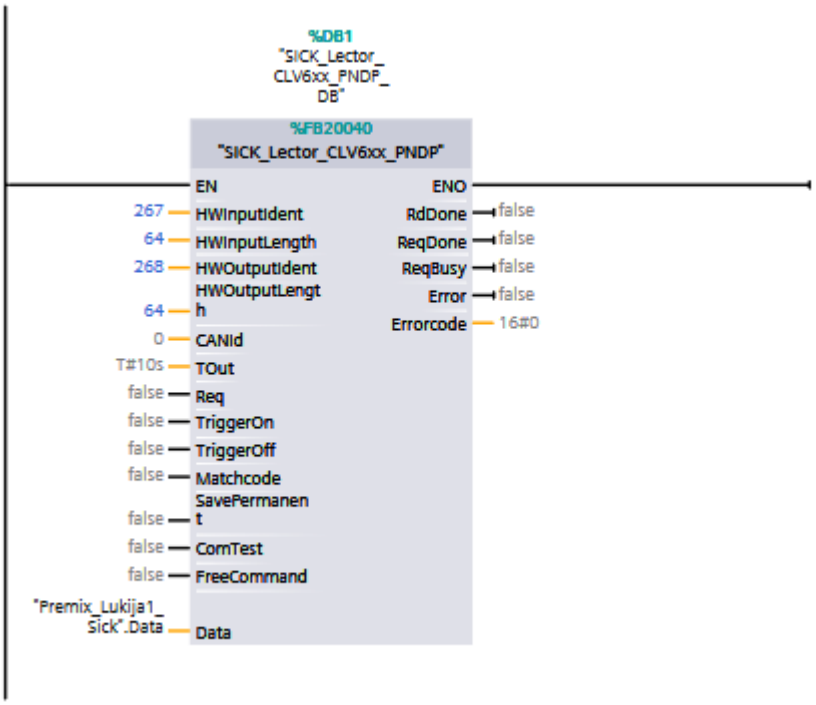
Viivakoodinluennan muodostamisessa on otettava huomioon GS1-standardien tunnistekoodien pituus, koska tunnistekoodit voivat olla vaihtelevan pituisia. Vapaavalintaisten pituuksien tunnistekoodien perästä mahdollisesti löytyy erotinmerkki, joka pitää löytyä luennan yhteydessä, koska tunnistekoodeja voidaan kerätä yhteen merkkijonoon useampia ja ohjelman on tunnistettava nämä. Tunnistekoodit, joilla on vakiopituus, niin eivät tarvitse erotinmerkkiä perään, koska ohjelma osaa lukea nämä määrättyjen merkkien pituuksien mukaan.

Viivakoodinluentaa varten oli tehtävä tunnistekoodeille datalohko, johon tallennetaan luetut datat. Kuvassa 27 on esitetty datalohko, jota käytetään datojen tallentamiseen sekä näiden pituuksien määrittämiseen. ”DataReady” -osion datojen tyyppi on bool, joka kertoo, että mitä tunnistekoodeja on luettu. ”Data” -osion datojen tyyppi on string, jonne luettu data tulee näkymään kokonaisuudessaan. ”Numerical” -osio on numeraallisille datoille, kuten netto- ja bruttopainolle sekä määrälle. Määrän tyyppi on esitetty kokonaislukuna ja netto- ja bruttopaino on esitetty reaalina. ”Config” -osiossa on määritelty datojen pituus, jossa luku nolla tarkoittaa, että kyseisen datan pituus on vapaavalintainen. ”OtherCode” -osioon voidaan tallentaa viivakoodinlukijalle erilaisia ohjauskomentoja, kuten esimerkiksi käynnistys-, lopetus-, hyväksymis- tai peruutuskomentoja.



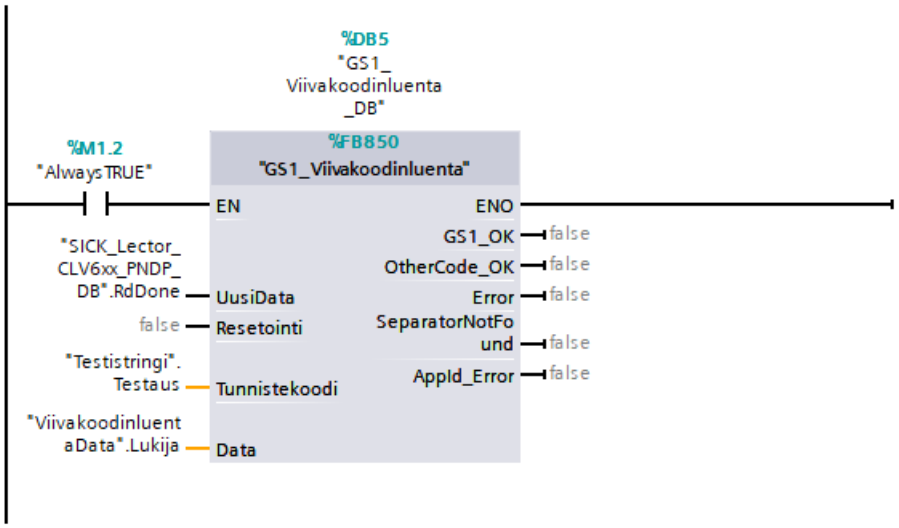
KUVA 27. Viivakoodinluennan datalohko

Kuvassa 28 on esitetty työssä esimerkkinä käytetty Sick -viivakoodinlukijan vakio-ohjelma. Lukijan tulon sekä lähdön pituudet sekä tunnistenumerot on tarkastetteva system constants -osiosta. Tässä tapauksessa lukijan tulon pituus on 64 tavua ja tunnistenumero 267 sekä lähdön pituus on 64 tavua ja tunnistenumero on 268.



KUVA 28. Sick -viivakoodinlukijan vakio-ohjelma

Kuvassa 29 on esitettynä viivakoodinluentaohjelma. Lohkossa tuloina on ”UusiData”, joka toimii triggerinä Sick -viivakoodinlukijan vakio-ohjelmasta. Jos lukijan lähettämä lukutulos vastaanotetaan onnistuneesti, niin bitti nousee pystyyn ja, jos lukutuloksen vastaanotto epäonnistuu, niin lopetetaan luenta suoraan tähän. Resetointi on triggeri, joka nollaa viivakoodinluennan datat sekä lohkon lähdöt. Tunnistekoodin paikalle on linkattu tässä tapauksessa ”Testistringi” datalohko, jotta ohjelman testaaminen onnistui ilman viivakoodinlukijaa ja, jotta tunnistekoodi saatiin syötettyä käsin. Muussa tapauksessa viivakoodinlukijalta tulisi tunnistekoodi. Data on tyypiltään inout, joka käsittelee ”ViivakoodinluentaDatan” datalohkon, joka käsittelyn jälkeen ”sylkäisee” tämän pihalle. Lähtöinä on OK-bitit, jotka kertovat mitä tunnistekoodeja on luettu sekä errorit, jotka kertovat virheen havaitsemiseksi. Virhe voi muodostua, jos erotinmerkkiä tai sovellustunnusta ei löydy tai virhe tulee jossain toisessa vaihessa.



KUVA 29. Viivakoodinluentaohjelma

6.3.1 Viivakoodinluennan luominen SCL-kielellä

Viivakoodinluennan luominen aloitetaan perustoimintojen luomisella. Ensiksi alustetaan lähdöt, jotta vältytään väärien virheiden muodostamiselta. Seuraavaksi luodaan resetoinnin toiminta IF- sekä FOR/NEXT-komentojen avulla, joissa nollataan kaikki datat, kun resetointi on päällä.

Kun luetaan jokin viivakoodi, niin ensimmäisenä tarkistetaan ”muut koodit” osio, että onko tallennettuja komentoja merkitty datalohkoon. FOR/NEXT-komennolla käydään datat läpi yksi kerrallaan ja ehtojen mukaan, jos koodi löytyi, niin ilmoitetaan siitä sekä lopetaan luku. Muussa tapauksessa nollataan ja jatketaan tunnistekoodin purkamista seuraavassa osiossa. Tämä on esitetty kuvassa 30.

```
93  REGION Muu koodi
94      // ************************ MUU KOODI ***********************************
95
96      //Tarkastetaan ensimmäisenä "muu koodit" osio
97      //Haetaan indexit 1-20
98      FOR #Index := 1 TO 20 DO
99
100         //Tarkastetaan tallennetut tunnistekoodit läpi
101         IF #Tunnistekoodi = #Data.BarCode.Data_OtherCode[#Index].Code THEN
102
103             //Koodit löytyi
104             #Data.BarCode.Data_OtherCode[#Index].CodeFound := TRUE;
105             #OtherCode_OK := TRUE;
106         ELSE
107             //Nollataan "muu koodit"
108             #Data.BarCode.Data_OtherCode[#Index].CodeFound := FALSE;
109         END_IF;
110     END_FOR;
111
112     // Jos "muu koodi" löytyi niin lopetetaan
113     IF #OtherCode_OK THEN
114         RETURN;
115     END_IF;
116
117 END_REGION
```

KUVA 30. Muut koodien tarkistaminen

Tunnistekoodit haetaan sovellustunnuksien avulla, joka suoritetaan WHILE-komennon avulla. WHILE-komento on käsky, jossa jokin ehto tai ehdot totetutuu tai ei toteudu. Toteutuessa komento aiheuttaa silmukan, joka toistaa tätä toistuvasti, kunnes toteutusehto täyttyy. WHILE-silmukan sisälle voidaan myös ohjelmoida muita ohjelmasilmukoita muilla silmukkamuuttujilla. Tässä esimerkissä, jos tunnistekoodin pituus on enemmän kuin nolla ja ohjelman tila ei ole virheessä, silmukka toteutuu. Tunnistekoodi on siirretty apumuuttujaan, josta siitä puretaan vasemmalta alkaen kaksi merkkiä, jotka tuodaan sovellustunnuksena ulos omaan muuttujaan ja muunnetaan tämä kokonaisluvuksi. Tämä on esitetty kuvassa 31.

```
133     //Tunnistekoodien luku
134     //Jos saatu tunnistekoodin lukumäärä on suurempi kuin 0 ja erroria ei ole niin luetaan tunnistekoodit
135     WHILE (#StringLenghtCode > 0) AND NOT #Error DO
136
137         //Luetaan tunnistekoodeista sovellustunnus (kaksi merkkiä)
138         #AppIdString := LEFT(IN := #ApuString, L := 2);
139
140         //Sovellustunnuksen tarkistaminen
141         //Jos sovellustunnuksen alku ei vastaa "muu koodit" tai GS1-koodeja niin annetaan virhe
142         IF (#AppIdString[1]<'0') OR (#AppIdString[1]>'9') OR (#AppIdString[2]<'0') OR (#AppIdString[2]>'9')THEN
143             #Error := TRUE;
144             RETURN;
145         END_IF;
146
147         //Muutetaan saatu sovellustunnus numeeriseksi
148         STRG_VAL(IN := #AppIdString,
149                  FORMAT := 0000,
150                  P := 1,
151                  OUT => #AppIdInt);
```

KUVA 31. Sovellustunnuksien lukeminen

Tämän jälkeen voimme poistaa tunnistekoodista äsken haetut sovellustunnuksien merkit. Tässä on myös huomioitava se, että sovellustunnuksien pituus voi olla 2–4 merkkiä, joten IF-komennolla erotetaan samanpituiset sovellustunnukset ja ehdon toteutuessa poistetaan oikea määrä sovellustunnuksien merkkien määrästä. Tämän jälkeen luetaan tunnistekoodi ilman sovellustunnusta sekä hae-

taan erotinmerkkiä tunnistekoodin perästä. Netto- ja bruttopainossa on huomioitava, että sovellustunnuksesta jää neljäs merkki tässä vaiheessa vielä poistamatta. Viimeinen merkki on numero, joka kertoo mahdollisen desimaalin paikan painossa. Tämä merkki poistetaan myöhemmässä vaiheessa, kun luetaan netto- tai bruttopainoa. Sovellustunnuksien hakeminen on esitetty kuvassa 32.

```
153             //Jos sovellustunnuksen kaksi ensimmäistä numeroa viittaavat netto- tai bruttopainoon, sarjanumeroon tai GLN-koodiin
154             IF (#AppIdInt = 31) OR (#AppIdInt = 33) OR (#AppIdInt = 40) OR (#AppIdInt = 41) THEN
155
156                 //Luetaan tunnistekoodeista uudestaan sovellustunnus (kolme merkkiä)
157                 #AppIdString := LEFT(IN := #ApuString, L := 3);
158
159                 //Muutetaan saatu sovellustunnus numeeriseksi
160                 STRG_VAL(IN := #AppIdString,
161                         FORMAT := 0000,
162                         P := 1,
163                         OUT => #AppIdInt);
164
165                 //Erotellaan sovellustunnus tunnistekoodista
166                 #ApuString := RIGHT(IN := #ApuString, L := (#StringLenghtCode - 3));
167
168             //Jos sovellustunnuksen kaksi ensimmäistä numeroa viittaavat GRAI-, GIAI- tai GSRN-koodiin
169             ELSIF #AppIdInt = 80 THEN
170
171                 //Luetaan tunnistekoodeista uudestaan sovellustunnus (neljä merkkiä)
172                 #AppIdString := LEFT(IN := #ApuString, L := 4);
173
174                 //Muutetaan saatu sovellustunnus numeeriseksi
175                 STRG_VAL(IN := #AppIdString,
176                         FORMAT := 0000,
177                         P := 1,
178                         OUT => #AppIdInt);
179
180                 //Erotellaan sovellustunnus tunnistekoodista
181                 #ApuString := RIGHT(IN := #ApuString, L := (#StringLenghtCode - 4));
182
183             ELSE
184                 //Jos edellä olevat ehdot eivät toteudu
185                 //Erotellaan kaksi merkkinen sovellustunnus tunnistekoodista
186                 #ApuString := RIGHT(IN := #ApuString, L := (#StringLenghtCode - 2));
187             END_IF;
```

KUVA 32. Sovellustunnuksen hakeminen

Tunnistekoodien haku suoritetaan CASE-komennolla edellisessä vaiheesta poimituilla sovellustunnuksien arvoilla. CASE-komento on käsky, joka luo monihaaran, joka suorittaa yhden käskyn useista komentosekvenseistä lausekkeen arvosta riippumatta. Lausekkeen arvon oltava kokonaisluku tai bittijono. Kun suoritetaan CASE-komento, lausekkeen arvoa verrataan useiden vakioiden arvoihin. Jos tämä lausekkeen arvo on yhtä suuri kuin vakion arvo, ehto täyttyy ja suoritetaan ehdon takana olevat komennot.

Tässä esimerkissä kuvitellaan, että luetaan viivakoodi, josta löytyy SSCC-koodi eli sarjatoimitusyksikkökoodi. SSCC-koodin sovellustunnus on 00, joilloin siitä saadaan kokonaislukuna nolla. CASE-komennon ehto täyttyy ja suorittaa sovellustunnuksen alla olevan ohjelman. Ensimmäisenä vertailemme datalohkoon tallennettua SSCC-koodin pituutta. Jos vertailussa olevan datan pituus on suurempi kuin nolla, niin siirretään tämän pituus apumuuttujaan, jotta seuraavissa vaiheissa tallennettua dataa ei tulla muuttamaan. Jos datan pituus on nolla, joka tarkoittaa, että tunnistekoodin pituus on vapaavalintainen, niin nostetaan ”FreeLenght” bitti ylös. Seuraavassa vaiheessa tunnistekoodista etsitään erotinmerkkiä. Jos erotinmerkki on löytynyt tunnistekoodista, joko ensimmäisestä tai viimeisestä merkistä, poistetaan tämä. Tämä on esitetty kuvassa 33.

```
205             //Aletaan lukea tunnistekoodit saaduilla sovellustunnuksien avulla
206             CASE #AppIdInt OF
207
208                     // ************* Sovellustunnus 00, SSCC -koodi ************************************
209
210                 0:  //Jos saatu sovellustunnuksen arvo on 0, luetaan SSCC-koodi
211                     //Jos tunnistekoodin pituus on suurempi kuin nolla
212                     IF #Data.BarCode.Data_GS1.Config.Lenght.GS1_00_SSCC > 0 THEN
213
214                         //Luetaan datan pituus apumuuttujaan
215                         #DataLenght := #Data.BarCode.Data_GS1.Config.Lenght.GS1_00_SSCC;
216                     ELSE
217                         //Muuten tunnistekoodilla vapaavalintainen pituus
218                         #FreeLenght := TRUE;
219                     END_IF;
220
221                     //Jos separaattori katkaisee tunnistekoodin niin lopetetaan siihen luku
222                     IF #SeparatorFound AND (#LocSeparator < #DataLenght) THEN
223                         #DataLenght := #LocSeparator - 1;
224                     ELSIF #SeparatorFound AND (#LocSeparator > #DataLenght) THEN
225                         #DataLenght := #LocSeparator - 1;
226                     END_IF;
```

KUVA 33. SSCC-koodin luenta

Jos tunnistekoodi havaittiin, niin nostetaan OK-ilmoitukset ylös sekä puretaan tämä määrättyyn SSCC-koodin dataan. Jos tunnistekoodissa on useampi sovellustunnus, niin poistetaan apumuuttujasta luettu data ja luetaan jäljellä olevan tunnistekoodin pituus. Jos ”FreeLenght” bitti on ylhäällä ja erotinmerkkiä ei löydy tunnistekoodista, niin annetaan virheet sekä nollataan datat. Tämä on esitetty kuvassa 34.

```
228                     //Nostetaan OK bitit ylös
229                     #Data.BarCode.Data_GS1.DataReady.GS1_00_SSCC := TRUE;
230                     #GS1_OK := TRUE;
231
232                     //Puretaan saatu tunnistekoodi dataan vasemmalta alkaen SSCC-koodin pituuteen
233                     #Data.BarCode.Data_GS1.Data.GS1_00_SSCC := LEFT(IN := #ApuString, L := #DataLenght);
234
235                     //Puretaan luettu tunnistekoodi ja luetaan jäljelle jäävän tunnistekoodin pituus
236                     #ApuString := RIGHT(IN := #ApuString, L := (#StringLenghtCode - #DataLenght));
237                     #StringLenghtCode := LEN(#ApuString);
238
239                     //Jos vapaavalintaisessa pituisessa tunnistekoodista ei löydy separaattoria niin virhe
240                     IF #FreeLenght AND NOT #SeparatorFound THEN
241                         #SeparatorNotFound := TRUE;
242                         #GS1_OK := FALSE;
243                         #Data.BarCode.Data_GS1.DataReady.GS1_00_SSCC := FALSE;
244                         #Data.BarCode.Data_GS1.Data.GS1_00_SSCC := '';
245                     END_IF;
```

KUVA 34. SSCC-koodin luenta

Netto- ja bruttopainossa data vielä muutetaan reaaliksi, jotta saadaan mahdollinen desimaalin paikka selville sekä numeraallinen arvo painolle. Ensiksi haetaan tunnistekoodista ensimmäisen merkin paikalta sovellustunnuksesta jätetty viimeinen merkki, joka kertoo missä sijaitsee pilkun paikka. Kun pilkun paikka on määritetty, poistetaan tämä merkki tunnistekoodista, jolloin saadaan paino luettua. Muutetaan data stringistä kokonaisluvuksi ja kokonaisluvusta reaaliseksi. FOR/NEXT-komennon avulla haetaan kerroin, jolla reaalinen paino kerrotaan. Esimerkiksi, jos luetaan nettopainoksi string muodoltaan 0010000, tämä muutetaan kokonaisluvuksi, joka on 10000. Tämä vielä muutetaan reaaliksi, jolloin arvon perään muodostetaan desimaali eli 10000,0. Jos desimaalien määräksi on määritetty kaksi, niin kertoimeksi tästä saadaan 0,01. Lopulliseksi reaaliseksi nettopainoksi saadaan 100,0 kg, kun kerrotaan luettu data 10000,0 sekä kerroin 0,01.

```
653                 //Muutetaan nettopainon data numeeriseksi
654                 STRG_VAL(IN := #Data.BarCode.Data_GS1.Data.GS1_310n_NetWeight,
655                          FORMAT := 0000,
656                          P := 1,
657                          OUT => #WeightInt);
658
659                 //Muutetaan nettopaino reaaliseksi
660                 #WeightReal := LINT_TO_REAL(#WeightInt);
661
662                 //Haetaan nettopainoon mahdollinen pilkun paikka
663                 #Multiplier := 1.0;
664                 FOR #Index := 1 TO #NumberOfDecimals DO
665                     // Nolla ?
666                     #Multiplier := #Multiplier / 10.0;
667                 END_FOR;
668
669                 //Kerrotaan reaalinen nettopaino kertoimella
670                 #Data.BarCode.Data_GS1.Numerical.GS1_310n_NetWeight := #WeightReal * #Multiplier;
```

KUVA 35. Netto- ja bruttopainon muuttaminen numeraaliseksi

6.3. Ohjelmalohkojen testaaminen

Ohjelmalohkot lopuksi testattiin mahdollisimman kattavasti, ehtojen mukaisesti sekä ehtoja muuttamalla, jolloin varmistetaan lohkojen toimivuus. Tässä esimerkissä käydään eränumeron muodostamisen sekä luennan testaaminen. Testaamisen suoritukset ja tulokset merkittiin Excel-tiedostoon.

Muodostamislohkojen testaaminen on esitetty kuvassa 36. Eränumeron pituus voi olla 1-20 merkkiä pitkä. Eränumeron muodostamisessa piti ottaa huomioon erotinmerkin käyttö, jos pituus on vähemmän kuin 20. Eränumeroa muuttamalla ja erotinmerkkiä käyttämällä testattiin lohkon toimivuus. Ehtojen täyttyessä lohko muodostaa eränumeron ilman sovellustunnusta sekä sovellustunnuksen kanssa. Jos ehto ei täyty, niin tulee virheilmoitus ja koodia ei muodosteta.

GS1_10_BATCHLOT:		
Ehdot:	**Tulos:**	**Testattu:**
Jos eränumero = 20	Muodostaa koodin	ok
Jos eränumero > 20 ja separaattori 0	Antaa virheen	ok
Jos eränumero < 20 ja separaattori 0	Antaa virheen	ok
Jos eränumero < 20 ja separaattori 1	Muodostaa koodin	ok
Jos eränumero > 20 ja separaattori 1	Antaa virheen	ok
Koodi	Muodostaa koodin	ok
Koodi + sovellustunnus	Muodostaa koodin	ok

Kuva 36. Eränumeron muodostamislohkon testaaminen

Viivakoodinluennan testaaminen on esitetty kuvassa 37. Testaaminen suoritettiin kahdella eri tavalla, ensiksi testattiin yksittäin ja lopuksi syötettiin monta eri sovellustunnuksella olevaa tunnistekoodia, jolloin ohjelman on osattava luettava ja tunnistettava koodit. Kuvan 37 esimerkissä on esitetty vain pelkästään eränumeron testaaminen ilman muita tunnistekoodeja.

10_BATCHLOT							
Ehdot:					**Tulos:**		**Testattu:**
Jos eränumero on 1-20 separaattorin kanssa					Muodostaa koodin		ok
Jos eränumero on 1-20 ilman separaattoria					Antaa virheen		ok

Kuva 37. Eränumeron luennan testaaminen

7 YHTEENVETO

Opinnäytetyön tavoitteena oli luoda Siemens TIA PORTAL-ohjelmointiympäristössä GS1-standardin yleisimmistä sovellustunnuksista tunnistekoodien muodostamiseen sekä näiden viivakoodinluentaan tarkoitetut ohjelmalohkot sekä dokunmentoida nämä kommenttien muodossa suoraan ohjelmaan. Tarkoituksena oli myös tutustua teoriassa erilaisiin talteenotto- ja yksilöintimenetelmiin sekä tuotteiden jäljitettävyyteen yleisesti sekä myös elintarviketeollisuudessa.

Opinnäytetyön tuloksena ohjelmalohkot saatiin luotua sekä testattua mahdollisimman monilla ehdoilla toimivuuden varmistamiseksi. Ohjelmoinnin aikana periaatteet ja rakenteet dokumentoitiin kommenteilla suoraan ohjelmaan. Yksilöinnin selitteet esitetään jokaisen ohjelmalohkon yhteydessä. Projekteissa voi olla tarvetta tunnistekoodin muodostamiselle sekä viivakoodinlukemiselle, joten tätä työtä voidaan hyödyntää jatkossa. Ohjelmapakettia voidaan muokata myöhemmin tarvittaessa asiakkaiden tarpeitten mukaan.

Opinnäytetyön toteutuksessa ei ilmentynyt suurempia ongelmia, mutta haasteita riitti senkin edestä. GS1-tunnistekoodien muodostaminen onnistui lähes vaivatta. Pieniä muutoksia joutui myöhemmässä vaiheessa tekemään, koska erotinmerkkiä ei otettu huomioon ensimmäisellä kerralla, jolloin tämä puutos korjattiin. Viivakoodinluentaohjelman toteutus oli melko aikaa vievää, koska uusia ongelmia ilmeni toisen ratkettua. Suurin haaste oli erotinmerkin sijainnin hakemisessa tunnistekoodista, jotta ohjelma osaisi lopettaa luennan kyseiseen merkkiin ja lukea seuraavan sovellustunnuksen sekä tallentaa luetut datat oikean pituisena datalohkoon. Työn suunnitteluun ja toteutukseen sai aina tarvittaessa apua sekä ideoita opinnäytetyön ohjaajalta, jonka ansiosta työ saatiin päätökseen.

8 LAINATUT LÄHTEET

asiakas.gs1.fi.

Atlas RFID Store. Rfid beginners guide. *Atlas rfid store sivusto.* [Online] [Viitattu: 23. Heinäkuu 2020.] https://www.atlasrfidstore.com/rfid-beginners-guide/.

Bracken, Jim;Bowler, Des ja D.Buckley. 2015. GS1 Global Meat and Poultry TraceabilityGuideline, Part 1. The GS1 System. *GS1 sivusto.* [Online] Marraskuu 2015. [Viitattu: 26. Heinäkuu 2020.] https://www.gs1.org/docs/traceability/GS1_Global_Meat_and_Poultry_Guideline_Part1_The_GS1_System.pdf.

Burkert. The value of QR codes for fast identification and ordering of OEM parts. *Burkert sivusto.* [Online] [Viitattu: 22. Heinäkuu 2020.] https://www.burkert.co.uk/en/Company-Career/What-s-New/Press/Media/Technical-Reports/Technical-Reports-additional-topics/The-value-of-QR-codes-for-fast-identification-and-ordering-of-OEM-parts.

Electrical Technology. Electrical Technology sivusto. *What is Distributed Control System (DCS)?* [Online] [Viitattu: 3. Elokuu 2020.] https://www.electricaltechnology.org/2016/08/distributed-control-system-dcs.html.

—. What is Industrial Automation, Their Types and Hierarchy of an Industrial Automation System. *Electrical Technology sivusto.* [Online] [Viitattu: 29. Heinäkuu 2020.] https://www.electricaltechnology.org/2015/09/what-is-industrial-automation.html.

Exor. 2019. What is a Human-Machine Interface or HMI. *Exor sivusto.* [Online] 7. Helmikuu 2019. [Viitattu: 3. Elokuu 2020.] https://www.exorint.com/en/blog/2019/02/07/what-is-a-human-machine-interface-and-do-you-make-or-buy-it.

GS1. About GS1. *GS1 sivusto.* [Online] [Viitattu: 24. Heinäkuu 2020.] https://www.gs1.org/about.

GS1 AISBL. 2020. GS1 General Specification. *GS1 sivusto.* [Online] Tammikuu 2020. [Viitattu: 19. Marraskuu 2020.] https://www.gs1.org/sites/default/files/docs/barcodes/GS1_General_Specifications.pdf.

—. 2020. Sovellustunnukset. *GS1 sivusto.* [Online] 2020. [Viitattu: 19. Marraskuu 2020.] https://www.gs1.fi/fi/sovellustunnukset.

GS1. Traceability. *GS1 sivusto.* [Online] [Viitattu: 11. Elokuu 2020.] https://www.gs1.org/traceability-retail.

gs1.org.

Hibbard, Kathryn. 2019. Hierarchy of Industrial Automation Systems. *Kingstar sivusto.* [Online] 1. Heinäkuu 2019. [Viitattu: 29. Heinäkuu 2020.]

https://www.autodesk.com/products/eagle/blog/rfid-works-antenna-design/. *How RFID Works.* Autodesk, s.l. : s.n.

Inductive Automation. 2018. What is HMI? *Inductive Automation sivusto.* [Online] 10. Elokuu 2018. [Viitattu: 3. Elokuu 2020.]

—. 2018. What is SCADA? *Inductive Automation sivusto.* [Online] 12. Lokakuu 2018. [Viitattu: 3. Elokuu 2020.] https://inductiveautomation.com/resources/article/what-is-scada.

inductiveautomation.com.

Insta. Tietoa meistä: Insta. *Insta.* [Online] [Viitattu: 22. Heinäkuu 2020.] https://www.insta.fi/tietoa-meista.

Miller, Josh. 2019. RFID and the Differences in Passive, Semi-Passive, and Active Tags. *Computype sivusto.* [Online] 24. Huhtikuu 2019. [Viitattu: 23. Heinäkuu 2020.] https://www.computype.com/blog/rfid-and-the-difference-in-passive-semi-passive-and-active-tags.

QAD. Definition of Enterprise Resource Planning (ERP). *QAD.* [Online] [Viitattu: 27. Heinäkuu 2020.] https://www.qad.com/what-is-erp.

qr-koodit.fi.

Scandit Products/Solutions. 2015. Types of barcodes: Choosing the Right Barcode. *Scandit sivusto.* [Online] 26. Tammikuu 2015. [Viitattu: 23. Heinäkuu 2020.] https://www.scandit.com/blog/types-barcodes-choosing-right-barcode/.

Siemens AG. 2019. Identification Systems. *Siemens AG sivusto.* [Online] 2019. [Viitattu: 19. 11 2020.] https://assets.new.siemens.com/siemens/assets/api/uuid:7251a4c1-d532-4964-a239-ec6fa933e6ee/broschuere-optische-identifikation-en.pdf.

Smiley, Suzanne. 2019. Active rfid vs passive rfid. *Atlas RFID Store sivusto.* [Online] 10. Joulukuu 2019. [Viitattu: 23. Heinäkuu 2020.] https://www.atlasrfidstore.com/rfid-insider/active-rfid-vs-passive-rfid.

Stein, Adriana. 2020. Static vs dynamic qr code. *QR Code Generator.* [Online] 03. Tammikuu 2020. [Viitattu: 22. Heinäkuu 2020.] https://www.qr-code-generator.com/blog/static-vs-dynamic-qr-code/.

The Global Language of Business. The GS1 system of standards. *GS1 sivusto.* [Online] [Viitattu: 24. Heinäkuu 2020.] https://www.gs1.org/sites/default/files/docs/architecture/AG_Flyer_final.pdf.

Unitronics. What is Industrial Automation? *Unitronicsplc sivusto.* [Online] [Viitattu: 29. Heinäkuu 2020.] https://unitronicsplc.com/what-is-industrial-automation/.

Woodford, Chris. 2019. How data matrix codes work. *Explain that stuff sivusto.* [Online] 11. Lokakuu 2019. [Viitattu: 22. Heinäkuu 2020.] https://www.explainthatstuff.com/how-data-matrix-codes-work.html.

WorkWise. WorkWise sivusto. *What is a Manufacturing Execution System (MES)?* [Online] [Viitattu: 3. Elokuu 2020.] https://www.workwisellc.com/erp-software/what-is-mes/.