CODEN:LUTEDX/(TEIE-3100)/1-38(2021)

Industrial Electrical Engineering and Automation

Automatisering av mätstation för utsäde

Oscar Persson

Division of Industrial Electrical Engineering and Automation
Faculty of Engineering, Lund University

Rapport till Examensarbete inom Automation

Automatisering av mätstation för utsäde

LUNDS UNIVERSITET
Lunds Tekniska Högskola

LTH Ingenjörshögskolan vid Campus Helsingborg
Avdelningen för industriell elektroteknik och automation
Lunds Tekniska Högskola

Oscar Persson
Handledare på LTH: Henriette Weibull
Handledare på företag: Håkan Sträng

LTH Ingenjörshögskolan vid Campus Helsingborg
Lunds universitet
Box 882
251 08 Helsingborg

LTH School of Engineering
Lund University
Box 882
SE-251 08 Helsingborg
Sweden

Tryckt i Sverige
Lunds universitet
Lund 2021

Sammanfattning

Examensarbetet syftar till att utveckla automatiseringen av en maskin som delar ett prov av frön. Detta prov består av frön för utsäde och kommer från jordbrukare för analys. Maskinen som delar provet kallas *provdelare* och delar provet i två mängder där en mängds vikt kan specificeras. Syftet med examensarbetet har varit att förbättra denna process genom att lägga till en streckkodsläsare som skannar in provets ID och användarens användarnamn. Projektet har även lagt till integrering till överordnat mätsystem där resultatet ifrån den uppdaterade maskinen lagras enligt specifikation.

Nyckelord

TwinCAT; PLC; LIMS; CSV; ST; CAB

Abstract

The goal of this thesis is to improve a machine which divides a sample of seeds. This sample contains seeds from farmers for analysis. The machine which divides the sample is called a *sample divider* and divides one sample into two parts where one weight can be chosen. The improvement of the process encompasses the adding of a barcode scanner which scans the sample number and username. The project has also added integration to the superior measurement system where the results from the updated machine is saved according to specification.

Keywords

TwinCAT; PLC; LIMS; CSV; ST; CAB

Förord

Denna rapport beskriver arbetet och resultatet från mitt examensarbete om *provdelaren* och även de problem jag stötte på och hur dem löstes.

Tills sist vill jag tacka mina två lärare Mats Lilja och Henriette Weibull även Håkan Sträng och Janne Mujunen från Demab och Krister Danielsson1 från Beckhoff för att jag fick låna en av deras PLC:er.

Innehåll

1 Inledning

1.1 Bakgrund

Examensarbetet utfördes på företaget Demab AB. Demab är en helhetsleverantör av nyckelfärdiga lösningar för industriautomation. De befinner sig i Helsingborg och levererar projektering, programmering och leverans av projekt. De erbjuder även service och reparation av nya och gamla system.

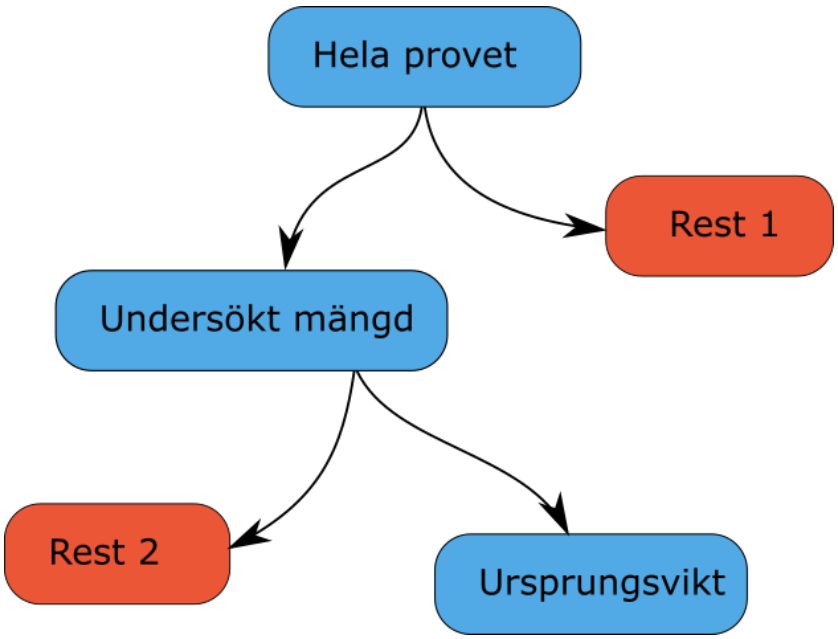
Kunden är företaget Frökontrollen som analyserar frön för utsädes som t.ex havre och vete. Lantbrukare skickar in ett prov på ca. 2.5 kg av sina frön för analys, denna bestämmer frönas kvalitet, användningsområde och även priset.

Uppkomsten att analysera frön kom då landet hade haft missväxt och det varit ont om mat. Då sågs det till att de frön med den bästa kvalitén fick bli utsäde till nästa år för att öka minska risken för missväxt. Företag så som Frökontrollen har en viktig roll för ser till att minska missväxten i landet.

Figur 1: Nuvarande system hos Frökontrollen.

1.2 Systemet idag

Maskinen som examensarbetet har modifierat är maskinen *provdelaren*, se blå maskin i figur 1. Den delar upp en mängd frön i två delar, där vikten på en av dessa delar är specificerad av användaren. Då mängderna i de två behållarna måste ge en representativ bild av hela provet är det viktigt att provet blir representativt fördelat. Eftersom ett prov inte är uniformt och även innehåller andra frön och partiklar är detta ännu viktigare. Delningen görs representativ genom att låta provet delas när det faller genom *provdelaren*, eftersom det då delas oberoende hur innehållet är fördelat. De två behållarna i botten av *provdelaren* får då liknande fördelning av andra frön och partiklar då provet är delat. Efter det att provet delats två gånger genom *provdelaren* enligt figur 2, finns två mängder som kan användas för analys. Dessa mängder har namnen ”Undersökt mängd”, ”Ursprungsvikt” och är definierade med dessa namn av Frökontrollen.



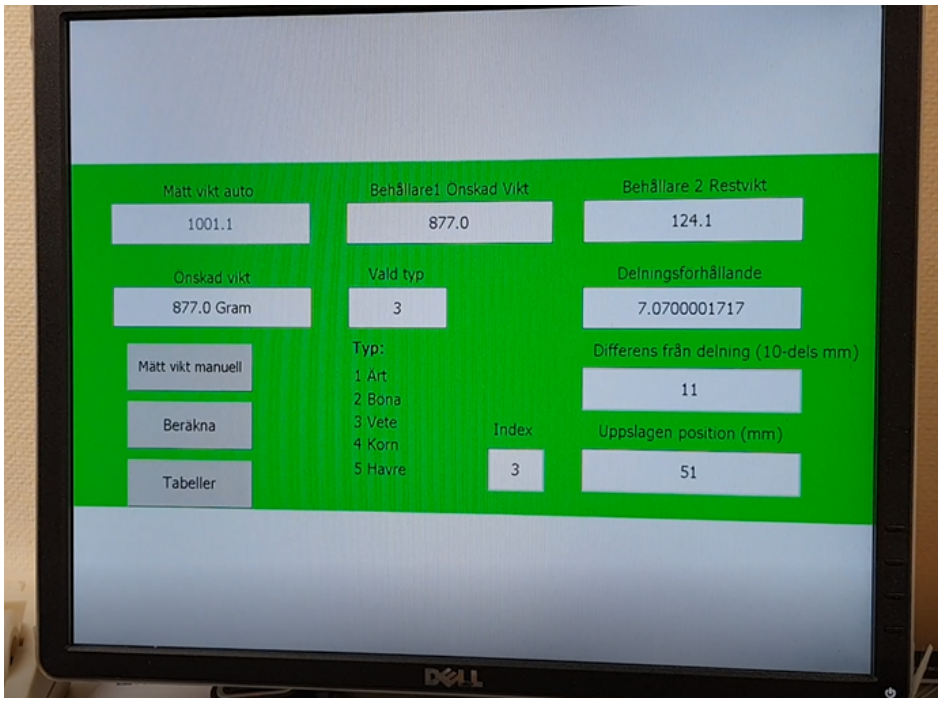
Figur 2: Delningsprocessen för ett prov.

Vikterna ”Undersökt mängd” och ”Ursprungsvikt” som fås ut från *provdelaren* är viktiga att spara för efterföljande analyser och för att samla provdata tillsammans i mätsystemet LIMS. Provdata i tabell 1 ges av en sekvens och sparades inte av PLC programmet utan skrevs ner på papper för att sedan

manuellt skrivas in i mätsystemet LIMS. Denna manuella hantering riskerade att orsaka fel som i sin tur kan påverka efterföljande analys.

Tabell 1: Exempelresultat som ska sparas i LIMS.

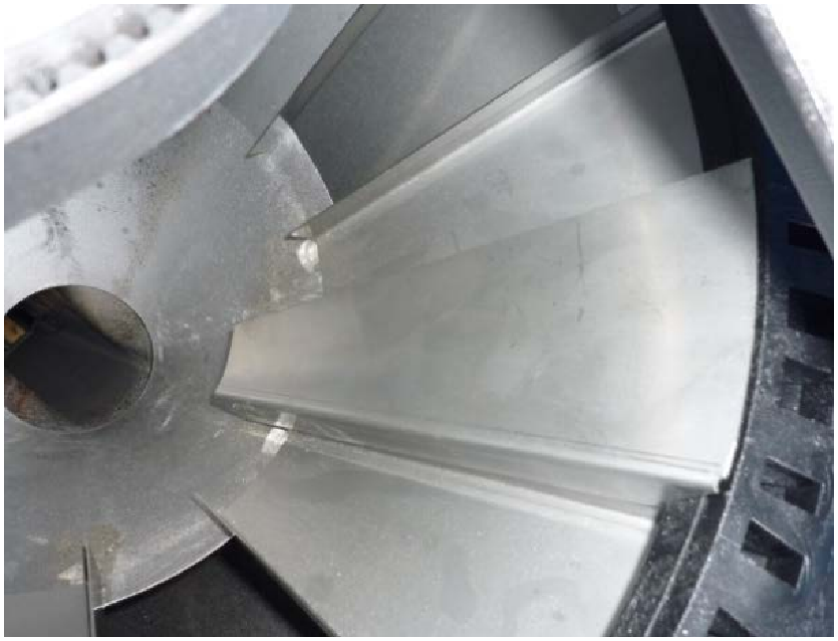
MSO	Användarnamn
2020-10-15 07:36	Analysdatum och klockslag
Korn	Typ av gröda
Prov-ID #####	Provets nummer
1023 g	Undersökt mängd
870 g	Ursprungsvikt



Figur 3: Tidigare användargränssnittet.

I figur 3 visas användargränssnittet för *provdelaren* som syns i figur 1 som används av Frökontrollen. För att dela ett prov mäts först hela provets vikt med hjälp av en våg. Vikten från vågen förs därefter över till användargränssnittet genom ett knapptryck på vågen. Då matas vikten in i programmet och vikten på provet syns i fältet ”Mätt vikt auto”. Efter det skrivs den önskade

vikten efter provdelningen i behållare ett in i fältet ”Önskad vikt”. Denna vikt används för att ställa in *provdelaren* till en inställning som ger en vikt så nära den önskade vikten som möjligt.



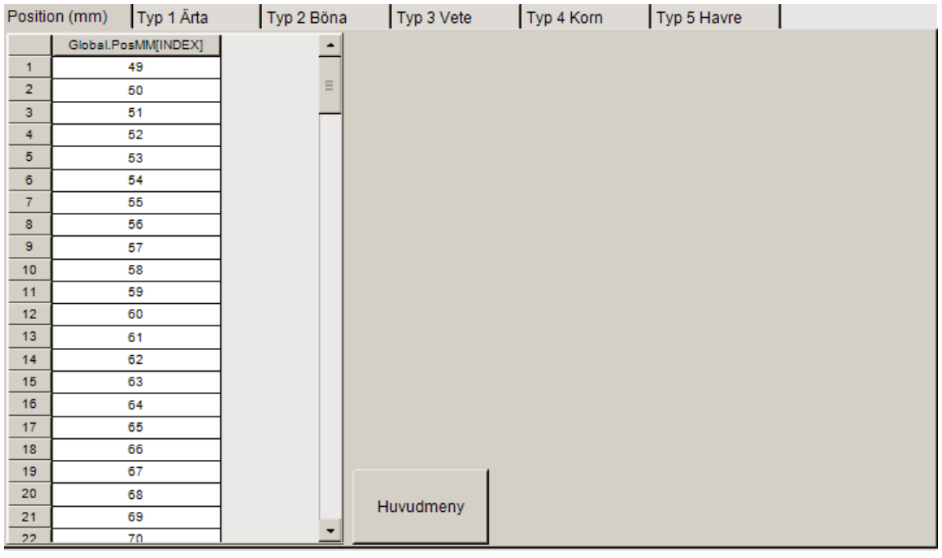
Figur 4: Inuti provdelaren.

Därefter väljs grödan genom att ändra siffran i fältet ”Vald typ” till den kod som provets gröda har. Koderna för de fem olika grödorna syns i listan under fältet ”Vald typ”. Provets gröda väljs eftersom olika grödor faller olika igenom *provdelaren* och på så sätt måste justeras för sig. För att ställa in *provdelaren* klickas knappen ”Beräkna”, nu används den valda grödans tabell för att beräkna den närmsta möjliga delningen. Delningen sker med en två cirkulära skivor, se figur 4, en av skivorna kan rotera och på så sätt ändra delningsförhållandet mellan det som faller på ena skivan och det som faller igenom hållen i skivan. Eftersom delningsförhållandena i kransen är mekaniskt och alltid ger ungefär samma delning mellan de två behållarna, är det ett begränsat antal möjligheter som provet kan delas med. I detta fallet är *provdelaren* begränsad till 35 olika delningsförhållande på kransen och dessa förhållandena är inskrivna i PLC programmet. Delningsförhållandet för det prov som syns i figur 3 är 7.07 och fås genom att beräkna

$$\frac{\text{behållare 1}}{\text{behållare 2}}. \tag{1}$$

De delningsförhållanden som används vid justering av *provdelaren* syns om knappen *"Tabeller"* klickas, se figur 5. Dessa värden är beräknade genom att först ställa in *provdelaren* på en justering och sedan hälla igenom ett prov. Därefter kan förhållandet mellan behållare ett och två räknas ut och skrivas in i tabellen för grödan. Dessa tabeller syns som flikar i figur 5 och innehåller då alla delningsförhållanden för flikens gröda. Det som visas i fältet "Delningsförhållande" är delningsförhållandet, se ekvation 1, mellan behållare ett och två. Denna siffra jämförs med de tabeller som finns för att bestämma den närmaste inställningen för delningen. Den närmsta inställningen tas fram genom att PLC:n går igenom hela tabellen för den specifika grödan och väljer då det tabellvärde som är närmst.

Den skillnad i tiondels millimeter där den inställningen som finns och den mest optimala vissas i fältet "Differens från delning". Det index i tabellen som valdes för delningen står i fältet "Index" och positionen i mm på kransen syns i fältet "Uppslagen position". "Uppslagen position" är den position som är hämtad från tabellen som syns i figur 5.



Figur 5: Tabeller med grödornas olika delningsförhållande.

1.3 Syfte

Syftet med examensarbetet har varit att minska arbetstiden för en mätsekvens till *provdelaren* samt att öka kvalitén genom att minska risken för felaktig hantering. Detta har gjorts genom att vidareutveckla användargränssnittet och automatiskt spara värdena som kan ses i tabell 1 i mätsystemet LIMS. Sekvensen ska också bli enklare och tidseffektivare genom att skanna in ID:et på provet och användarnamnet med en streckkodsläsare.

1.4 Målformulering

Examensarbetet ska ge en bättre version av mätsystemet som kommet att underlätta mätning, minska risken för fel och förkorta tidsåtgången. Detta genom att utveckla ett nytt användargränssnitt för *provdelaren*, så att mätresultaten lätt ska kunna sparas i en CSV fil. Provdata som ses i tabell 1 är den data som ska sparas i CSV. Det ska kopplas in en streckkodsläsare för att kunna skanna in provets ID och användarens användarnamn med hjälp av en streckkod.

1.5 Problemformulering

Innan examensarbetet hade påbörjats fanns det några frågor kring hur vissa saker fungerande. De frågor som var specifika och antogs vara besvarade innan projektets slut finns nedan.

1. Hur implementeras en automatisk process som sparar mätvärdena till LIMS-systemet?
2. Hur fås mätresultaten från nuvarande process?
3. Hur för en streckkodsläsare över ett värde till en PLC?
4. Hur fungerar grafiska gränssnitt tillsammans med en PLC?

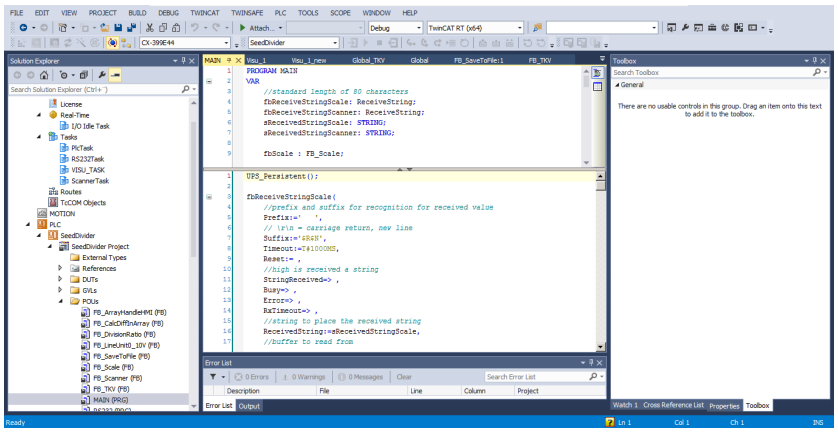
1.6 Avgränsningar

Mätresultaten kommer att hämtas från det befintliga systemet och sedan sparas i en CSV fil som gränssnitt mot LIMS. Däremot kommer inte mätvärdena beräknas eftersom beräkningsdelen redan är fungerande i det befintliga systemet.

2 Teknisk bakgrund

2.1 TwinCAT

För att slutföra projektet krävdes kunskap i utvecklingsmiljön TwinCAT, se figur 6, som används för att programmera Beckhoff PLC:er. I TwinCAT finns det möjlighet att programmera i språken ST, IL, FBD, LD, CFC och SFC [1]. I programmet kopplas PLC:n upp genom att först ha anslutits med en ethernet-kabel till det lokala nätverket och sedan har PLC:n sökts upp i TwinCAT. När PLC:n är identifierad via IP-adress klicka på knappen ”Logga in” för att koppla upp sig till PLC:n. Nu syns det om PLC:n körs för nuvarande eller är stoppad, dessa tillstånd går att skifta emellan närsomhelst då TwinCAT är ”inloggad” till PLC:n.

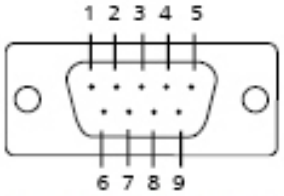


Figur 6: Hur programmeringsmiljön TwinCAT ser ut.

På vänstersidan i figur 6 ses projektet i sin helhet, här syns alla filer som finns i det öppnade projektet. I mitten visas filen som är öppen och dess innehåll, i detta fallet ”MAIN” filen där huvudprogrammet finns i detta projekt. I övre delen av filen syns deklarationen av variabler och i nedre delen logiken som exekveras varje programcykel. Längst ner är ett fönster som visar felmeddelanden för programmet, då det byggs och under exekvering kan felmeddelande dyka upp här.

2.2 Skanner & RS-protokollen

Ett av kraven i projektet var att kunden ville ha en skanner [2] för att förenkla för användaren och öka kvalitén på mätningarna. Denna skanner anslöts med en DB9-kontakt, se längst upp i figur 7, via RS422 protokoll men då skannern använde RS232-protokollet användes en adapter för att konvertera signalen mellan de två protokollen. Protokollen använder båda en DB9-kontakt men de använder olika pinnar på olika sätt, som kan ses i figur 7, och går därför inte att använda tillsammans.



PIN	RS-232	RS-485 (4W)	RS-485 (2W)	RS-422
1	DCD	TxD-(A)	---	TxD-(A)
2	RXD	TxD+(B)	---	TxD+(B)
3	TXD	RxD+(B)	Data+(B)	RxD+(B)
4	DTR	RxD-(A)	Data-(A)	RxD-(A)
5	GND	GND	GND	GND
6	DSR	--	---	---
7	RTS	--	---	---
8	CTS	--	---	---
9	--	--	---	---

Figur 7: Olika RS protokoll [3].

I figur 7 ses skillnaden mellan de två protokollen RS232 och RS422. RS232 har en linje för att skicka/ta emot och använder $\pm$3V som ett och noll, medan RS422 har två linjer och skickar med endast $\pm$0.4V. Detta är möjligt då RS422 har differentiella signaler, vilket gör att det är möjligt att använda en lägre spänning jämfört med RS232. Differentiella signaler gör även att RS422 kan användas på längre sträckor eftersom störningar har mindre påverkan på informationen som skickas. Differentiella signaler innebär att två ledare används för att skicka/ta emot data. Informationen som är skickad är inte relativt jord utan är skillnaden mellan de två ledarna. Den störningen som bildas på vägen är i teorin densamma och på så sätt elimineras den då skillnaden mellan ledarna används.

RS232 och RS422 använder en DB9-kontakt, en sådan kontakt har 9 pinnar och används för seriell kommunikation. Däremot använder inte RS422 alla 9 pinnar som finns på en DB9-kontakt och kan på så sätt använda en mindre kontakt. RS232 infördes 1960 och har idag blivit mestadels ersatt med USB. På de flesta enheter finns det inte längre en DB9-kontakt eftersom färre enheter använder den sortens kontakt. De flesta enheter använder idag USB-A men är på väg att bli ersatt med USB-C som är både mindre och kan kopplas på båda hållen.

2.3 LIMS

Det system som företaget Frökontrollen använder för att spara sina mätvärden heter LIMS och står för Laboratory Information Management System och sparar mätvärden från flera av Frökontrolls mätstationer. LIMS är en mjukvara som används för att ta hand om data och kan även läsa av filer och spara informationen i systemet. Frökontrollen måste hålla koll på all information angående ett prov för att sedan kunna skicka resultatet från alla analyser till lantbrukaren.

3 Metod

Det första som gjordes i projektet var att lära sig utvecklingsmiljön TwinCAT. Detta gjordes genom ett antal videor som förklarade grunderna om programmet. Videokursen använde en testrigg som fanns för att kunna testa programmen som skrevs under kursen. Då kursen var avklarad lästes programmet för det nuvarande systemet för att förstå hur det fungerade. Programmet kommenterades med förklaring på vad allt gjorde och de saker som inte förstods söktes upp. De delar av programmet som inte behövdes togs bort och andra delar förenklades för att enkelt kunna lägga till nya funktioner senare.

Figur 8: Skannern som användes [2].

Därefter beställdes en skanner, se figur 8, RS232 till RS485 omvandlare och en 10 pin ethernet-kabel. Den PLC som användes lånades från Beckhoff och var likvärdig PLC:n som fanns i nuvarande systemet hos Frökontrollen.

När PLC:n hade anlänt kopplades den upp mot datorn via en ethernetkabel till det lokala nätverket. En skärm kopplades även upp till PLC:n med en DVI kabel. Efter lite problem med att föra över programmet till PLC:n kördes till sist programmet, ny kod kunde nu läggas till.

När resten av delarna hade anlänt var det dags att koppla upp skannern till

PLC:n. Först testades omvandlaren med en usb till RS232-omvandlare och när kommunikationen mellan dessa fungerade kopplades skannern och PLC:n in till omvandlaren.

När skannern kopplats in och fungerande behövdes mer information från slutkunden om hur programmet skulle användas. Då anordnades ett möte med Frökontrollen för att fastställa tekniska krav och hur gränssnittet mot användaren skulle se ut.

Först planerades det vad som skulle finnas på skärmen till HMI:n och sedan implementerades de olika delarna. Först gjordes det möjligt att väga in proven så som det förut var implementerat och därefter kunna skanna in provnummer och användarnamn. När de delarna var på plats implementerades sparfunktionen så att mätsystemet LIMS kunde läsa av mätresultaten och spara dessa.

3.1 Källkritik

Källan [3] har används för att leta rätt på information om hur RS422 protokollet skulle vara inkopplat. Eftersom inkoppling efter denna källa fungerade så stämmer källan.

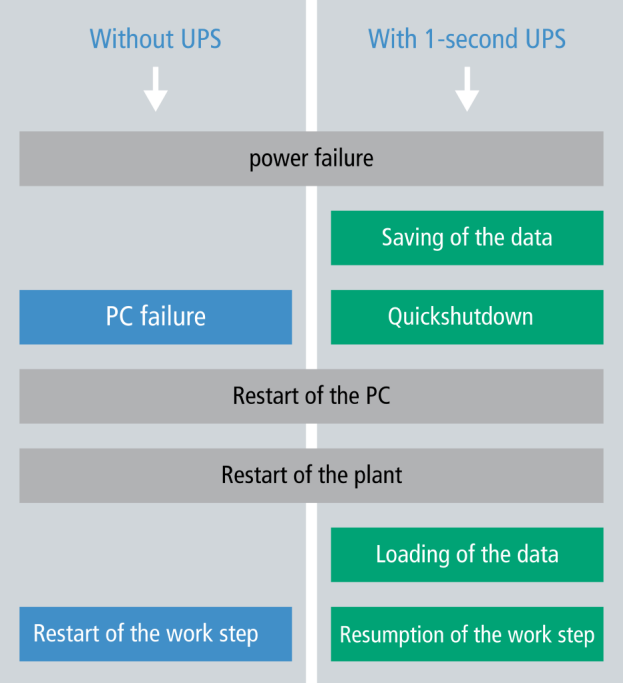
Produkten från [2] är den produkt som används och eftersom det är från en säljare är det väldigt säkert att informationen är korrekt. för bild

Från [4] användes en bild, denna bild är tagen från tillverkaren hemsida och är hur systemet fungerar.

Från boken [1] användes information kring de programmeringsspråk som användes. Eftersom programmet är fungerande stämmer även det som beskrevs i boken.

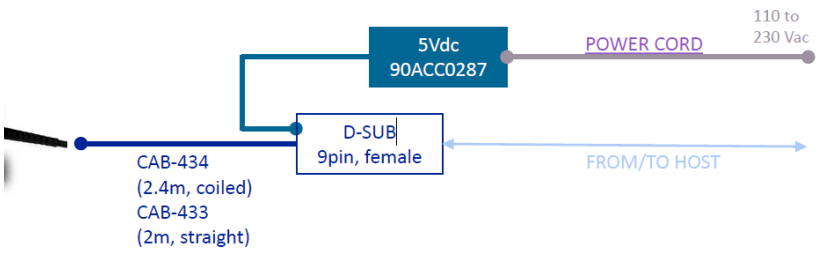
4 Analys

Under examensarbetets gång uppstod en rad problem som behövde lösas. Det första var när PLC:n skulle programmeras med en kopia av kundens program. PLC:n kopplades upp med en ethernetkabel till nätverket och anslutningen till TwinCAT fungerade. Men när PLC:n programmerades uppstod ett felmeddelande och programmeringen misslyckades. Försök att identifiera felmeddelandet gjordes men inget vidare svar hittades. När programmet genomgicks på nytt upptäcktes ett beroende till Frökontrollens system, detta beroende var kopplat till den hårdvara som används av Frökontrollen. Då en annan typ av PLC användes fick beroendet ändras till det som stöddes av den likvärdiga hårdvaran som skulle programmeras. Därefter gick det att programmera PLC:n och starta den. Detta beroende behöver ändras tillbaka när programmet ska användas till Frökontrollens system. Då problemet var löst framgick det varför Beckhoff hade öppnat en fil i TwinCAT då felmeddelandet dök upp. Filen som hade öppnats var där felet fanns, men detta upptäcktes inte förrän efter problemet var löst. Beroendet var till UPS:en, vilket visas i figur 9 och används för att spara data då strömmen bryts. När PLC:n sedan får ström igen återupptas exekvering och PLC:n återställs till tidigare tillstånd.



Figur 9: Hur UPS fungerar [4]

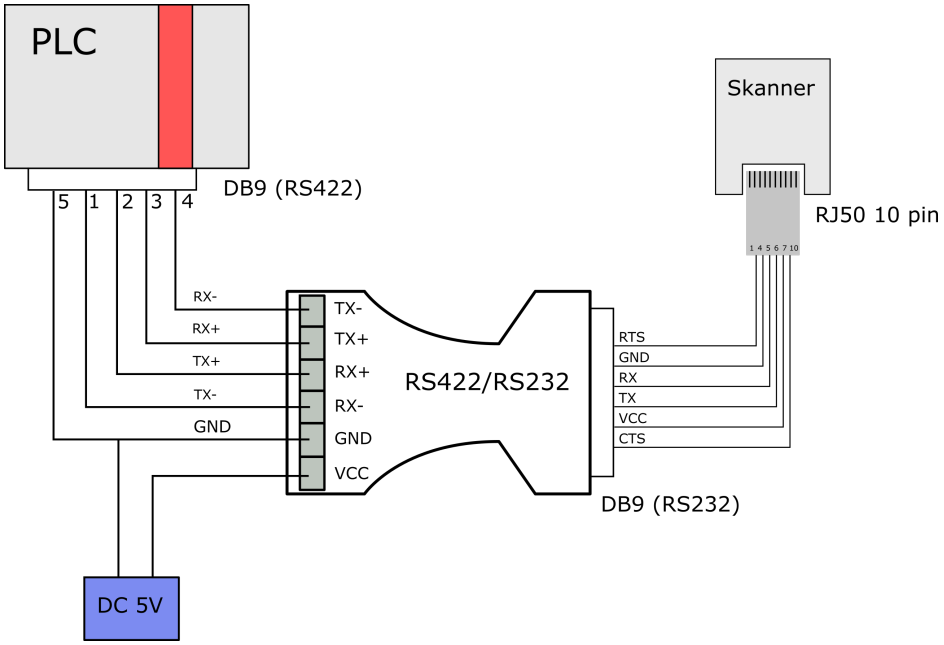
Nästa utmaning var att koppla in skannern. När den anslöts till nätaggregatet startade den inte, nätaggregatet byttes ut mot flera andra nätaggregatet men med samma resultat. Därefter kontaktades leverantören och efter detaljerat datablad skickats upptäcktes att skannern krävde en CAB-434/433 kabel, se figur 10. Kabeln som var levererad var dock en CAB-327 kabel och är spegelvänd mot CAB-434/433 och då spänningen till skannern vändes startade skannern. Skillnaden mellan CAB-327 och CAB-434/433 konstaterades med en multimeter genom att kartlägga pinnarna på den levererade CAB-327 kabeln.



Figur 10: Information från leverantören.

När skannern indikerade att den var redo, var det dags att koppla upp den till datorn för att säkerställa att kommunikationen fungerade. Problemet som uppstod var att adaptern som omvandlade mellan USB och DB9 inte kunde identifieras som en COM-port. Detta berodde på att det var bristfälliga drivrutiner installerade till enheten. En drivrutin [5] laddades ner och installerades till enheten, vilket löste problemet och adaptern var nu identifierad som en COM-port.

Nu när det var säkert att kommunikationen fungerade fram till datorn skulle omvandlaren kopplas mellan PLC:n och skannern. Först undersöktes det vilka pinnar en RS422-kontakt hade för kommunikation och därefter kontakterna in enligt hänvisningarna. Efter inkoppling kunde skannern testas med en streckkod, men då det gjordes blinkade inte lampan på PLC:n som skulle blinka då den tar emot data. Kablarna till omvandlaren testades med ett antal olika konfigurationer till, enligt RS485-protokollet, även RS485-protokollet spegelvänd och några till enligt andra framsökta bilder. Det som löste problemet var en bild som fanns i databladet till PLC-modulen för RS422/485 kontakten där det stod hur kontakten var kopplad. Denna bild var liknande de flesta förutom att både in- och utpinnarna till kabeln var märkta. Då var det klart att den allra första kopplingen var rätt förutom att TX-kontakten ska gå till RX och inte TX till TX. I figur 11 nedan kan kopplingen in och ut ur omvandlaren ses.



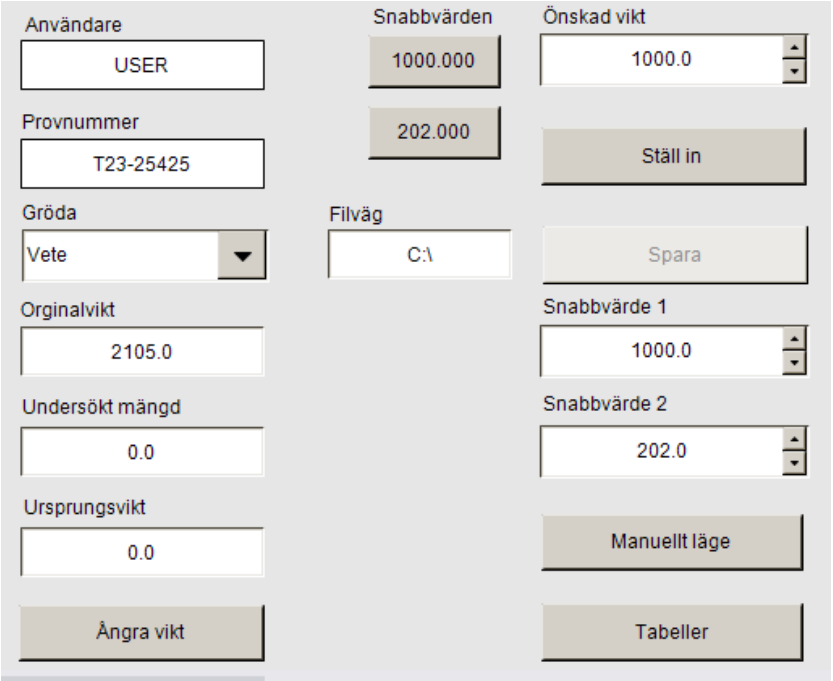
Figur 11: Diagram över koppling ut och in till omvandlare

Programmet valdes att skrivas i strukturerad text eftersom Frökontrollens projekt var skrivet i det språket.

5 Resultat

Slutleverans till kund blev ett program som kan ta emot värden från en skanner och en våg och sedan spara värdena i en fil för vidare sparande av mätresultaten i mätsystemet LIMS.

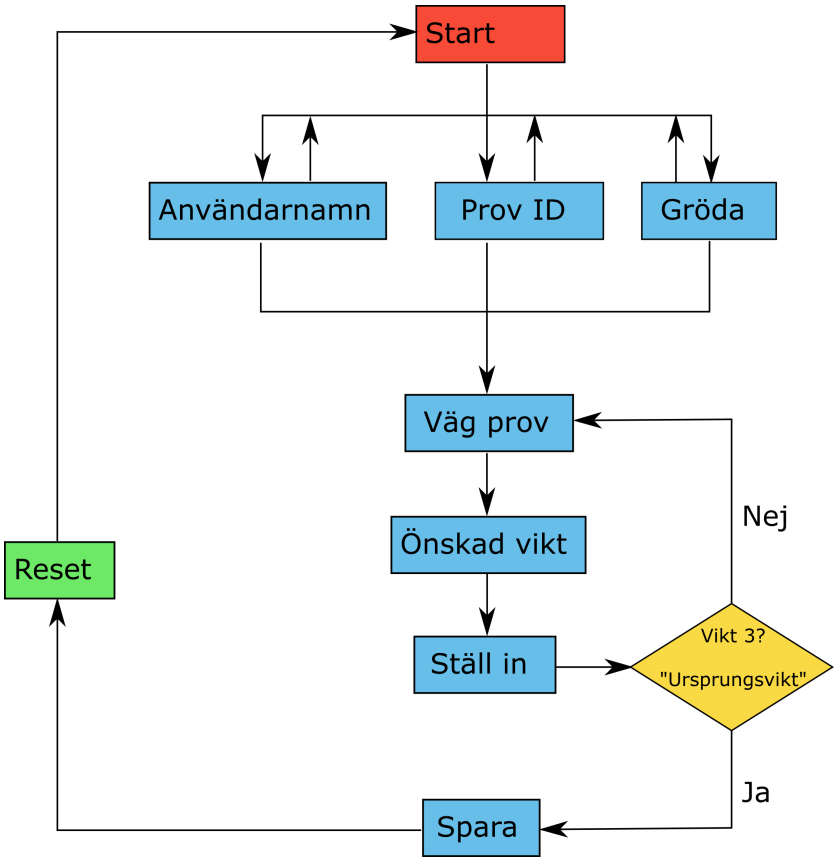
Ett antal extra funktioner som att skanna in användarnamn, använda snabbvärden för önskad vikt och filväg som går att ändra direkt i HMI:n lades till under projektet för att underlätta arbetsprocessen.



Figur 12: Nya användargränssnittet.

När programmet i figur 12 med flödet enligt figur 13 startar är alla fälten tömda och användarnamnet skannas då in med en streckkod med personalens användarnamn. Därefter skannas även ID-numret som befinner sig på en provlapp in med streckkodsläsaren, därefter väljs även provets grödan i den expanderbara listan. Nu matas vikten på provet in med ett tryck på vågen och fältet ”Originalvikt” fylls i med det värdet som stod på vågen. Nu ska provet delas första gången och då fylls fältet ”Önskad vikt” i med den önskade vikten på en av behållarna under *provdelaren*. Därefter trycks knappen ”Ställ in” för att ställa in *provdelaren* till närmsta korrekta delning. När provet är

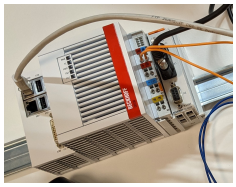
delat sätts provet på vågen och knappen på vågen trycks igen, den nya vikten kommer nu skrivas in i fältet ”Undersökt mängd”. Därefter delas provet igen på samma sätt, vikten kommer nu skrivas in under ”Ursprungsvikt” istället. När alla värden nu är ifyllda går det att trycka på knappen ”Spara”.



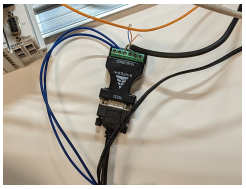
Figur 13: Användargränssnittsflödet

På skärmen finns det några knappar och fält till som inte behövs i den vanliga arbetsprocessen. Knapparna ”Snabbvärden” används för att enkelt fylla i värden i fältet ”Önskad vikt”, då det vanligtvis är samma två värden som matas in i fältet varje gång provet delas. Knappen ”Ställ in” är gjord så att den inte går att trycka på om ”Önskad vikt” är noll. Knappen ”Manuellt värde” går till det tidigare användargränssnittet se figur 3 då det ger extra information som kan vara bra om något går fel.

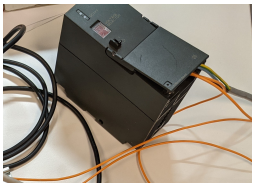
(a) Streckkodsläsaren.



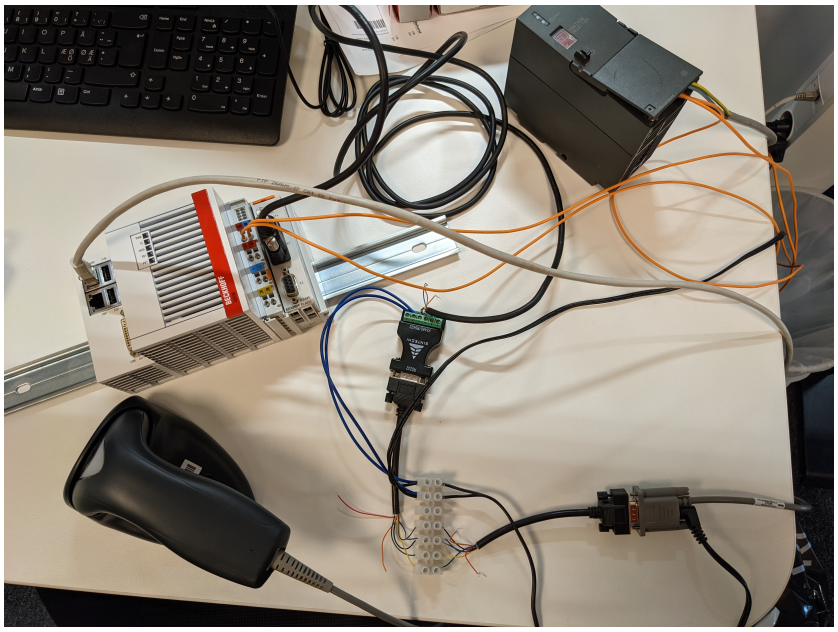
(b) PLC:n som användes.



(c) Omvandlare till skannern.



(d) Nätaggregat till PLC:n.



(e) Sammankopplingen.

Figur 14: Verkliga kopplingen.

I figur 14 visas de individuella delarna till systemet och även hur systemet är kopplat i verkligheten.

Sparfunktionen som implementerades för att spara filer kan ses i Appendix A. Den är skriven i strukturerad text och fungerar på följande vis:

1. En fil skapas med provets id.
2. Texten som ska skrivas i filen förbereds.
3. Texten skrivs in i filen
4. Filen stängs så och sparas med provets data, se tabell 1.

När hela sekvensen är genomgången finns det en fil med provets id som namn

och dess mätdata i filen formaterat som en CSV-fil. Om något går fel under sparandet av filen kommer filen att stängas ner och allt avbrytas. Dessutom kommer värdena i användargränssnittet vara kvar, alltså inte bli rensat som det normalt görs.

6 Slutsats

1. Hur implementeras en automatisk process som sparar mätvärdena till LIMS?

   Då LIMS-systemet endast läser filer i en specifik mapp var det inte nödvändigt att skicka data till LIMS-systemet. Det som gjordes för att LIMS-systemet skulle spara data i servern, var att skapa en CSV fil med de data som syns i tabell 1 och sedan spara denna fil i en mapp för inläsning av LIMS-systemet.

2. Hur fås mätresultaten från nuvarande process?

   Värdena som ska spara i CSV filen som gränssnitt mot LIMS fås från programmet genom ett antal olika variabler. Dessa variabler matas in i ett funktionsblock som sedan sparar värdena i filen.

3. Hur för en streckkodsläsare över värde till en PLC?

   Skannern som användes kommunicerar med en DB9 kontakt med protokollet RS232, men eftersom PLC:n inte har en sådan kontakt används en omvandlaren mellan PLC:n och skannern. Omvandlaren konverterar mellan RS232 och RS422 som PLC:n stödjer med ett IO kort. Efter det är behövs ett funktionsblock sättas up i programmet för att koppla variabler i med hårdvaran.

4. Hur fungerar grafiska gränssnitt tillsammans med en PLC?

   Grafiskt gränssnitt i Beckhoff definieras helt grafiskt i deras programmeringsmiljö TwinCAT. Det som har gjorts är att dra in olika komponenter från en lista och sedan tilldela olika värden till komponenternas olika variabler.

6.1 Reflektion över etiska aspekter

I mätdata från provdelaren sparas användarnamnet på den person som gjort arbetet för att på så sätt veta vem som gjort vad. Detta ska även kunna användas då något blivit fel för att lätt kunna fråga den som utfört arbetet. Denna information som sparas i LIMS kommer endast att finnas i LIMS och inom företaget.

6.2 Framtida utvecklingsmöjligheter

En framtida utvecklingsmöjlighet är att bättre integrera vågen i *provdelaren* så att den delar provet mer noggrant. Detta görs genom att ha en våg under en av behållarna kopplad direkt till PLC:n. Vågens värde skulle registreras direkt i programmet och bestämma vilken bägare som ska fyllas. Den slutliga vikten på ett av delproven skulle då kunna synas på skärmen direkt utan att manuellt behöva väga behållaren. Eftersom systemet då reglerar mot en våg under behållaren skulle inte provet behöva vägas i början.

7 Terminologi

- HMI

  Står för Human-Machine Interface och är gränssnittet där slutanvändaren använder applikationen, i detta fallet syns applikationen på en datorskärm inkopplad till PLC:n.

- RS protokoll

  RS i de olika protokollen så som 422 och 232 står för Recommended Standard och är för seriell kommunikation.

- LIMS

  LIMS står för Laboratory Information Management System och är mjukvara för att spara och följa data tillsammans.

- DB9

  En DB9-kontakt används för seriell kommunikation och har 9 pinnar men finns även i andra storlekar med fler eller färre antal pinnar. D-delen står för att kontakten är D-formad, B-delen står för kontaktens storlek och 9 står för antal pinnar kontakten har. Enligt namnkonventionen för D-sub kontakter heter kontakten egentligen DE9 men eftersom de flesta använder DB9 har namnet kvarstått.

- CSV

  CSV står för ”comma separated values” och är en enkel struktur för att spara en tabell med värden till en fil. Överst i filen skrivs namnen på kolumnerna separerade med ett komma. På raden därefter sparas första tabellens rads värden också separerade med komma. Raderna därefter är precis som förra, alltså radens värden separerade med komma.

8 Källförteckning

Referenser

[1] K. H. J. Tiegelkamp, *IEC 61131-3: Programming Industrial Automation Systems*. Springer, Berlin, Heidelberg, 2010.

[2] "Datalogic Touch 65 Pro Multi-Interface Black." (2021), URL: https://www.dustin.se/product/5010745671/touch-65-pro-multi-interface-black (hämtad 2021-05-04).

[3] "DB9 RS23, RS422, RS485 Pinouts." (2011), URL: http://8051-pic-avr.blogspot.com/2011/01/db9-rs23-rs422-rs485-pinouts.html (hämtad 2021-05-05).

[4] "1-second UPS (persistent data)." (2021), URL: https://infosys.beckhoff.com/english.php?content=../content/1033/cx51x0_hw/2574537867.html (hämtad 2021-06-05).

[5] "IO Cable PL 2303 Drivers Generic Windows PL2303 Prolific." (2021), URL: https://github.com/hitfzyangdianshi/IO-Cable_PL-2303_Drivers-Generic_Windows_PL2303_Prolific (hämtad 2021-05-03).

Figurer

Tabeller

Appendix A

```
CASE step OF
  0:
    IF RisingEdge.Q THEN
      hFile := 0;
      bBusy := TRUE;
      bError := FALSE;
      nErrId := 0;
      step := 1;
    END_IF

  1:
    fbFileOpen(bExecute := FALSE);
    fbFileOpen(
      sNetId := sNetId,
      sPathName := sPath,
      nMode := FOPEN_MODEWRITE OR FOPEN_MODETEXT,
      ePath := PATH_GENERIC,
      bExecute := TRUE,
      tTimeout := tTimeOut
    );
    step := 2;

  2:
    fbFileOpen(bExecute := FALSE);
    IF NOT fbFileOpen.bBusy THEN
      IF fbFileOpen.bError THEN
        nErrId := fbFileOpen.nErrId;
        bError := TRUE;
        step := 50;
      ELSE
        step := 3;
        hFile := fbFileOpen.hFile;
      END_IF
    END_IF
```

```
3:
  fbFilePuts(bExecute := FALSE);

  csvString := '';
  fbCSVWriter.eCmd := eEnumCmd_First;

  FOR nColumn := 0 TO 5 BY 1 DO
    fbCSVWriter(
      pBuffer := ADR(csvString),
      cbBuffer := SIZEOF(csvString) - 1,
      putValue := STRING_TO_CSVFIELD(columnNames[nColumn], FALSE),
      bCRLF := nColumn = 5 // if at end last column
    );

    IF fbCSVWriter.bOk THEN
      fbCSVWriter.eCmd := eEnumCmd_Next;
    //else error
    END_IF
  END_FOR
  //replace $R$L with $L to avoid double $R
  IF RIGHT( csvString, 2) = '$R$L' THEN
    csvString := REPLACE(csvString, '$L', 2, LEN(csvString) - 1 );
  END_IF


  fbFilePuts(
    sNetId := sNetId,
    hFile := hFile,
    sLine := csvString,
    bExecute := TRUE,
    tTimeout := tTimeOut
  );
  fbFilePuts.bExecute := FALSE;

  step := 4;

4:
```

```
  fbFilePuts(bExecute := FALSE);
  IF NOT fbFilePuts.bBusy THEN
    IF fbFilePuts.bError THEN
      nErrId := fbFilePuts.nErrId;
      bError := TRUE;
      step := 50;
    ELSE
      step := 5;
    END_IF
  END_IF

5:
  csvString := '';
  fbCSVWriter.eCmd := eEnumCmd_First;
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(sUser, FALSE),
    bCRLF := FALSE // if at end last column
  );
  fbCSVWriter.eCmd := eEnumCmd_Next;
  ntGetTime.START := FALSE;
  ntGetTime(
    NETID := sNetId,
    START := TRUE,
    TMOUT := T#2S
  );
  step := 6;
6:
  ntGetTime(START := FALSE);
  IF NOT ntGetTime.BUSY THEN
    IF ntGetTime.ERR THEN
      nErrId := ntGetTime.ERRID;
      bError := TRUE;
      step := 50;
    ELSE
      step := 7;
```

```
    END_IF
  END_IF
7:
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(SYSTEMTIME_TO_STRING(ntGetTime.TIMESTR), FA
  );
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(sCrop, FALSE)
  );
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(sSampleNumber, FALSE)
  );
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(sUndersoktMangd, FALSE)
  );
  fbCSVWriter(
    pBuffer := ADR(csvString),
    cbBuffer := SIZEOF(csvString) - 1,
    putValue := STRING_TO_CSVFIELD(sUrsprungsVikt, FALSE),
    bCRLF := TRUE // if at end last column
  );
  //replace $R$L with $L to avoid double $R
  IF RIGHT(csvString, 2) = '$R$L' THEN
    csvString := REPLACE(csvString, '$L', 2, LEN(csvString) - 1 );
  END_IF

  fbFilePuts(bExecute := false);
  fbFilePuts(
    sNetId := sNetId,
```

```
    hFile := hFile,
    sLine := csvString,
    bExecute := TRUE,
    tTimeout := tTimeOut
  );
  step := 8;
8:
  fbFilePuts(bExecute := FALSE);
  IF NOT fbFilePuts.bBusy THEN
    IF fbFilePuts.bError THEN
      nErrId := fbFilePuts.nErrId;
      bError := TRUE;
      step := 50;
    ELSE
      step := 9;
    END_IF
  END_IF
9:

  fbFileClose(bExecute := FALSE);
  fbFileClose(
    sNetId := sNetId,
    hFile := hFile,
    bExecute := TRUE,
    tTimeout := tTimeOut
  );
  step := 10;
10:

  fbFileClose(bExecute := FALSE);
  IF NOT fbFileClose.bBusy THEN
    IF fbFileClose.bError THEN
      nErrId := fbFileClose.nErrId;
      bError := TRUE;
    END_IF
    hFile := 0;
    step := 50;
```

```
      //file saved
    END_IF
  50:
    IF hFile <> 0 THEN
      //close file
      step := 10;
    ELSE
      //exit
      step := 0;
      bBusy := FALSE;
    END_IF
END_CASE
```