CHALMERS
UNIVERSITY OF TECHNOLOGY

Sjukhus	Distrikt	Förvaltningsobjekt	Byggnad	Våningsplan	System	Objekt	Objektnamn PLC	Objekttyp	Fabrikat PLC
Sahlgrenska	G	5000	A	1	X1	Y1	G5000_A1_X1_Y1	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	A	1	X2	Y2	G5000_A1_X2_Y2	Cirkulationspump	Beckhoff
Sahlgrenska	G	5000	A	1	X3	Y3	G5000_A1_X3_Y3	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	A	1	X4	Y4	G5000_A1_X4_Y4	Fläkt	Beckhoff
Sahlgrenska	G	5000	A	1	X5	Y5	G5000_A1_X5_Y5	Larm	Beckhoff
Sahlgrenska	G	5000	A	2	X6	Y6	G5000_A2_X6_Y6	Nattkyla	Beckhoff
Sahlgrenska	G	5000	A	2	X7	Y7	G5000_A2_X7_Y7	Spjäll	Beckhoff
Sahlgrenska	G	5000	A	2	X8	Y8	G5000_A2_X8_Y8	Spjäll	Beckhoff
Sahlgrenska	G	5000	A	2	X9	Y9	G5000_A2_X9_Y9	Ventilställdon	Beckhoff
Sahlgrenska	G	5000	A	3	X10	Y10	G5000_A3_X10_Y10	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	A	3	X11	Y11	G5000_A3_X11_Y11	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	A	4	X12	Y12	G5000_A4_X12_Y12	Givare	Beckhoff
Sahlgrenska	G	5000	A	5	X13	Y13	G5000_A5_X13_Y13	Fläkt	Beckhoff
Sahlgrenska	G	5000	A	5	X14	Y14	G5000_A5_X14_Y14	Fläkt	Beckhoff
Sahlgrenska	G	5000	A	5	X15	Y15	G5000_A5_X15_Y15	Larm	Beckhoff
Sahlgrenska	G	5000	A	5	X16	Y16	G5000_A5_X16_Y16	Cirkulationspump	Beckhoff
Sahlgrenska	G	5000	B	1	X17	Y17	G5000_B1_X17_Y17	Nattkyla	Beckhoff
Sahlgrenska	G	5000	B	1	X18	Y18	G5000_B1_X18_Y18	Ventilställdon	Beckhoff
Sahlgrenska	G	5000	B	1	X19	Y19	G5000_B1_X19_Y19	Spjäll	Beckhoff
Sahlgrenska	G	5000	B	1	X20	Y20	G5000_B1_X20_Y20	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	B	1	X21	Y21	G5000_B1_X21_Y21	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	B	2	X22	Y22	G5000_B2_X22_Y22	Givare	Beckhoff
Sahlgrenska	G	5000	B	2	X23	Y23	G5000_B2_X23_Y23	Cirkulationspump	Beckhoff
Sahlgrenska	G	5000	B	2	X24	Y24	G5000_B2_X24_Y24	Fläkt	Beckhoff
Sahlgrenska	G	5000	B	2	X25	Y25	G5000_B2_X25_Y25	Givare	Beckhoff
Sahlgrenska	G	5000	B	2	X26	Y26	G5000_B2_X26_Y26	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	B	2	X27	Y27	G5000_B2_X27_Y27	Belysning_WC	Beckhoff
Sahlgrenska	G	5000	B	2	X28	Y28	G5000_B2_X28_Y28	Larm	Beckhoff
Sahlgrenska	G	5000	B	2	X29	Y29	G5000_B2_X29_Y29	Ventilställdon	Beckhoff
Sahlgrenska	G	5000	B	2	X30	Y30	G5000_B2_X30_Y30	Fläkt	Beckhoff

Autogenerering av PLC kod

Utveckling av verktyg med Visual Basic for Applications

Examensarbete inom högskoleingenjörsprogrammet Elektroteknik

LINNEA KNÖÖS
ANNA MIDTBÖ

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020

EXAMENSARBETE INOM ELEKTROTEKNIK 2020

Autogenerering av PLC kod

Utveckling av verktyg med Visual Basic for Applications

LINNEA KNÖÖS
ANNA MIDTBÖ

CHALMERS
UNIVERSITY OF TECHNOLOGY

Institutionen för Elektroteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2020

Autogenerering av PLC kod
Utveckling av verktyg med Visual Basic for Applications
LINNEA KNÖÖS
ANNA MIDTBÖ

Handledare: Jacob Thorsell, Acobia AB
Examinator: Veronica Olesen, Institutionen för Elektroteknik

EXAMENSARBETE INOM ELEKTROTEKNIK 2020
Institutionen för Elektroteknik
Chalmers tekniska högskola
SE-412 96 Göteborg
Telefon: +46 (0)31-772 1000

Framsida: Bild av Excelark
Typeset in LaTeX
Göteborg, Sverige 2020

Sammanfattning

Acobia AB har utvecklat ett eget verktyg för fastighetsstyring genom programvaran TwinCAT2. Verktyget fungerar som ett hjälpmedel för automatgenerering av kod till Beckhoff Programmable Logic Controller, PLC. I dagsläget används verktyget för styrning av bland annat fläktar, belysning och temperatur på Sahlgrenska Universitetssjukhus. Acobia önskar frångå TwinCAT2 till fördel för den nya versionen TwinCAT3. Syftet med projektet är att undersöka möjligheten att skapa ett nytt vektyg för automatiserad kodgenerering för TwinCAT3.

Genom att skapa en prototyp för styrning av belysning i TwinCAT3 studerades möjligheten att likt originalverktyget få ett Excelark att kommunicera med Twin CAT3. Genom att skapa makron i programspråket Visual Basic for Applications, kopplades Excelarket samman med TwinCAT3 som vidare kunde styra en PLC.

Med det gamla Excelarket som utgångspunkt, skapades ett nytt Excelark med tydligare struktur och ett mer användarvänligt utseende. I Excelarket definierades de block som önskades användas i ett potentiellt framtida automatiserat fastighetssystem. De valda blocken namngavs utefter var de ska verka. De valda blocken skapades därefter i TwinCAT3 tillsammans med tillhörande variabellista. Utöver detta sammanlänkades variabelnamnen till rätt ingångar och utgångar på blocken.

Abstract

Acobia AB has developed a tool for Industrial automation of heating, lighting and ventilation through the software TwinCAT2. It provides automatic code generation for Beckhoff Programmable Logic Control, PLC. Currently Acobia uses the tool for controlling fans, lighting and temperature at Sahlgrenska University Hospital, but since the release of the new revision, TwinCAT3, Acobia would like to move away from using TwinCAT2. The purpose of the project is to explore the possibility of creating a new tool which gives automated code generation for TwinCAT3.

A prototype for lighting control was initially created in TwinCAT3. This was to subsequently investigate the possibility of making an Excel file communicate with the updated software in the same way that the old version of the tool achieved. By creating macros in the data language Visual Basic for Applications, the Excel file was linked together with TwinCAT3.

The old Excel file was used as a benchmark when designing the more user friendly structure. The idea of the Excel file was to make future automated systems for properties to work as simply as possible. The user would define the blocks needed for the system, and through Excel, they were given unique names. The selected blocks were then created in TwinCAT3 together with an associated variable list. In addition, the variable names were linked to the correct inputs and outputs on the blocks.

Keywords: Automation, PLC, TwinCAT3, Excel, Property automation

Förord

Denna rapport har utförts som avslutande moment i högskoleingenjörsprogrammet Elektroteknik på Chalmers tekniska högskola. Examensarbetet gjordes i samarbete med företaget Acobia AB under vinterhalvåret 2019-2020.

Vi vill rikta ett speciellt tack till Jacob Thorsell, vår handledare på Acobia AB som varit till stor hjälp under projektets gång. Vi vill även rikta ett stort tack till Veronica Olesen, handledare på Chalmers tekniska högskola för sitt engagemang samt vägledning och stöttning under projektets gång.

Linnea Knöös & Anna Midtbö, Göteborg, Mars 2020

Innehållsförteckning

Förkortningar

AD - Analog till digital
C# - C Sharp
CFC - Continuous Function Chart
CPU - Central Processing Unit
DA - Digital till analog
FBD - Function Block Diagram
PLC - Programmable Logic Controller
PWM - Pulse Width Modulation
SFC - Sequential Function Chart
ST - Structured text
TwinCAT2 - The Windows Control and Automation Technology 2
TwinCAT3 - The Windows Control and Automation Technology 3
VBA - Visual Basic for Applications

1
Inledning

Projektet som utförts i samarbete med Acobia AB rör automatgenerering av kod till PLC för fastighetsstyrning. Projektet har som slutmål att effektivisera och förenkla arbetet kring nya installationer hos Acobias kunder.

1.1 Bakgrund

Idag använder Acobia AB ett egenutvecklat verktyg för att automatgenerera kod till Beckhoff Programmable Logic Controller, PLC, genom programvaran The Windows Control and Automation Technology 2, TwinCAT2.

Verktyget innehåller funktionsblock för styrning av bland annat fläktar, belysning, cirkulationspumpar och spjäll. Inom fastighetsstyrning är dessa block vanligt förekommande. I praktiken använder Acobia dessa specifika block för att styra inomhustemperatur och ljus i salar på Sahlgrenska Universitetssjukhus. Som en del av verktyget och för att definiera standardkomponenter i PLC-programmet används en Excelfil. I Excelfilen namnges variabler som sedan knyts till definierade standardkomponenter. När Excelfilen färdigställts kan programmet genereras.

Dagens moderna PLC har frångått TwinCAT2 till fördel för den nya upplagan TwinCAT3. Till skillnad från TwinCAT2 har TwinCAT3 stöd för högnivåspråk. Dessutom bygger TwinCAT3 på Visual Studio [1].

1.2 Syfte

Syftet med projektet är att undersöka möjligheten att skapa ett nytt verktyg för automatiserad kodgenerering för TwinCAT3. Vektyget ska utvecklas med hjälp av Excel och Visual Basic for Applications programmering. Därtill ska projektet bidra till utveckling av verktyget samt att göra det mer användarvänligt än dess föregångare.

1.3 Mål

- Skapa grundläggande funktionsblock för fastighetsautomatisering
- Sammanlänka PLC och Excel via VBA-kod
- Test av det nya verktyget i TwinCAT3
- Designa ett nytt Excel-ark

1.4 Avgränsningar

TwinCAT3 kommer vara enda programvaran som det framtagna verktyget ska fungera för. Verktyget kommer ej att implementeras i ett verkligt system.

2

Teknisk bakgrund

I följande avsnitt beskrivs och förklaras de komponenter och program som använts under projektets gång. Verktyget ska interagera med Programmable Logic Controller, PLC och därför beskrivs dess funktionalitet samt uppbyggnad.

2.1 Fältbussar

Fältbussar är den teknik som används för att möjliggöra informationsutbyte mellan olika instrument. Det kan ske via digitala signaler eller en kombination av analoga och digitala signaler [2]. Detta är viktigt då stora mängder av information skickas mellan olika givare, regulatorer, instrument och datorer i de anläggningar som strävar efter att ha en högre grad av automatisering i sina fabriker. Då avståndet mellan instrument i en anläggning kan vara stort är det viktigt att denna informationsöverföring sker på ett korrekt sätt. Informationen, såsom bilder och meddelanden skickas seriellt med binära signaler enligt protokoll [2].

Instrument som ska skicka eller ta emot information genom en eller flera fältbussar kräver en inbyggd mikrodator. Mikrodatorn omvandlar analoga mätvärden till digitala signaler och informationen kan skickas iväg på kabeln enligt protokoll [2].

Det finns flera sorters fältbussar vars egenskaper skiljer sig från varandra. Vilken sort som lämpar sig bäst är varierande och beror ofta på tillämpningen [2].
På Sahlgrenska Universitetssjukhus använder sig Acobia AB utav Ethernet for Control Automation Technology, EtherCAT, vid olika PLC-installationer.

2.1.1 EtherCAT

EtherCAT är utvecklad av företaget Beckhoff och är baserad på Ethernet tekniken för datorkomunikation via en buss [3]. För att överföra datan undersöks först om bussen används. Om detta är fallet väntar den ett förinställt tidsinvervall innan den sedan försöker skicka datan på nytt. Först när bussen inte används överförs datan [4]. Bland de snabbare Ethernet-standarderna är 100BASE-TX den mest förekommande och den som EtherCAT använder sig utav. Då den använder sig utav dataöverföring i två riktningar så har 100BASE-TX den effektiva datahstigheten på 100 Mb/s [5].

2.2 Programmable Logic Controller

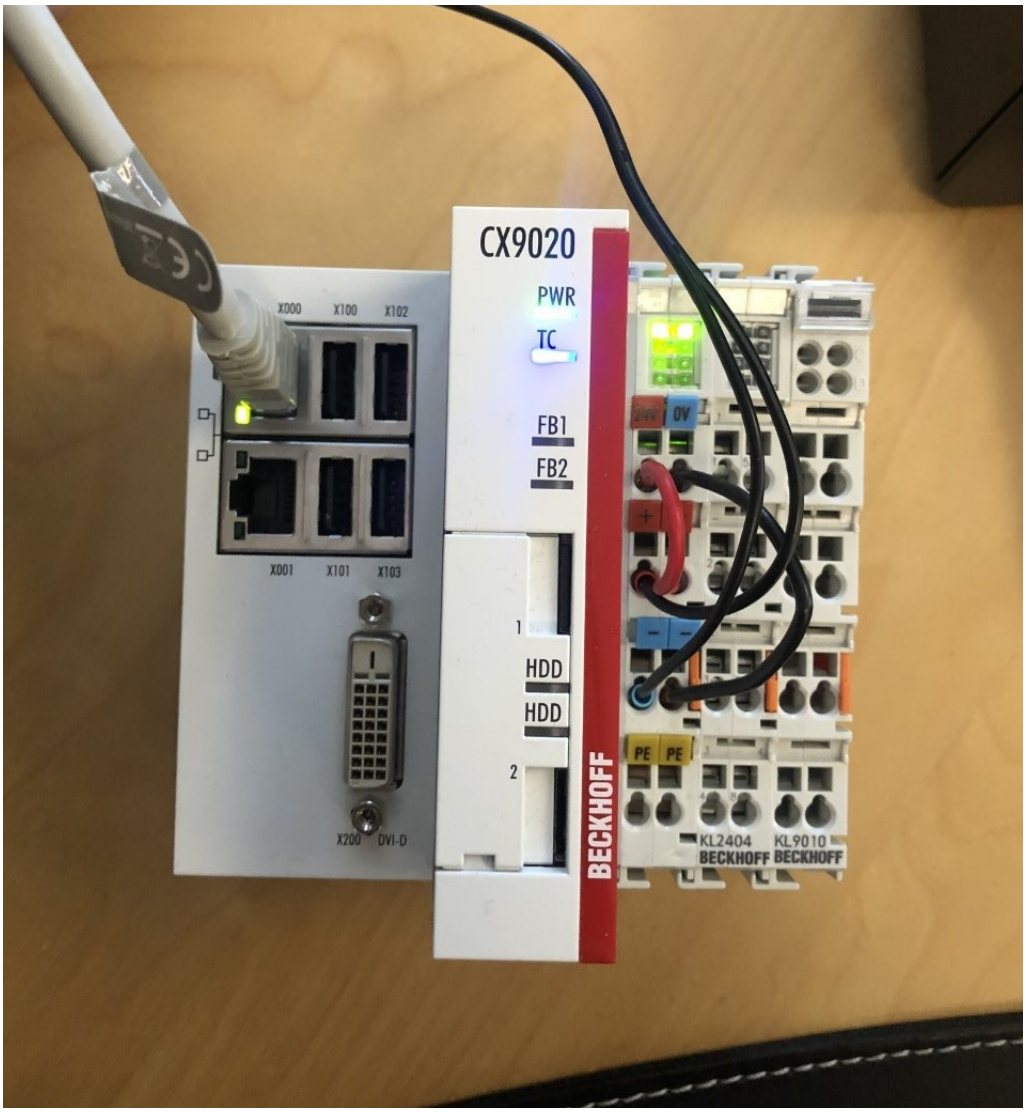
En PLC, programmable logic controller, har två minnen, ett programminne som lagrar instruktioner och ett dataminne som lagrar konstanter och variabler. Dataminnet delas i sin tur in i flyktigt dataminne och icke-flyktigt dataminne. Vid spänningsbortfall försvinner informationen lagrad i ett flyktigt dataminne, men kvarstår i ett icke-flyktigt dataminne [6].

Centralenheten hos en PLC kallas CPU och styr med hjälp av ett operativsystem PLC-systemet. PLC:n jämför programmets ingångar och sätter därefter utgångarna. Den inkommande datan skickas ut på PLC:ns interna buss, där den vidarefördelar datan till respektive enhet som är anslutna till PLC:n [7]. Vid PLC:ns in- och utgångar sitter det även lysdioder för indikering av signalstatus på in- och utgångarna. De underlättar även felsökningen eftersom de kan användas till att lokalisera om felet beror på en yttre koppling eller till styrprogrammet.

In- och ut-gångarna till PLC:n kan vara båda analoga och digitala. De analoga ingångarna från process-sensorer måste gå via en analog till digital omvandlare (AD-omvandlare) likaså behöver de analoga utgångarna till process-ställdon gå via en digital till analog omvandlare (DA-omvandlare).

2.2.1 PLC CX9020

CX9020 är en PLC modell utvecklad av Beckhoff [8], se figur 2.1. Operativsystemet för modellen är Microsoft Windows Embedded Compact 7 och använder sig av TwinCAT som automatiseringprogramvara. Modellen går att driva med eller utan visualisering och är ett kraftfullt PLC- och rörelsekontrollsystem som går att beställas med ett fältbuss-, ljud- eller seriegränssnitt. Dess strömförsörjning ligger på 2A [9]. Om detta inte är tillräckligt så finns möjligheten att koppla in strömförsörjningsmoduler, exempelvis modellen CX1100-000x som genom sin interna buss strömförsörjer alla andra systemkomponenter [1].



Figur 2.1: Modell CX9020 av Beckhoff

2.3 AD-omvandlare

För att göra det möjligt att omvandla en analog signal till en digital serie med ettor och nollor måste signalen samplas [10]. Att sampla signalen innebär att man tar regelbundna stickprov av hela signalen och på så sätt ändras signalen från att vara kontinuerlig till att bli diskret. Tiden mellan dessa stickprov betecknas som T_s och antalet bitar som samplas skrivs som n.

En AD-omvandlare är starkt beroende utav vilken referensspänning, U_{ref}, som används. Genom att dela upp referensspänningen i 2^n ekvidistanta nivåer kan AD-omvandlarens upplösning ΔU beräknas, se ekvation (2.1) [11].

$$\Delta U = \frac{U_{ref}}{2^n} \tag{2.1}$$

Den analoga ingången A_{in}, motsvaras av den digitala utgången D_{ut}, multiplicerat med AD-omvandlarens upplösning ΔU, som visas i ekvation (2.2), där ekvation (2.1) använts för härledning. [11].

$$A_{in} = D_{ut} \cdot \Delta U = D_{ut} \cdot \frac{U_{ref}}{2^n} \tag{2.2}$$

För att få det producerade binära talet så skrivs ekvation (2.2) om på följande sätt:

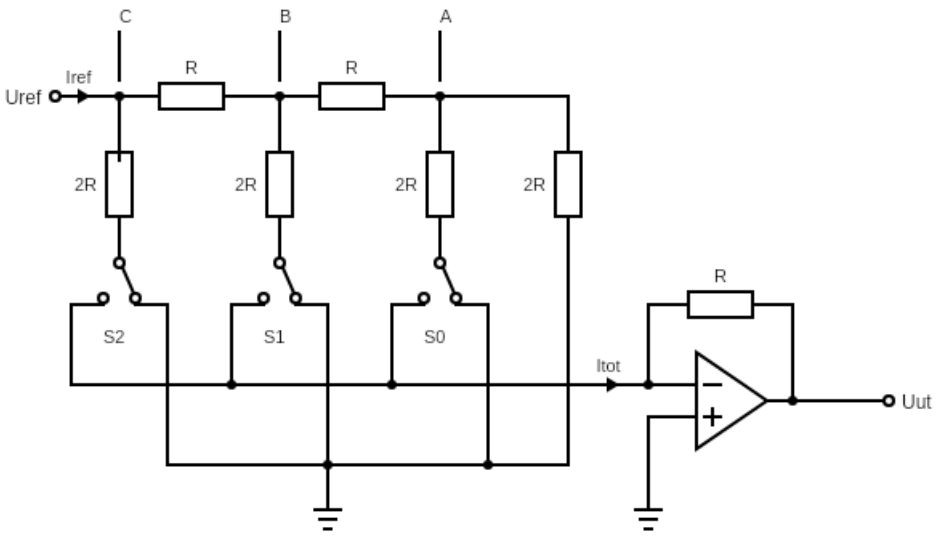
$$D_{ut} = \frac{A_{in}}{\Delta U} = \frac{A_{in}}{U_{ref}} \cdot 2^n \tag{2.3}$$

2.4 DA-omvandlare

Den vanligaste metoden för DA-omvandlig är $R-2R-stege$ metoden. Integrering av en pulse width modulation (pwm) signal är en annan vanlig metod som ofta används i mikrokontrollers.

2.4.1 R-2R-stege

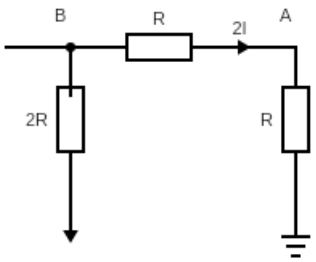
Eftersom en $R-2R-$stege endast använder en resistorstorlek storlek får den bättre precision vid tillverkningen än en DA-omvandlare med viktande resistanser vars koppling avvänder sig utav resistansvärden inom flera tiopotenser [11]. En $R-2R-$stege bygger på Kirchhoffs första lag, Kirchhoffs strömlag och figur 2.2 visar en 3-bitars DA-omvandlare som använder sig utav en $R-2R-$stege.



Figur 2.2: R-2R-stege för en 3 bitars DA-omvandlare

I punkt A i figur 2.2 går det att se att den inkommande strömmen från vänster kommer delas jämnt på mitten och fördelas i respektive gren. Detta sker när det är två lika stora motstånd ner till jord från den punkten. Om den inkommande strömmen är $2I_0$, går I_0 ner i varje gren och S_0=I_0. Eftersom de två 2R resistorerna är parallellkopplade blir den totala resistansen i punkt A lika med R vilket går att se i ekvation (2.4). Figur 2.3 visar det ekvivalenta kretsschemat för punkt B. Eftersom resistorn R i punkt B är seriekopplad med den totala resistansen R i punkt A går det att skriva om det ekvivalenta kretsschemat för punkt B med ett motstånd på 2R, se ekvation (2.5).

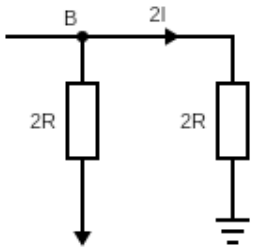
$$2R//2R = \frac{2R * 2R}{2R + 2R} = \frac{4R^2}{4R} = R \tag{2.4}$$



Figur 2.3: Ekvivalenta kretsschemat för punkten B

$$R_{tot} = R + R = 2R \tag{2.5}$$

Figur 2.4 visar det ekvivalenta kretsschemat för punkt B med resistansen 2R. Då punkt B är likadan som punkt A innebär det att även i denna punkt är den totala resistansen ner till jord R och strömmen som går ner i varje gren kommer att vara lika stor. Eftersom $2I_0$ gick till punkt A måste strömmen som går ner från punkt B vara lika stor. Den inkommande strömmen till punkt B måste därför vara $4I_0$ enligt Kirchhoffs första lag [12]. Samma process för att ändra det ekvivalenta kretsschemat för punkt B sker även i punkt C. Resultatet blir detsamma förutom storleken på strömmen som går ner i de olika grenarna. Då den inkommande strömmen till punkt B behöver vara $4I_0$ kommer även strömmen som går ner till sensorn från punkt C vara $4I_0$. Den inkommande strömmen till punkt C som är I_{ref} kommer därmed vara $8I_0$.



Figur 2.4: Förenklat ekvivalenta kretsschemat för punkten B

Eftersom att den totala resistansen till höger om varje punkt är R underlättas uträkningen av värdet för I_0. Detta görs i punkt C med hjälp av U_{ref} och faktumet att den totala resistansen till höger om punkten är R, se ekvation (2.6).

$$U_{ref} = 8 \cdot I_0 \cdot R \Rightarrow I_0 = \frac{1}{8} \cdot \frac{U_{ref}}{R} \tag{2.6}$$

De tre switcharna är det som används för att avgöra vad det binära talet är. Om S_0 står till vänster blir det binära talet 001. Om S_1 står till vänster är det binära talet istället 010. Om både S_0 och S_1 står till vänster är det binära talet 011. Det är endast strömmen från de switchar som står till vänster som når operationsförstärkaren och som därmed påverkar utsignalen. Det anmäla uttrycket för utsignalen med n bitar visas i ekvation (2.7).

$$A_{ut} = -\frac{D_{in}}{2^n} \cdot U_{ref} \qquad D_{in} = 0, 1, 2, ..., 2^n - 1 \tag{2.7}$$

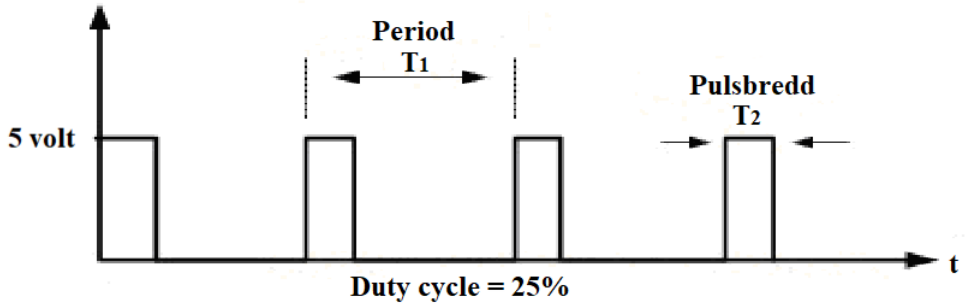
DA-omvandlarens upplösning definieras som spänningsändringen på utgången när den binära insignalen ändras ett steg. Den beräknas på samma sätt som upplösningen för AD-omvandlaren, se ekvation (2.1).

För att få $R - 2R-$stegen att använda sig av flera bitar behövs endast en till knytpunkt som är konstruerad på samma sätt som de andra punkterna adderas till höger om punkten C, med två resisorer på storleken R och 2R och en switch.

2.4.2 PWM-signal

De flesta mikrokontrollers har inbyggda AD-omvandlare, men det förekommer sällan att de har inbyggda DA-omvandlare. Istället brukar de vara försedda med en *pwm*-utgång som ger en pulsbreddsmodulerad signal [11].

Perioden för signalen är konstant däremot går det att påverka signalens duty cycle, kvoten mellan pulsbredden och periodtiden angiven i procent, se figur 2.5 och ekvation (2.8).

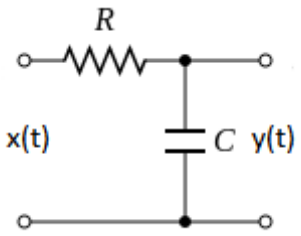


Figur 2.5: Puls med duty cycle på 25%

$$Dutycycle = \frac{T_2}{T_1} \cdot 100\% \tag{2.8}$$

När signalens medelvärde är proportionell mot duty cyclen, som visas i ekvation (2.9), fås en spänning som är proportionell mot duty cyclen. Därmed behövs endast ett filter som har medelvärdsbildande egenskaper på *pwm*-utgången för att skapa en DA-utgång. Exempelvis går det att använda ett enkelt resistor-kondensator-filter (RC-filter) som det medelvärdsbildande filtret [11]. Ett sådant filter går att se i figur 2.6

$$\frac{1}{T_1} \cdot \int_0^{T_1} x(t)dt = \frac{1}{T_1} \cdot \int_0^{T_2} 1dt = \frac{1}{T_1} \cdot [t]_0^{T_2} = \frac{T_2}{T_1} \tag{2.9}$$



Figur 2.6: Kretsschema över ett RC-filter i form av ett lågpassfilter

Ekvation (2.10) visar hur filtret fungerar som en integrator när RC $>>$ T_1.

$$RC >> T_1 = \frac{2\pi}{\omega_1} \Rightarrow R >> \frac{1}{\omega_1 C} \Rightarrow R + \frac{1}{\omega_1 C} \approx R \tag{2.10}$$

2.5 TwinCAT

TwinCAT2 är ett mjukvarusystem utvecklat av den tyska PLC leverantören Beckhoff. Systemet är en plattform som alla Beckhoffs styrsystem använder sig av för programmering av PLC. TwinCAT3 är uppföljaren till TwinCAT2, med möjlighet till fler programmeringsspråk än sin föregångare. Under projektets gång användes de grafiska programspråken CFC, continuous function chart och FBD, function block diagram.

2.5.1 Visual Studio

Det är Microsoft som ligger bakom programutvecklingsmiljön Visual Studio. TwinCAT3 kan integreras med Visual studio och därmed kan högnivåspråk användas vid programmering. Visual Studio stödjer en rad olika programspråk, däribland Visual Basic.

2.6 VBA

Visual Basic for Applications är ett programmeringsspråk utvecklat av Microsoft för att skriva makros. Ett makro är en samling av instruktioner som är programmerade att automatisera kommandon som tidigare utförts manuellt i Excel. Exempel på sådana manuella kommandon kan vara förflyttning av data mellan olika celler eller beräkna antal gånger ett specifikt ord förekommer i en kolumn. Genom en inspelningsfunktion kan användaren skapa egna makron som skrivs som VBA instruktioner. Detta är fördelaktig om användaren vet hur instruktionen ska utföras i Excel, men inte hur koden för den ser ut.

2.7 DiffMerge

DiffMerge är ett verktyg skapat av Sourcegear. Programmet gör det möjligt för användaren att jämföra filer med varandra. Programmets funktion är att underlätta för användaren att urskilja skillnader mellan filerna på ett grafiskt sätt genom att rödmarkera ändringarna. DiffMerge går att använda på Windows, Mac OS X och Linux [13].

2.8 Sensorer

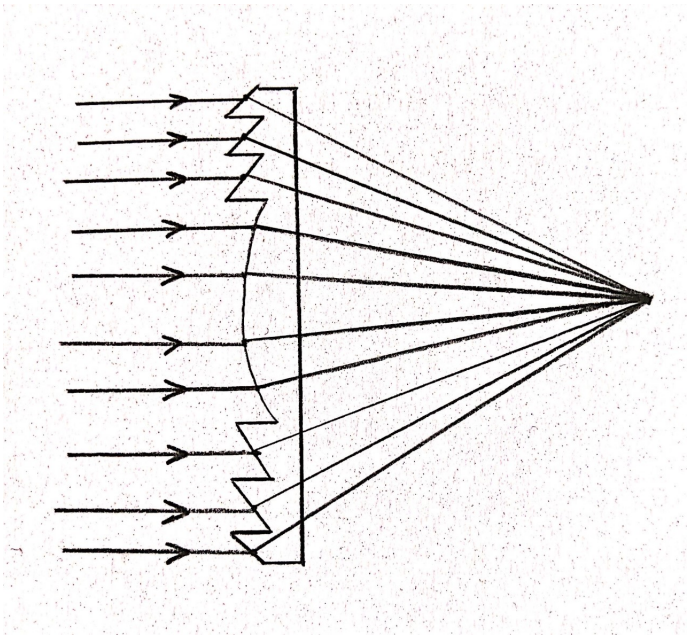
Detektorer har funnits länge, men av mekanisk karaktär. Det kunde handla om en snubbeltråd för att påvisa en människa eller djurs närvaro. I dagens samhälle förlitar man sig istället till elektronik då antalet faktorer att detektera, har breddats.

Utan sensorer skulle tillverkningsprocesserna hos dagens moderna industrier inte vara möjliga att driva. Sensorer behövs för att mäta storheter som nivå, tryck och temperatur och är avgörande för en fungerande tillverkningsprocess.

Likt tillverkningsindustrin har sensorer en stor roll inom fastighetsstyrning. Projektet berör sensorer som detekterar rörelse och vidare följer en beskrivning av funktionaliteten hos en vanligt förekommande variant.

2.8.1 Passiv infraröd detektor

Värme från en människa eller ett objekt kan detekteras med en passiv infraröd detektor, en så kallad PIR-detektor. Funktionen bakom en PIR-detektor är pyroelektricitet där ojämnt formade jonkristaller skapar elektrisk spänning om de utsätts för värme [14]. Sensorn består av två stycken bredvid varandra placerade sådana kristallelement. Framför dessa placeras en lins, för att bredda sensorns registreringsfält. För att koncentrationen av den registrerade IR-strålningen i kristallelementen ska bli så hög som möjligt används en lins. [15]. En Fresnel-lins består av prismor placerade likt en kedja efter varandra och fungerar likt en konvex lins men dess konstruktion gör den näst in till platt. [16]. Se illustration av en Fresnel-lins i figur 2.7.



Figur 2.7: IR-strålning genom en Fresnel-lins

Ett tomt registreringsområde innebär lika potential över båda kristallelementen då de registrerar lika mängd IR-strålning från den omkringliggande miljön. Eftersom kristallelementen är placerade bredvid varandra kommer de att registrera värmen från ett objekt vid olika tidpunkt, beroende på ifrån vilket håll objektet närmar sig. Eftersom objektet är i rörelse kommer utslag ske på kristallelementen efter varann. Spänningen som genereras i sensorn skiftar därmed mellan olika värden och utgången går hög. Då objektet därefter befinner sig utom räckhåll för kristallelementens detetkteringsområde går utsignalen låg. Den sekventiella uppvärmningen av kristallerna resulterar i en temperaturskillnad samt en laddningsskillnad. För

att registrera spänningsskillnaderna används en MOSFET-transistor och signalen därifrån kan användas för att låta en lampa tändas eller en dörr öppnas [15].

3
Metod

I följande avsnitt beskrivs projektets tillvägagångssätt samt de metoder som använts för att nå delmålen och uppfylla syftet med projektet. För att få en grundläggande förståelse för de ingående programvaror som användes i det ursprungliga verktyget efterforskades dessa via internet, böcker samt genom att öppna orginalverktyget i respektive programvara och studera.

Utöver att det nya verktyget skulle kunna användas på samma sätt som orginal-verktyget fanns inga övriga kravspecifikationer angivna. Verktyget bestod av kod programmerad i VBA som sammankopplade ett Excelark med TwinCAT2.

De ursprungliga funktionsblocken byggda i TwinCAT2 var CFC-block. Dessa byggdes delvis som FBD-block i den nya programvaran TwinCAT3. En del av funktionsblocken var standardblock utvecklade både av Beckhoff och Acobia för TwinCAT2 och krävde god programkännedom för att återskapas i TwinCAT3. Alltså block som sedan innan fanns att tillgå i Beckhoffs bibliotek.
Blocken tillhörande TwinCAT2 behövdes modifieras för att fungera till TwinCAT3, därför skapades ett eget grundläggande funktionsblock.

När ett fungerande funktionsblock skapats undersöktes förmågan att via VBA-programmering ändra exempelvis adressering av in- och ut-parametrar. På andra änden av VBA programmeringen fanns Excelarket. Excelarket fanns till för användaren och skulle vara en samlad plats där samtliga ändringar av PLC-programmet utförts. Det gamla verktygets Excelark återanvändes inte, utan ett nytt ark skapades anpassat till det nyskapade funktionsblocket.

För att få en helhetsbild av det system vars funktion verktyget ligger till grund för besöktes Sahlgrenska Universitetssjukhus. Där studerades de PLC:s Acobia installerat för styrning av fläktar som försörjde allmänna utrymmen och operations-salar med rätt temperatur. Vidare studerades användningen av fältbussar i installationen.

4
Genomförande

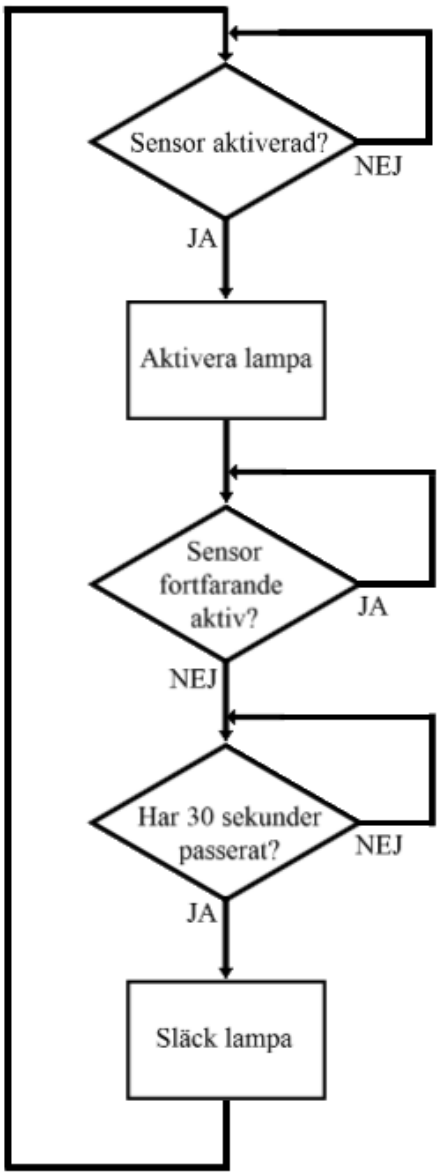
I följande avsnitt presenteras en utförlig beskrivning av projektets arbetsgång som i många fall utförts stegvis. Dessutom redogörs de för dem tankegångar såväl resonemang som förts under projektet.

4.1 Skapa ett grundläggande funktionsblock

Ett funktionsblock skapades med syfte att sköta styrning av belysning i badrum. Funktionsblocket hade följande specifikation: I varje badrum ska en sensor placeras, som vid rörelse aktiverar lamporna. 30 sekunder efter att en sensor slutat att registrera rörelse ska ljuset i badrummet släckas.

I figur 4.1 visas flödesschemat för funktionsblockets program. Programmet startar med att kontrollera om någon av sensorerna har aktiverats. Om inte, fortsätter programmet kontinuerligt att kontrollera om någon av sensorerna har aktiverats. Om en sensor har aktiverats, medför detta i sin tur att två lampor tänds. Sensorn kommer att återaktiveras oavbrutet fram tills dess att det inte sker någon rörelse i badrummet. I samband med att en sensor har aktiverats startas en timer inställd på 30 sekunder. Om sensorn återaktiveras inom loppet av 30 sekunder återställs timern och startar om från början. Efter att 30 sekunder passerat deaktiveras sensorn och ljuset släcks.

Funktionsblocket styrde endast ett badrum men kunde återanvändas fler gånger för att styra fler badrum, då anpassades endast in- och utgångarna därefter. Funktionsblocket hade totalt en ingång, en sensor, som representerade den sensor som tillhörde badrummet. Programmet hade två lampor som utgångar.



Figur 4.1: Flödesschema för styrning av belysning

4.2 Excelarkets funktion

Vid kompilering i TwinCAT3 genererades en fil från PLC-programmet som innehåller råtext. Denna fil ligger till grund för kommunikation mellan Excel och TwinCAT3. Vid närmare undersökning upptäcktes det att ändringar som skett i råtexten ledde till ändringar i PLC-programmet. Syftet med Excelarket är att smidigt kunna addera funktionsblock, ändra adresser samt addera variabler till PLC-programmet, vilket möjliggörs med VBA-programmering som redigerar i råtexten.

Det gamla verktyget kunde endast skapa funktionsblock med tillhörande variabellista. Vid utveckling av det nya verktyget var målet att knyta variabelnamnen direkt till funktionblockens in- och ut-gångar samt att möjliggöra adressändring. Tidigare sattes adresserna manuellt genom att redigera adresserna direkt i TwinCAT, vilket kunde vara tidskrävande.

4.3 Försök till adressändring

Det gjordes ett felaktigt antagande att adressplatserna kunde redigeras i samma råtext som funktionblocken och variablerna redigerades i. Varje funktionsblock som infogats i råtexten hade en rad som kallades 'adress'. Via VBA-kod redigerades denna specifika adressplats men ändringarna påverkade inte PLC programmet som förväntat. Därmed konstaterades det att råtexten inte var den fil där adressändringar skedde.

Vid vidare efterforskning, med hjälp av programmet DiffMerge, lokaliserades en fil där det skedde ändringar efter att adresserna ändrats manuellt i TwinCAT3. Filen innehöll få radbrytningar och att lokalisera de exakta platser där en adressändring gjordes, var inte möjlig via DiffMerge, se figur 4.2. Detta försvårade att göra adressändringar i Excel. Adressändring kunde således inte realiseras med detta verktyg, utan precis som i det gamla kommer adresser endast kunna sättas direkt i programmet TwinCAT3. Däremot förverkligades valmöjligheten att namnge variabler, bestämma vilken typ de var samt att kunna ange dem som ingångar eller utgångar. Detta realiserades på samma tillvägagångssätt som vid försöket till adressändring.

Figur 4.2: Fil där adressändring förekom

5

Design av det nya Excelarket

Excelarket i orginallösningen var någorlunda invecklad för en person som inte tidigare varit involverad i projektet. Det innehöll flertalet kolumner med liknande namn och irrelevanta variabelnamn. Makrosen var långa och sparsamt kommenterade.

Efter att det konstaterats att verktyget inte kunde användas för att påverka variablernas adresser, kändes dess användningsområde begränsat. Visionen var att det nya Excelarket skulle ha samma funktioner som det gamla och att adress-ändringen skulle bli ett nytt och användbart tillskott. Likt det gamla Excelarket programmerades det nya till att skapa variabellista och infoga block i TwinCAT3 utefter vad användaren specificerat i Excelarket.

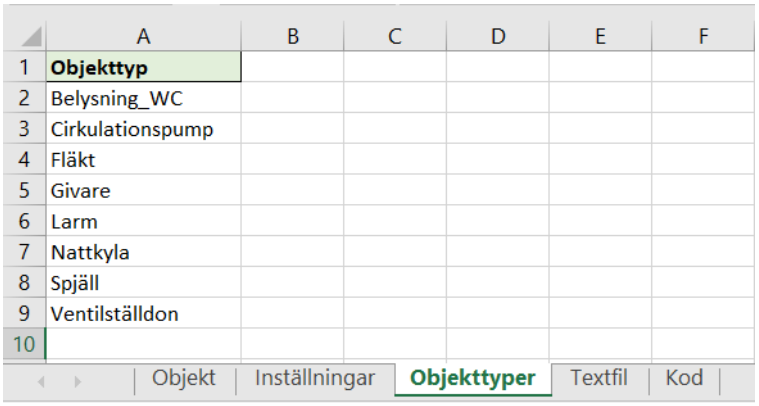
Låt säga att en fastighet ska utrustats med ett nytt ventilationssystem med tillhörande fläktar och spjäll. Det nya Excelarket ger användaren möjligheten att välja vilket block som ska skapas och namnge det efter vilket syfte det har. Användaren väljer 3 fläktar som ska användas för våning 2 i byggnad A. Användaren vill också ha 3 tillhörande spjäll. När antal och sort är valda skapas dessa block direkt i en main-fil för ventilationsprogrammet i TwinCAT3. Utöver detta skapas även en tillhörande variabellista för de ingående parametrarna. Variablerna fick unika namn efter parametrarna valda i Excelarket. Vad som skiljer det nya Excelarket från det gamla är att variabelnamnen knyts direkt till blocken, samt att blocken namnges unikt utefter vad användaren valt.

Eftersom funktionerna bakom Acobias egna fastighetsblock inte fanns tillgängliga och därmed aldrig återskapades i TwinCAT3, programmerades makrot efter det grundläggande funktionsblock för belysning som tidigare nämnts. Koden i makrot förbereddes för att kunna använda sig av de fastighetsblock som Acobia skapat sen tidigare och inte bara för det grundläggande belysningsblocket. Den informationen som krävs för att i framtiden ange fastighetsblocken i Excelarket, är antal rader som varje block är uppbyggt av samt var i dessa rader variabelknytningen sker.

Excelarket innehåller fem blad, endast tre av dem är direkt intressanta för användaren. Resterande två blad arbetar i bakgrunden och innehåller information som ligger till grund för att makrot ska fungera. Nedan introduceras bladen djupgående.

5.1 Blad: Objekttyper

Under bladet Objekttyper listas de olika funktionsblock som finns tillgängliga att välja mellan under bladet Objekt. Här listas de för att skapa en rullgardinsmeny. Det finns även möjlighet att addera fler funktionsblock om behovet för fler varianter av funktionsblock skulle uppstå i framtida projekt. I figur 5.1 visas innehållet i bladet Objekttyper.



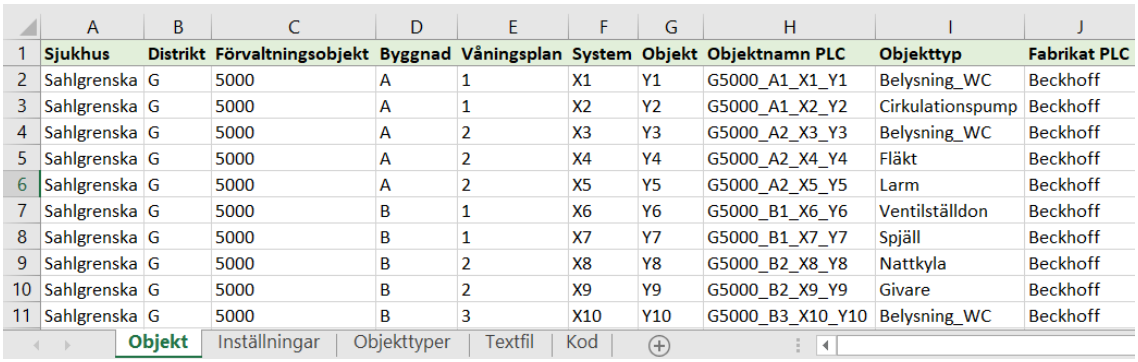
	A	B	C	D	E	F
1	**Objekttyp**					
2	Belysning_WC					
3	Cirkulationspump					
4	Fläkt					
5	Givare					
6	Larm					
7	Nattkyla					
8	Spjäll					
9	Ventilställdon					
10						

Objekt | Inställningar | **Objekttyper** | Textfil | Kod

Figur 5.1: Excelbladet Objekttyper

5.2 Blad: Objekt

I Excelbladet Objekt har en struktur för namnangivelse av variabelnamn skapats. Först anges vilket distrikt, sjukhus, byggnad och våningsplan det valda blocket ska tillhöra. Här sker även valet av block-sort. Slutligen sammansätts kolumnerna och bildar ett namn för det specificerade block som valts. Figur 5.2 visar bladet Objekt.



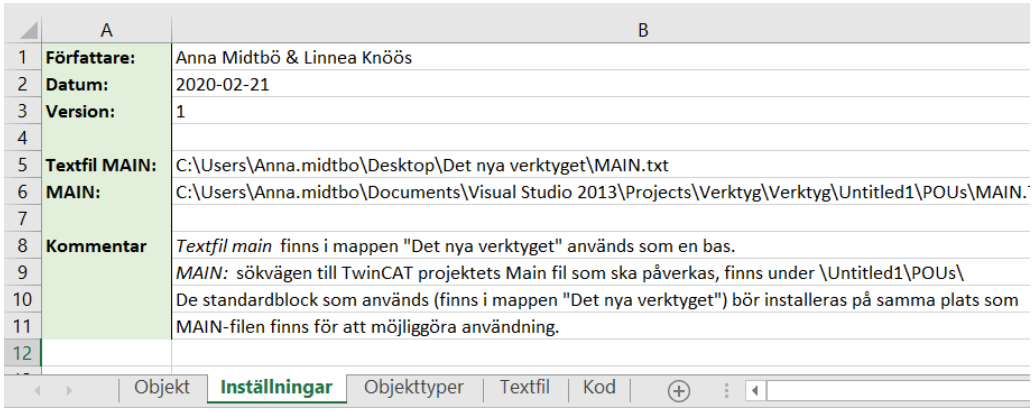
	A	B	C	D	E	F	G	H	I	J
1	**Sjukhus**	**Distrikt**	**Förvaltningsobjekt**	**Byggnad**	**Våningsplan**	**System**	**Objekt**	**Objektnamn PLC**	**Objekttyp**	**Fabrikat PLC**
2	Sahlgrenska	G	5000	A	1	X1	Y1	G5000_A1_X1_Y1	Belysning_WC	Beckhoff
3	Sahlgrenska	G	5000	A	1	X2	Y2	G5000_A1_X2_Y2	Cirkulationspump	Beckhoff
4	Sahlgrenska	G	5000	A	2	X3	Y3	G5000_A2_X3_Y3	Belysning_WC	Beckhoff
5	Sahlgrenska	G	5000	A	2	X4	Y4	G5000_A2_X4_Y4	Fläkt	Beckhoff
6	Sahlgrenska	G	5000	A	2	X5	Y5	G5000_A2_X5_Y5	Larm	Beckhoff
7	Sahlgrenska	G	5000	B	1	X6	Y6	G5000_B1_X6_Y6	Ventilställdon	Beckhoff
8	Sahlgrenska	G	5000	B	1	X7	Y7	G5000_B1_X7_Y7	Spjäll	Beckhoff
9	Sahlgrenska	G	5000	B	2	X8	Y8	G5000_B2_X8_Y8	Nattkyla	Beckhoff
10	Sahlgrenska	G	5000	B	2	X9	Y9	G5000_B2_X9_Y9	Givare	Beckhoff
11	Sahlgrenska	G	5000	B	3	X10	Y10	G5000_B3_X10_Y10	Belysning_WC	Beckhoff

Objekt | Inställningar | Objekttyper | Textfil | Kod

Figur 5.2: Excelbladet Objekt

5.3 Blad: Inställningar

Under bladet Inställningar finns namn på författarna, version samt datum då verktyget skapats. Utöver det anges en sökväg till den main-fil som skapas när noll funktionblock förekommer i programmet. Sökvägen till TwinCAT-projektets POU-mapp, innehållande samtliga filer för de fastighetsblock som ska användas, finns också angiven, se figur 5.3. Genom att låta användaren redigera sökvägarna direkt i detta Excelblad, behöver inte redigeringen ske i VBA koden. Därmed reduceras risken att något i koden ändras oavsiktligt. Dessutom ska verktyget kunna brukas på fler datorer än den verktyget skapades på.



Figur 5.3: Excelblad Inställningar

5.4 Blad: Kod

Vid kompilering av ett PLC-program skapas en fil med textrader. Filer skapade från PLC-program med noll block, ett block och två block jämfördes i programmet DiffMerge. Detta gjordes för att studera hur innehållet i filerna påverkades av antalet block som använts. Vid användning av minst ett block, visade DiffMerge på att det adderats rader med text till filens slut. Vidare visade DiffMerge att dessa ändelserader förekom en gång oavsett antal adderade block och förekom ingen gång om noll block hade adderats. Ändelseskillnaderna som lokaliserades i DiffMerge sparades i Excelbladet Kod för att återkommande kunna användas, se figur 5.4.

	A
1	Ändelsekod för användning av en/flera block
2	<Type n="Boolean">System.Boolean</Type>
3	<Type n="BoxTreeAssign">{9873c309-1f09-4ebf-9078-42d8057ef11b}</Type>
4	<Type n="BoxTreeBox">{acfc6f68-8e3a-4af5-bf81-3dd512095a46}</Type>
5	<Type n="BoxTreeOperand">{9de7f100-1b87-424c-a62e-45b0cfc85ed2}</Type>
6	<Type n="Flags">{668066f2-6069-46b3-8962-8db8d13d7db2}</Type>
7	<Type n="Int32">System.Int32</Type>
8	<Type n="Int64">System.Int64</Type>
9	<Type n="Network">{d9a99d73-b633-47db-b876-a752acb25871}</Type>
10	<Type n="NWLImplementationObject">{25e509de-33d4-4447-93f8-c9e4ea381c8b}</Type>
11	<Type n="Operand">{c9b2f165-48a2-4a45-8326-3952d8a3d708}</Type>
12	<Type n="Operator">{bffb3c53-f105-4e85-aba2-e30df579d75f}</Type>
13	<Type n="OutputItemList">{f40d3e09-c02c-4522-a88c-dac23558cfc4}</Type>
14	<Type n="ParamList">{71496971-9e0c-4677-a832-b9583b571130}</Type>
15	
16	
17	

Figur 5.4: Ändelseraderna i Excelbladet Kod

Utöver att agera plats att förvara textrader används bladet Kod för att skapa de in- och utsignaler som behövs till de valda blocken och att namnge dem utefter vad som specificerats. I figur 5.5 och 5.6 visas de celler som använder objektnamnet som angetts i Excelbladet Objekt. Till namnet adderas en ändelse som påvisar dess användningsområde, vilken typ samt om det är en ingång eller utgång.

B	C	D	E
VAR	Belysning Namn	Lampa 1	Lampa 2
VAR_INPUT	G5000_A1_X1_Y1	G5000_A1_X1_Y1_Lampa1 AT %Q*: BOOL;	G5000_A1_X1_Y1_Lampa2 AT %Q*: BOOL;
VAR_OUTPUT	G5000_A1_X5_Y5	G5000_A1_X5_Y5_Lampa1 AT %Q*: BOOL;	G5000_A1_X5_Y5_Lampa2 AT %Q*: BOOL;
END_VAR			

Figur 5.5: Namnger lampa och sätter som BOOL och utgång

F	G
Sensor	Block Namn
G5000_A1_X1_Y1_Sensor AT %I*: BOOL;	G5000_A1_X1_Y1: Belysning_WC;
G5000_A1_X5_Y5_Sensor AT %I*: BOOL;	G5000_A1_X5_Y5: Belysning_WC;

Figur 5.6: Namnger samt sätter sensor som BOOL och ingång

Slutligen används bladet Kod för att göra de ändringar som behövs för att möjliggöra sammanlänkning mellan variablernamn och funktionsblock. Textraden som kopplar en variabel till ett block ser ut på följande sätt för utgången Lampa1:

<v n=”Operand”>”OBJEKTNAMN_Lampa1”</v>

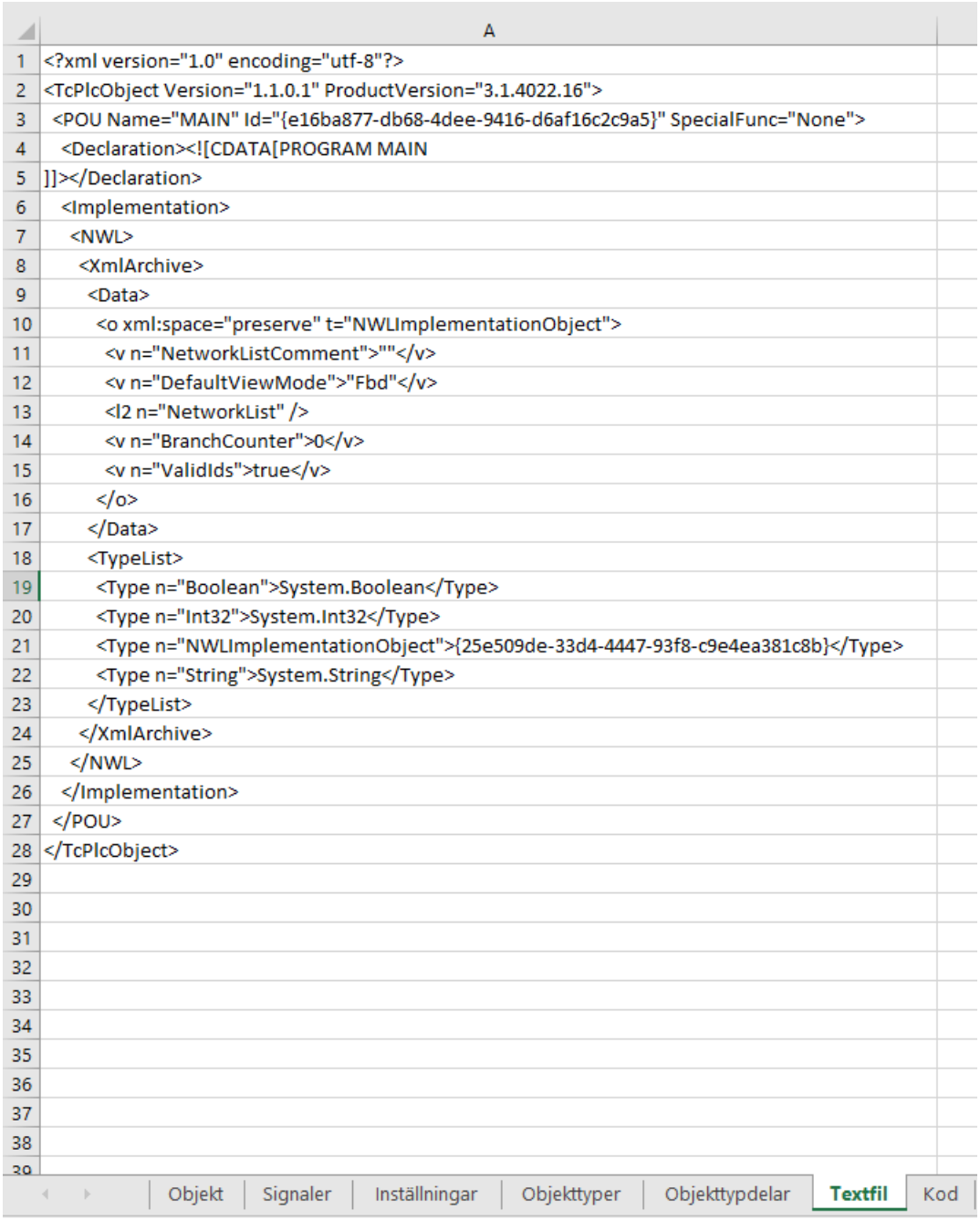
Då det endast är objektnamnets som ska ändras så har raden delats upp i olika celler, se figur 5.7. Objektnamnet sätts i kolumn I och i kolumn K finns den omgjorda textraden.

H	I	J	K
Lampa 1			
<v n="Operand">"	G5000_A1_X5_Y5	_Lampa1"</v>	<v n="Operand">"G5000_A1_X5_Y5_Lampa1"</v>
Blocknamn			
<v n="Operand">"	G5000_A1_X5_Y5	"</v>	<v n="Operand">"G5000_A1_X5_Y5"</v>
Lampa 2			
<v n="Operand">"	G5000_A1_X5_Y5	_Lampa2"</v>	<v n="Operand">"G5000_A1_X5_Y5_Lampa2"</v>
Sensor1			
<v n="Operand">"	G5000_A1_X5_Y5	_Sensor"</v>	<v n="Operand">"G5000_A1_X5_Y5_Sensor"</v>

Figur 5.7: Uppdelning av textrad så att specifika ändringar kan ske

5.5 Blad: Textfil

Som tidigare nämnt skapar TwinCAT3 en råtext när ett nytt PLC-program kompileras. Excelbladet Textfil innehåller en råtext från ett tomt PLC-program. Råtexten kommer hädanefter att kallas MAIN. I figur 5.8 visas MAIN-filens textrader. Dessa textrader uppdateras kontinuerligt när användaren specificerar fastighetblock under bladet Objekt. Under sektion 6, *Verktygets funktionalitet* beskrivs denna kontinuerliga uppdatering närmare.



Figur 5.8: Excelbladet Textfil

6
Verktygets funktionalitet

I följande avsnitt förklaras verktygets funktionalitet samt VBA kodens funktioner. Under projektets gång har följande filer skapats:

- Excelfil som innehåller en mall för Excelarkets struktur
- En add-in fil som innehåller all VBA kod
- Textfilen MAIN
- Funktionsblocket Belysning_WC

Makrot skapat i VBA möjliggör utbytet av information mellan valda parametrar i Excelarket och TwinCAT3. Samtliga filer är nödvändiga för att verktyget ska fungera och är därför sparade tillsammans i en gemensam mapp. När fler funktionsblock i framtiden skapas ska de också sparas där. Funktionsblocken måste sedan bli kopierade till POU-mappen för det PLC projekt som verktyget ska användas för.

Samtliga följande avsnitt beskriver stegvis hur koden fungerar och utförs ej av användaren själv, om inget annat anges.

6.1 Hämta textfil

MAIN-filens grundstruktur som verktyget använder sig av kopieras till en sträng. I Excelbladet Textfil, kolumn A skrivs informationen på strängen ut.

6.2 Ändelseändring

När minst ett block valts i Excelarket krävs först en ändring av MAIN-filens slutrader. Detta sker genom att skapa luckor åt de rader text som ska adderas till grundstrukturen. De textrader som behövs finns under Excelbladet Kod och kopieras därifrån och klistras in på sin rätta plats i textfilen.

6.3 Skapa plats åt angivna block

I Excelbladet Objekt har användaren angett de funktionsblock som önskas användas i PLC programmet. Ett funktionsblock är uppbyggt av ett specifikt antal rader text unikt för det blocket. För funktionsblocket Belysning_WC är antalet rader 134. Dessa 134 rader text som blocket är uppbyggt av finns sparat under Excelbladet Kod.

Precis som vid addering av ändelserader krävs luckor i textfilen för att göra plats åt de rader som funktionsblocken består av. Antalet för ändelseraderna är konstant och det går därför att direkt flytta omkringliggande textrader till specifika celler i Excelbladet, då de alltid kommer förflyttas till samma plats. Antal textrader som sedan ska kopieras in i MAIN baseras på antal angivna block. Radantalet hos de valda funktionsblocken beräknas. Funktionen Insert möjliggör förflyttningen av raderna. Funktionen skriver informationen som är sparad på sista raden och arbetar sig uppåt. För att uppnå detta adderas de rader som redan finns i MAIN till den beräknade storleken på blocken. De rader som behövs flyttas, klipps ut från och placeras på den nya uträknade platsen. Rad 13 och 14 i MAIN-filen är alltid densamma när minst ett block används och ändras därför från början. Teckenkombinationen <o> sätts in på rad A14 och "öppnar"insättningen av blocken.

6.4 Addera block

Blocket Belysning_WC är som tidigare nämnt uppbyggt av 134 rader text. Vid insättning av fastighetsblock varierar innehållet på slutraden beroende på dess placering. Ett block med placering sist i textfilen innehåller slutraden </I2> och påvisar att avsnittet för blockraderna ska stängas. Om fler block av samma typ specificerats ska insättningen av blocken fortsätta att vara öppen, således ska sista raden istället vara <o>. Detta sker genom en If-sats. För att få blocken på rätt plats i MAIN används en funktion som lokaliserar den första tomma raden i filen och adderar blocket där. Vid användning av flera block upprepas processen.

6.5 Skapa plats åt variabellista

Ett funktionsblock behöver ett antal variabler kopplat till dess in- och utgångar för att få det att fungera i PLC-programmet. Platsen för variabellistan skapas på samma sätt som platsen för blocken. Funktionsblocket Belysning_WC använder sig utav fyra variabler, en för att namnge det unika blocket, en ingång för sensorn samt två utgångar för lampor. Antalet rader som Belysning_WC blockets variabellista behöver är därför 4 multiplicerat med antalet funktionsblock av den typ som ska användas. Variabellistan består av ytterligare sex rader kod som innehåller öppning och stängning för insättning av variabler av typen blocknamn, ingångar och utgångar.

6.6 Skapa variabelnamn

Variablerna som används till funktionsblocken ska ha ett unikt namn. I Excelbladet Objekt anger användaren vilket funktionsblock som ska användas men även information om vad blocket ska kontrollera och var det ska installeras. Följande parametrar som distrikt, sjukhus, byggnad och objekttyp anges. En sammanslagning av samtlig information bildar ett funktionsblocks objektnamn. Objektnamnet ligger till grund för funktionsblockets slutliga namn och namnet på variablerna till dess in- och utgångar. Vid användning av flera funktionsblock av samma typ är det nödvändigt att ha tydliga namn på variablerna, då det underlättar vid installation och felsökning.

Variabelnamn skapas för ett funktionsblock i taget. I Excelbladet Objekt undersöks kolumnen Objekttyp radvis. När funktionsblocket vars variabler ska skapas är angivet, kopieras det angivna objektnamnet som står på samma rad i kolumnen Objektnamn. I Excelbladet Kod används det kopierade objektnamnet för att skapa variablerna till funktionsblocket. Vad variabeln ska användas till adderas till dess namn, exempelvis sensor eller lampa. Härnäst kommer ett förtydligande exempel:

Objektnamn = G500_A1_X 1_Y1
Objekttyp = Belysning_WC
Namnet på blocket Belysning_WC = G500_A1_X 1_Y1
Som tidigare nämnts så har Belysning_WC en ingång, Sensor, samt två utgångar, Lampa 1 och Lampa 2.
Ingångens namn: G500_A1_X 1_Y1_Sensor
Utgångarnas namn: G500_A1_X 1_Y1_Lampa1, G500_A1_X_Lampa2

Innan det går att lägga till variablerna i MAIN-filen så behövs även den textrad som definierar om det är en ingång eller utgång, samt vad för typ variabeln är. Funktionsblocket Belysning_WC använder endast variabler av typen BOOL till sina in- och utgångar. För att göra en variabel till en ingång adderas textraden: AT %I*, till utgångar: AT %Q*.

6.7 Addering av variabellista

Efter att samtliga variabler skapats läggs de till i textfilen på ett liknande sätt som funktionsblocken gjorts. Samma funktion för att lokalisera den första tomma raden användes, men istället för att addera alla variabler på en gång så behövs de läggas in gruppvis. Inledningsvis läggs "VAR" till i textfilen, följt av alla blocknamn och avslutas sedan med "END_VAR". Processen upprepas med de övriga två grupperna, enda skillnaden är att "VAR" byttes ut mot "VAR_INPUT" och "VAR_OUTPUT". De avslutas alltid med "END_VAR". Detta sker via en Do-While-sats och en loop.

6.8 Sammanlänkning av variabelnamn och funktionsblock

Alla funktionsblock och dess variabler är nu insatta i textfilen. Det sista som ska göras innan det överförs till TwinCAT3 är att sammanlänka variablerna med det funktionsblock som de tillhör.

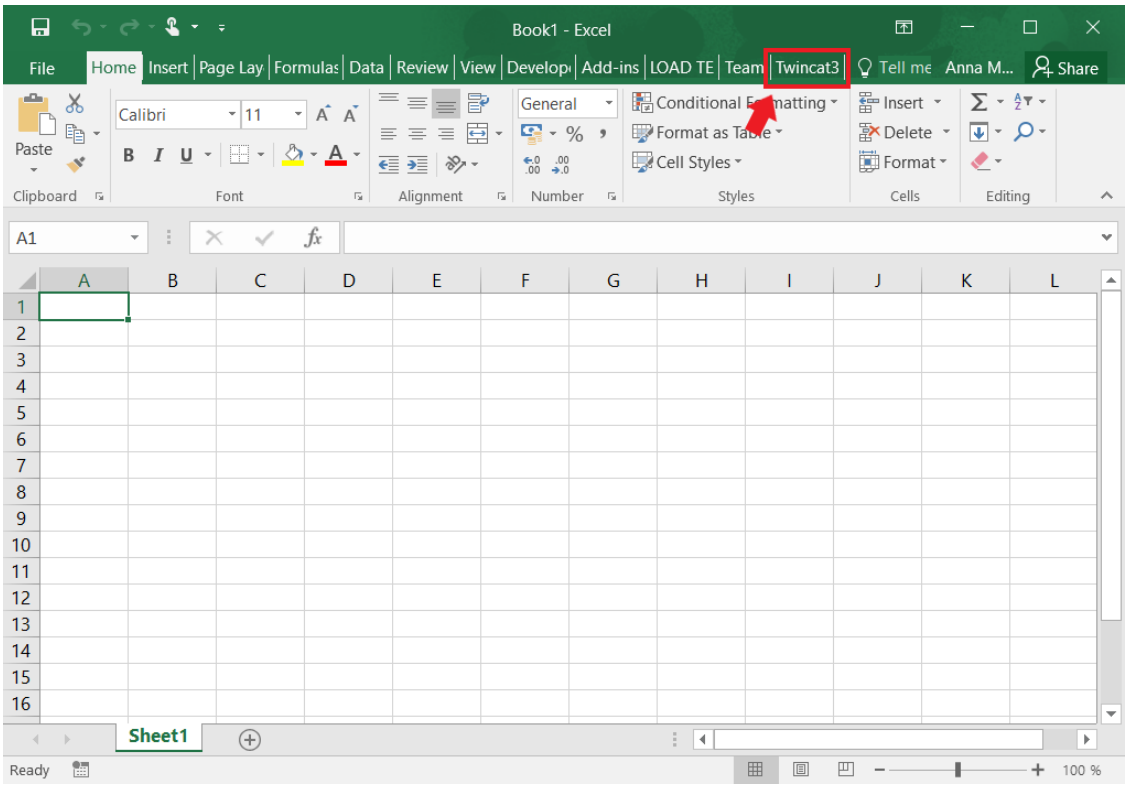
På förhand har DiffMerge använts för att lokalisera var dessa sammanlänkningar sker i textraderna som bygger upp funktionsblocken. För funktionsblocket Belysning_WC sker de vid raderna 14, 39, 58 och 84 i funktionblockets textuppbyggnad. VBA koden använder sig utav en matematisk formel så att den alltid hittar och kan ändra dessa platser oberoende av hur många block som används.
Ändringen sker genom att koden kopierar textraden för sammanlänkningen som finns under Excelbladet Kod och klistrar in den på den uträknade platsen.

6.9 Uppladdning av textfil

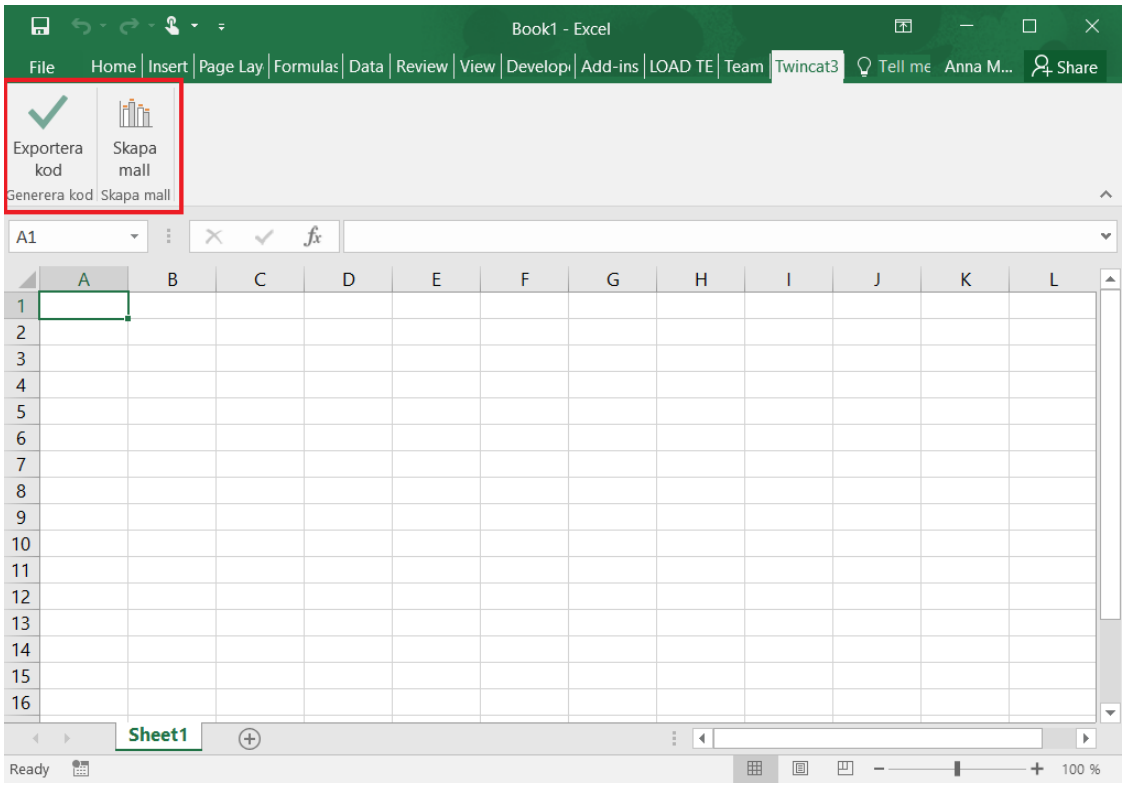
Alla nödvändiga justeringar i Excelbladet Textfil har nu genomförst. Det som återstår är uppladdning av textfilen till PLC-programmets MAIN-fil. Textfilen som skapats skrivs över, rad för rad, genom en Do-while-sats och en loop.

6.10 Add-in

Avslutningsvis skapades ett add-in till Excel som innehåller all VBA-kod och är uppdelad i två funktioner. Platsen där add-in:et förekommer visas i figur 6.1. Principen för add-in är att VBA-koden som skapats ska kunna öppnas i Excel utan tillgång till original Excelarket. Funktionerna visas i figur 6.2. När användaren trycker på ”Skapa mall” återskapas en mall av Excelarket som behövs för att använda verktyget. Trycker användaren på ”Exportera kod” kör programmet och skapar de block som specificeras till TwinCAT3. Detta gör programmet mer stilrent och upplevs professionellt.



Figur 6.1: Plats där add-in nås i Excelark



Figur 6.2: Funktioner för add-in

7
Resultat

Under projektet har det fastställts att MAIN-filen kan, genom VBA-programmering, redigeras för att sätta variabler som in- och utgångar såväl som att ge dem ett värde. Vidare konstaterades det att det ej är möjligt att koppla in- och utgångar till en specifik adress i MAIN-filen, utan genomförs manuellt i TwinCAT3.

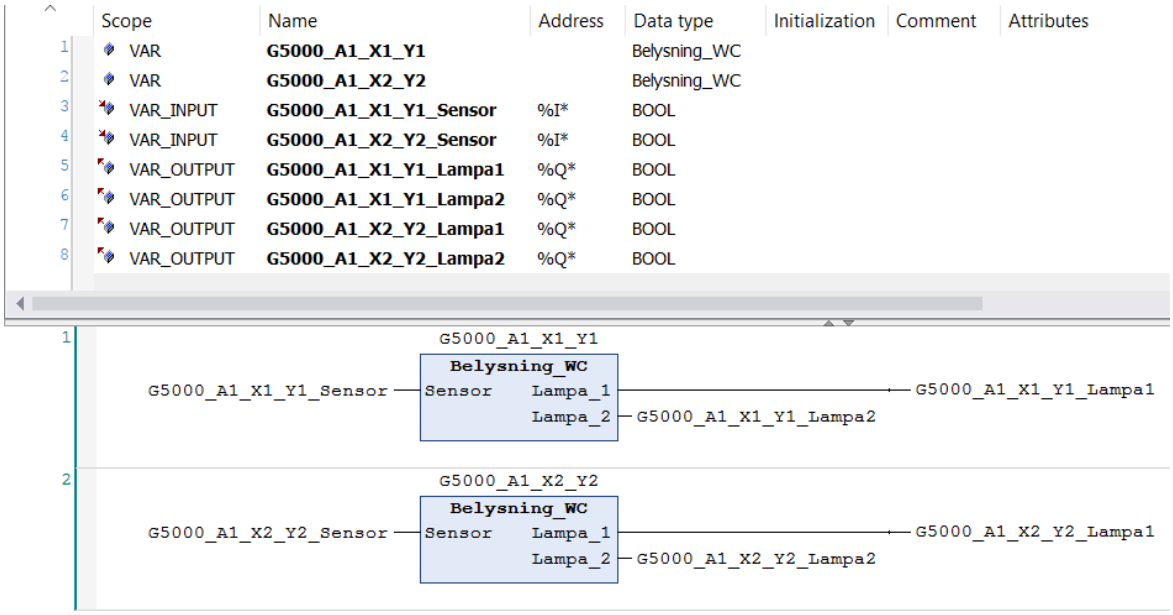
Excelarket har i jämförelse med det ursprungliga verktyget fått ett uppdaterat utseende som förtydligar för användaren. Genom det nya Excelarket kan block namnges efter dess funktionalitet och i vilken byggnad, våningsplan och lokal det ska nyttjas. Därefter infogas de valda blocken från Excel i TwinCAT3. Excelarket kan i dagsläget endast addera block av typen Belysning_WC till PLC då inget annat fastighetsblock ännu gjorts tillgängligt i TwinCAT3. Däremot är koden anpassad för att tillämpa flera olika sorters fastighetsblock. Först när dessa block implementerats i TwinCAT3 kommer de att kunna tillämpas i koden och fungera tillsammans med verktyget.

De skapade blocken har alla egna namn som automatiskt blir knutna till blockets in- och utgångar. Användaren behöver själv endast ändra adresserna och justera variabelnamnen efter behov, exempelvis om flera block ska använda sig utav samma sensor. I figur 7.2 visas resultat för två block specificerade i Excelarket i figur 7.1 .

	A	B	C	D	E	F	G	H	I	J	K
1	**Sjukhus**	**Distrikt**	**Förvaltningsobjekt**	**Byggnad**	**Våningsplan**	**System**	**Objekt**	**Objektnamn PLC**	**Objekttyp**	**Fabrikat PLC**	**Objektbeskrivning**
2	Sahlgrenska	G	5000	A	1	X1	Y1	G5000_A1_X1_Y1	Belysning_WC	Beckhoff	
3	Sahlgrenska	G	5000	A	1	X2	Y2	G5000_A1_X2_Y2	Belysning_WC	Beckhoff	
4											
5											
6											

Objekt | Signaler | Inställningar | Objekttyper | Objekttypdelar | Textfil | Kod

Figur 7.1: Inställningar för användning av två block



	Scope	Name	Address	Data type	Initialization	Comment	Attributes
1	VAR	**G5000_A1_X1_Y1**		Belysning_WC			
2	VAR	**G5000_A1_X2_Y2**		Belysning_WC			
3	VAR_INPUT	**G5000_A1_X1_Y1_Sensor**	%I*	BOOL			
4	VAR_INPUT	**G5000_A1_X2_Y2_Sensor**	%I*	BOOL			
5	VAR_OUTPUT	**G5000_A1_X1_Y1_Lampa1**	%Q*	BOOL			
6	VAR_OUTPUT	**G5000_A1_X1_Y1_Lampa2**	%Q*	BOOL			
7	VAR_OUTPUT	**G5000_A1_X2_Y2_Lampa1**	%Q*	BOOL			
8	VAR_OUTPUT	**G5000_A1_X2_Y2_Lampa2**	%Q*	BOOL			

Figur 7.2: Resultatet i TwinCAT3

8
Slutsats

I följande avsnitt diskuteras resultatet och projektet som helhet, ur ett etiskt, moraliskt såväl som hållbart perspektiv. Utöver detta diskuteras de samlade erfarenheterna projektet bringat. Vilka för- och nackdelar har lösningen som presenterats och vilka förbättringsmöjligheter finns?

8.1 Diskussion

En omfattande del av projektet handlade om VBA programmering. Ett språk som aldrig tidigare introducerats under elektroingenjörsutbildningen. Visst finns det likheter mellan VBA och C++ som i sin tur bygger på C, men språket som helhet var nytt för oss.

Den gamla lösningen var som tidigare nämnt svårförstådd. Koden var sparsamt kommenterad och som utomstående var det därmed komplicerat att förstå sig på koden skriven i de olika makrosen. Det förekom variabelnamn som var irrelevanta för vad de beskrev. När vi befann oss i startgroparna för examensarbetet fanns ingen tydlig kravspecifikation för vad som förväntades av det nya verktyget. Konsekvensen av detta var att vi inte kunde visualisera en tydlig målbild för projektet. Men i motsats till detta gav det oss utrymme för egen reflektion över hur vi ansåg att verktyget skulle operera lämpligast.

Vad förväntade sig Acobia av verktyget och vilka funktioner var tillåtna att addera? Verktygets utvecklare arbetade inte längre och därmed diskuterades aldrig verktygets funktionalitet med honom. Många av de diskussioner som förts i samråd med Acobia var längs vägen och därmed omformades slutbilden sent under projektets gång. Till en början var huvudmålet att testa om det gamla verktyget i den gamla programvaran kunde användas på samma sätt i den nya programvaran. Vi konstaterade att det gick. Vidare övergick målbilden till att bredda verktygets användningsområde, men adressändring i TwinCAT3 genom Excelbladet genomfördes aldrig. Att försöka få adressändringen att fungera önskvärt är ingen omöjlig uppgift, men hade gjort projektet för stort i förhållande till den återstående tiden. Att lösa problematiken kring adressändringen skulle fördelaktigt kunna ligga till grund för ett nytt examenarbete hos Acobia.

I rollen som anställd på ett företag med uppdrag att ansvara för programmering, installation och underhåll av ett automatiserat fastighetsystem är målbilden att systemet ska vara lätthanterligt, tydligt och överskådligt. Vi har lyckats skapa ett verktyg med hjälp av Excel och VBA som effektiviserar Acobias arbetsätt vid större automationsprojekt för fastigheter. Strukturen i det gamla Excelarket kvarstår, men har fått ett väldisponerat utseende, dessutom har vi visat att det är möjligt att knyta namnen direkt till blockens in-och utgångar vilket inte verktygets föregångare gjorde.

Fördelarna med det verktyg som erhållits ur projektet är främst dess användarvänlighet. Ett lätthanterligt verktyg som förenklar automatiserad kodgenerering. Verktyget som skapats är fullt fungerande. VBA koden utförs i flera olika steg, och någon med mer erfarenhet kring VBA hade med säkerhet kunnat utföra samma instruktioner i en mer kortfattad kod.

8.2 Etik & moral

Det råder få tvivel om att framtidens arbetsmarknad kommer att se annorlunda ut, av samtiden att dömma. En av anledningarna är samhällets ständiga törst på effektivisering. På uppdrag av Stiftelsen för Strategisk Forskning, SSF, undersökte Stefan Fölster, nationalekonom och professor vid Kungliga Tekniska Högskolan hur jobben i Sverige kommer att påverkas av automatiseringens framfart[17].

Vilka konsekvenser medför egentligen digitaliseringen? Vad kommer hända om vartannat av Sveriges jobb automatiseras inom 20 år som Fölster hävdar? Maskiner behöver inte längre samma handpåläggning som tidigare och företag kan därmed delvis eller fullständigt automatisera sina processer. För de anställda vars jobb blivit ersatta av teknik kan varsel vara påföljden. Å andra sidan kan automatisering bidra till nya jobb på annat håll. Dock är dessa jobb ofta av mer komplex karaktär och kräver högre utbildning eller mer troligt, en helt annan utbildning än för det jobb som byttes ut. Ett exempel på sådana nya jobb är installatör, tekniker och utvecklare av det automatiserade systemet [18].

Som tidigare nämnt har automatiserade processer en stor mängd fördelar och de verkar överallt omkring oss, ofta i det dolda. Så gör även den PLC som programmerats under detta projekt. En PLC programmerad för att styra ventilation, ljus och temperatur skapar förutsättning att annat arbete kan utföras på det mest fördelaktiga sätt i de fastigheter där den är lokaliserad. På ett sjukhusområde, med många byggnader och flertalet våningsplan ska det inte behöva läggas onödiga resurser på att en operationssal är korrekt belyst eller att temperaturen är behaglig för patienter på uppvaket.

8.3 Hållbarhet

Det första av de fem svenska energimålen för år 2020 lyder som följer. "Energianvändningen ska till 2020 vara 20 procent effektivare jämfört med 2008". I dagsläget distribueras 146 TWh till bostads- och servicesektorn vilket gör den till den del av samhället där energiåtgången är som störst. Tätt därefter kommer industrisektorn med 143 TWh/år [19].

Ett steg i rätt riktning mot minskad energiförbrukning är automation. Inom industrisektorn bidrar automation till jämna flöden där robotar är programmerade att utföra specifika arbeten, varken mer eller mindre. Konsekvensen av en skräddarsydd lösning är mindre slitage på robotar och dess ingående komponenter och därmed också en mindre miljöpåverkan under reparation eller testning av utbytt utrustning.

Acobia AB arbetar med att skapa smarta lösningar både inom industri- och fastighetssektorn. Projektet som gjorts tillsammans med Acobia AB hade effektiv fastighetsstryning som slutdestination. Som en del i projektet skapades ett system som skötte belysning i badrum. Genom att nyttja sensorer som reagerar på rörelse, behöver en lampa bara lysa så länge som det är nödvändigt. Genom att låta temperaturstyrningen i en fastighet automatiseras, elimineras snabba energikrävande manuella höjningar eller sänkningar vid utomhusliga temperaturförändringar. Som ett bättre alternativ regleras inomhustemperaturen kontinuerligt efter utomhustemperaturen. Automatiserade vattenkranar och tvålpumpar, med inbyggda rörelsedetektorer, på allmänna toaletter minskar risken för spridning av mikroorganismer från smittkälla till frisk individ, vilket på sjukhus är av största vikt.

Litteraturförteckning

[1] Beckhoff Automation AB, "Twincat 3 | extended automation (xa)." 2020. [Online]. Tillgänglig: https://www.beckhoff.com/english.asp?twincat/twincat-3-extended-automation-engineering.htm, hämtad: 07.05.2020.

[2] B. Thomas, *Modern reglerteknik*. Liber, 2011.

[3] Beckhoff Automation AB, "Real-time ethernet: Ultra high-speed right up to the terminal." 2020. [Online]. Tillgänglig: https://www.beckhoff.com/english.asp?ethercat/default.htm?id=23563557, hämtad: 24.02.2020.

[4] MS. Goodman, "Data communications," i *AccessScience*. [Online]. Tillgänglig: https://www-accessscience-com.proxy.lib.chalmers.se/content/data-communications/180900#, hämtad: 28.02.2020. DOI: 10.1036/1097-8542.180900.

[5] K. Langlois, T. van der Hoeven, D. R. Cianca, T. Verstraten, T. Bacek, B. Convens, C. Rodriguez-Guerrero, V. Grosu, D. Lefeber, and B. Vanderborght, "Ethercat tutorial: An introduction for real-time hardware communication on windows [tutorial]," *IEEE Robotics & Automation Magazine*, vol. 25, no. 1, pp. 22–122, 2018. hämtad: 28.03.2020. DOI: 10.1109/MRA.2017.2787224.

[6] W. Bolton, *Programmable Lodgic Controllers*. Elsevier, 2015.

[7] P. Seneviratne, *Building Arduino PLCs: The essential techniques you need to develop Arduino-based PLCs*. Apress, 2017.

[8] Beckhoff Automation AB, "Cx9020 | basic cpu module." 2019.[Online]. Tillgänglig: https://www.beckhoff.com/english.asp?embedded_pc/cx9020.htm , hämtad: 21.11.2019.

[9] Beckhoff Automation AB, "Compact high-performance controller for plc and motion control." 2012. [Online]. Tillgänglig: https://www.beckhoff.com/CX9020/, hämtad: 21.11.2019.

[10] B. Molin, *Analog Elektronik*. Stundetlitteratur, 2009.

[11] L. Bengtsson, *Elektriska mätsystem och mätmetoder*. Studentlitteratur, 2003.

[12] L. Bergström and L. Nordlund, *Ellära: krets-och fältteori*. Liber, 2012.

[13] Sourcegear, "DiffMerge Product Features." 2017. [Online]. Tillgänglig: https://sourcegear.com/diffmerge/, hämtad: 28.02.2020.

[14] A. Agulyansky, *Chemistry of Tantalum and Niobium Fluoride Compounds*. Elsevier Science, 2004.

[15] Electrical technology, "Infrared motion detector circuit – diagram, working applications." 2020. [Online]. Tillgänglig: https://www.electricaltechnology.org/2019/05/infrared-motion-detector-circuit.html, hämtad: 25.03.2020.

[16] W.T.Xie, Y.J.Daia, R.Z.Wanga, K.Sumathy, “Concentrated solar energy applications using Fresnel lenses: A review.” 2011. [Online]. Tillgänglig: https://www.sciencedirect.com/science/article/pii/S1364032111001341?casa_token=jr-7rfo-Dh4AAAAA:EZSvNSsvXVtTJ8MeHRc5-BJVTQJzPVcw3zFrYt5h6CXVEjAQu5JOALy9FEHP3EsnhFq_WPADwA#bib0240, hämtad: 21.04.2020 DOI:10.1016/j.rser.2011.03.031.

[17] S. Fölster, *Vartannat jobb automatiseras inom 20 år–utmaningar för Sverige. Stiftelsen för strategisk forskning, Stockholm*, 2014.

[18] Nicklas Wiborg, “Kommer de digitala processerna ta ditt jobb?.” 2019. [Online]. Tillgänglig: https://relight.se/blogg/kommer-de-digitala-processerna-ta-ditt-jobb, hämtad: 17.11.2019.

[19] Energimyndigheten, *Energiläget 2019 - En översikt*. Arkitektkopia AB, 2019.

references.bib url.bib