

Styrning av Siemens PLC med Android applikation

- En studie om en Android applikation för styrning av Siemens PLC S7-300/1200



Omar Adel
Erol Ali

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

Styrning av Siemens PLC med Android applikation

**– En studie om en Android applikation för styrning
av Siemens PLC S7-300/1200**



**LUNDS
UNIVERSITET**

Lunds Tekniska Högskola

**LTH Ingenjörshögskolan vid Campus Helsingborg
Industriell Elektroteknik och Automation**

Examensarbete:
Omar Adel
Erol Ali

© Copyright Omar Adel, Erol Ali

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
Media-Tryck
Biblioteksdirektionen
Lunds universitet
Lund 2013

Sammanfattning

Detta är ett examensarbete som beskriver processen för att ställa in webbservern i Siemens S7-300 och i S7-1200. Vidare beskrivs hur en applikation i Android utvecklas för att ansluta till PLC och sedan läsa av och skicka information. Det är på begäran av ÅF som Metod kapitlet skrivits som en steg-för-steg manual för att enkelt kunna upprepa samma projekt och få liknande resultat.

För att kunna kommunicera med PLC så har alla steg gjorts i rapporten och mobiltelefonen har kunnat starta och stoppa en motor samtidigt som den läst av dess värden.

Applikationen är avsedd att säljas av ÅF som en tjänst för deras kunder. Ett fast pris kunde inte sättas på applikationen då PLC program varierar i storlek. Varje applikation måste skräddarsys efter ett PLC program. Den integrerade webbservern valdes för att det var det billigaste alternativet för att upprätta en kommunikation mellan mobiltelefon och CPU. I rapporten beskrivs alternativa metoder till webbservern.

Nyckelord: Siemens S7-300/1200, Android, Integrerad Webserver, TIA Portal V11, Step 7, PLC

Abstract

This is a thesis that describes the procedure for setting up the web server on Siemens S7-300 and S7-1200. It also describes how to develop an application on Android to connect to the PLC and then read and send information. It is on the request of ÅF that the Method chapter is written as a step – by – step manual for easy repeat of the project and to get similar results.

To communicate with the PLC, all steps in the report have been made and the mobile phone has been able to start and stop a motor while reading its values.

The application is intended to be sold by ÅF as a service to their customers. A fixed price could not be decided because the PLC program is varying in size. Each application must be tailored to a PLC program. The integrated web server was chosen because it was the cheapest option for establishing the communication between the Smartphone and the CPU. The report describes alternative methods to the web server.

Keywords: Siemens S7-300/1200, Android, Integrated Web server, TIA Portal V11, Step 7, PLC

Förord

Detta examensarbete omfattar 22,5 högskolepoäng och är den avslutande delen för högskoleingenjörsutbildningen på programmet Elektro- och Automationsteknik vid Lunds Tekniska Högskola/Campus Helsingborg. Arbetet är utfört på ÅFs kontor i Helsingborg i samarbete med ÅF Industry och Institutionen för Industriell Elektroteknik och Automation vid Lunds tekniska högskola.

Ett tack riktas mot Peter Björklund som gav oss möjligheten att utföra vårt examensarbete vid ÅFs kontor i Helsingborg. Ett tack till Martin Trygg, vår handledare, och ett tack till hela ÅF kontoret. Slutligen vill vi tacka Mats Lilja, vårexaminator vid Lunds Tekniska Högskola/Campus Helsingborg.

Ordlista

Android	Googles operativsystem för mobila enheter
Applikation	Ett program i mobiltelefonen
AWP	Automation Web Programming
Emulator	Efterliknar en Android telefon
DB	Data Block
FBD	Function Block Diagram
Google Play	Applikations center för Android
IOS	Apple operativsystem för mobila enheter
LAD	Ladder Logic
OB	Organization block
Objective-C	Ett programmeringspråk
Plattform	Maskinvara med ett operativsystem
PHP	Hypertext Preprocessor
P_trig	Pulsblock i Step 7 som genererar pulser
Raspberry PI	Billig dator
Smartphone	Smart mobiltelefon
SOAP	Simple Object Access Protocol
STL	Statement List

Innehållsförteckning

1 Inledning	2
1.1 Bakgrund	2
1.2 Syfte	2
1.2.1 Frågeställningar	3
1.3 Avgränsning	3
2 Verktyg	4
2.1 Mjukvaruprogram	4
2.1.1 Eclipse	4
2.1.2 Android SDK	4
2.1.3 TIA Portal v11	4
2.1.4 SIMATIC Manager	4
2.1.5 Notepad++	5
2.2 Programmeringsspråk	5
2.2.1 HTML.....	5
2.2.2 CSS	6
2.2.3 Java	6
2.2.4 JavaScript	7
2.3 Materiellista	7
3 Metod	8
3.1 S7-1200	8
3.1.1 Konfigurera S7-1200.....	8
3.1.2 Inställningar för webbservern	9
3.1.3 Skapa, kompilera och ladda över program till CPU	9
3.2 S7-300	10
3.2.1 Konfigurera S7-300	10
3.2.2 Konfigurera hårdvaran	11
3.2.3 Konfigurera webbservern	12
3.2.4 Symboler och S7- program	13
3.2.5 Länka samman HTML-sidorna med S7-programmet.....	15
3.3 Android Applikationen	16
3.3.1 Tillvägagångssätt	16
3.4 User defined webpage	17
3.4.1 Generering av HTML-sidor.....	17
3.5 Fördjupning av Inställningar	18
3.5.1 S7-1200	18
3.5.2 S7-300	18
3.5.3 Android	19
3.5.4 HTML.....	19
3.6 Källkritik	20
4 Analys	21

4.1 Förstudie	21
4.2 Alternativa metoder	21
5 Resultat	23
5.1 Webbservern	23
5.2 Applikationen	23
6 Diskussion	25
7 Slutsats	27
7.1 Framtida utvecklingsmöjligheter	27
8 Referenser	28
8.1 Webblänkar	28
8.2 PDF manualer.....	29

1 Inledning

I detta kapitel beskrivs bakgrund, syfte och avgränsning för examensarbetet.

1.1 Bakgrund

Detta examensarbete är skrivet i ÅF som är ett ledande teknikkonsultföretag som erbjuder kvalificerade ingenjörer med lösningar för industrins processer. Uppdragen skiftar mellan olika områden så som från utveckling av mobiltelefoni till projektering. Med bas i Europa och kunder runt om i världen har ÅF cirka 6800 medarbetare [1].

Projektet var på eget initiativ och skulle omfatta en applikation som styr en motor men Peter Björklund, Area Manager i Helsingborgskontoret, omformulerade idén till att istället göra en applikation till en Siemens PLC som styr en motor eller en pump. En sådan applikation kan bli ett användbart verktyg på arbetsplatsen som ger en överblick på vad som är igång eller på vad som står still. Den kan underlätta för ansvariga operatörer att felsöka i en maskin och minska kostnaderna för företagen då applikationen meddelar när en maskin är ur funktion. Operatören kan även starta och stoppa en maskin med telefonen utan att befinna sig i samma rum. Det kan vara nödvändigt om något värde inte stämmer och bör kontrolleras.

Anledningen till att en mobilapplikation valdes vid styrning av PLC var att användning av smarta mobiltelefoner har bland Sveriges invånare ökat från 30 % till 50 % från åren 2011 till 2012 enligt frapps.se [2]. Enligt den statistiken så har hälften av Sveriges invånare en ”smartphone” som kan utnyttjas på arbetsplatsen till mer än bara samtal.

1.2 Syfte

Syftet med denna rapport är att undersöka Siemens möjligheter att kommunicera trådlöst med en Android mobiltelefon. Metoden i denna rapport skall enligt begäran från ÅF Industry kunna användas som en lathund för hur webbservern aktiveras, variabler länkas och hur en Android applikation skapas och ansluter till PLC. Metoden delas in i många delkapitel då det gör det lättnavigerat för en användare som följt stegen att gå tillbaka vid felsökning.

Syftet med detta projekt är inte att skriva PLC kod för en motor som i exemplet som nämns i kapitel 4 Metod. Därför skrivs inte all kod ut för motorexemplet.

1.2.1 Frågeställningar

- Kan en Androidbaserad mobiltelefon kommunicera med Siemens PLC S7-1200?
- Är det möjligt att läsa av värden?
- Tilldela nya värden?
- Skriva ut alarm status?
- Kan samma teknik användas för att kommunicera med Siemens PLC S7-300/400?

1.3 Avgränsning

Applikationen kommer att programmeras i Eclipse med Java för Android då detta är en öppen plattform. Detta för att vi redan är bekanta med utveckling av Android – applikationer.

ÅF ansåg att det var mest intressant att kunna kommunicera med Siemens PLC S7-300/400/1200.

2 Verktyg

Här beskrivs de hjälpmedel som använts för att kunna utföra detta projekt.

All programmering, nedladdning och överförning utfördes från Windows XP som körs på en PC, DELL D820.

2.1 Mjukvaruprogram

2.1.1 Eclipse

Eclipse är en kostnadsfri utvecklingsmiljö som bildades i januari 2004 av Eclipse Stiftelsen som en oberoende icke-vinstdrivande bolag för att fungera som förvaltare av Eclipse samfundet. I Eclipse utvecklas programvara och projekter i flera olika språk såsom Java, C, C++ och Python. Det finns i svenska och är tillgänglig för de flesta plattformar. Det går även att installera olika plug-in för Android och Webbserverar[3].

2.1.2 Android SDK

Android SDK är en verktygslåda som utvecklare använder för att skapa applikationer för Android-plattformen. Den innehåller en emulator och nödvändiga bibliotek för att kunna bygga Android – applikationer. När Android SDK är installerad i Eclipse som ett plug-in är det möjligt att utveckla Javaapplikationer för Android. Android SDK erbjuder även grafiskt användargränssnitt vilket gör det vänligt för utvecklarna att kunna få en överblick av layouten och designen av deras applikation[4].

2.1.3 TIA Portal v11

Totally Integrated Automation Portal är det senaste programmet som Siemens har utvecklat för S7-1200 familjen. Den ersätter föregångaren SIMATIC Manager. I TIA Portal v11 skapas alla projekt och kodas i samma program. Programmet klarar de olika programmeringstyperna, LAD, STL och FBD. Allt är integrerat i TIA Portal v11, en användbar funktion är ”Go Online” där får användaren en översikt av programmet. Ingångarna lyser grönt när de är till och rött när de är från. Detta ger en möjlighet för användaren att simulera programmet innan den laddas ner i PLC.

2.1.4 SIMATIC Manager

SIMATIC Manager är ett program som Siemens automation har utvecklat för konfigurering av inställningar för alla SIMATIC produkter. I Step 7 så kodas programmet som sedan skall köras och förs över till Siemens PLC S7-300/400. SIMATIC Manager låter programmeraren koda sitt projekt, konfigurera hårdvaran, korrigera symboler och slutligen ladda ned projektet

till PLC. I SIMATIC Manager finns det en funktion som gör det möjligt att testa sin kod genom att gå ”online” och få en översikt av exekvering av koden i runtime.

2.1.5 Notepad++

Notepad++ är ett konstadsfritt program som är framställt för Microsoft Windows operativsystem. Det är ett textredigeringsprogram för kodning av HTML, PHP, JavaScript samt en del andra språk. Notepad++ är kodad i C++ och är mycket kraftfull vid kompilering av projekt. Utvecklarna bakom programmet försöker använda så lite så möjligt av CPU kraften vid kompilering och därmed resulterar det hela till stabilare och snabbare program [5].

2.2 Programmeringsspråk

Här beskrivs de programmeringsspråk som användes för kommunikation och design.

2.2.1 HTML

”Hyper Text Markup Language” eller den kända förkortningen HTML är koden som används vid skapandet av webbsidor. Med HTML kan en hemsida kodas och visas på en webbläsare. Den första versionen av HTML lanserades redan 1991 och sedan dess har det släppts 7 versioner där den senaste är XHTML5 (år 2013). Syntaxen för HTML är följande:

```
<!DOCTYPE html>
<html>
<head>
</head>

<body>
<h1> Rubrik </h1>
<p> Paragraf, med andra ord ren text. </p>

</body>
</html>
```

HTML är ett märkspråk, det vill säga att det kodas med hjälp av vinkelparenteser. En instruktion uppkommer normalt i par till exempel ”<p> text </p>”. Den börjar med vinkelparenteser och avslutas med snedstreck för att tala om för webbläsaren var en instruktion börjar och slutar. För att en hemsida skall visas korrekt på webbläsaren så måste webbläsaren veta

webbsidans dokumenttyp och vilken version den har. Det görs med följande deklARATION: "`<!DOCTYPE html>`" (se syntaxen ovan). Eftersom där finns det många olika dokument på webben så är denna deklARATION nödvändig.

Webbläsarna visar inte vinkelparenteserna utan bara det som står mellan de. Däremot är vinkelparenteserna avgörande för webbläsaren då de talar om för webbläsaren hur innehållet skall visas.

Mellan "`<head></head>`" taggarna skrivs all nödvändig information för webbläsaren. Det är där JavaScripts eller CSS filer inkluderas, hemsidans titel samt en del annan information som har med funktionalitet att göra. Det som skrivs mellan "`<body></body>`" taggarna visas på webbläsaren. Det är där all information läggs in för att kunna framföra det på webbläsaren. Det går att använda flera HTML sidor och länka de med varandra. Att endast använda HTML kommer inte vara effektivt då HTML sidor är statiska. De kan inte utföra avancerade beräkningar eller något liknande. HTML sidor visar endast innehåll på en sida, däremot kan PHP eller JavaScript användas för att kunna omvandla HTML sidan till en dynamisk sida. Med hjälp av PHP eller JavaScript kan vanliga uttryck såsom "for", "while", "if" och "switch-case" utnyttjas [6].

2.2.2 CSS

W3C skapade Cascading Style Sheets, CSS, för att underlätta för utvecklare att designa webbsidor. CSS – filen är en separat fil där bl.a. storlek och färg bestäms på webbsidan. CSS – filen inkluderas inom "`<head></head>`" i HTML – filen såsom nedan.

```
<link rel="stylesheet" type="text/css" href="mystyle.css">
```

För mindre webbsidor med lätt design kan koden för design och layout göras direkt inom "`<head></head>`" och ingen separat CSS – fil behövs. [7].

2.2.3 Java

Java är ett objektorienterat programmeringsspråk som introducerades 23 maj 1995 av grundaren James Gosling på Sun Microsystems. Java finns från superdatorer för forskningsändamål till effektiva mobil applikationer. Detta språk är plattformsoberoende på grund av att Java kompilerar sin kod till bytekod istället för maskinkod. Den färdig kompilerade bytekoden kan i efterhand köras på en plattformsoberoende maskin.

Mer än 9 miljoner utvecklare använder idag språket Java vilket gör att Java är testat och utvecklat för att uppfylla utvecklarens förväntningar och mål [8].

2.2.4 JavaScript

JavaScript är ett av världens mest populära programmeringsspråk för HTML, webb applikationer, datorer, mobiltelefoner och för serverar. JavaScript omvandlar en HTML sidan från att vara en statisk till dynamisk med avancerade funktioner. JavaScript är snarlikt Java och C++. Det är möjligt att använda samma uttryck som i Java och C++; ”for”, ”while”, ”if” och ”switch case”. Med dessa funktioner tillgängliga underlättas HTML kodningen och det funktionaliteten ökar för en hemsida som utnyttjar JavaScript. För att kunna använda JavaScript i HTML krävs att det anropas på rätt sätt, det vill säga att ”<script> JavaScripts kodning </script>” måste skrivas inom ”<head>” eller ”<body>” delen. För tydlighetens skull så brukar de flesta programmerare ha JavaScripten i en särskilt fil och anropar den i ”<head>” som följande syntax: “<script src= ”filNamnet.js” type= ”text/javascript”></script> ”

Men för mindre webbsidor är det vanligt att JavaScripten utnyttjas direkt i ”<body>” eller ”<head>” delen. Alla dagens webbläsare klarar av att köra JavaScript och de är anpassade till det [9].

2.3 Materiellista

- DELL D820
- DELL D830
- Ethernet – kabel
- Frekvensomvandlare
- Motor
- Dlink Router
- Siemens S7-300 CPU 314C-2 PN/DP
- Siemens S7-1200 CPU 1212C AC/DC/RLY
- Plintar
- Kopplingskablar
- Spänningsomvandlare

3 Metod

Detta kapitel beskriver tillvägagångssättet för att uppnå de resultat som beskrivs i kapitel 5 [Resultat](#). För att kommunicera med både S7-1200 och S7-300 användes dess inbyggda webbservrar. Nedan följer information om hur webbservern aktiveras och hur en webbsida för CPU skapas och används som en Android applikation. I denna rapport används CPU 1212C AC/DC/RLY och S7-314C-2 PN/DP med programkod för att styra en motor. Alternativa metoder för denna PLC beskrivs i kapitel [4.2](#).

Stegen som anges nedan sker i den ordning de presenteras.

3.1 S7-1200

TIA Portal V11 ([2.1.3](#)) användes för att ställa in webbservern för CPU S7-1200.

Detta kapitel kan användas som en lathund till hur webbservern aktiveras, för djupare förståelse av utvalda punkter hänvisas läsare till kapitel [3.5](#)[16].

3.1.1 Konfigurera S7-1200

- PLC och datorn kopplades samman med en router och två Ethernet kablar.
- Ett nytt projekt skapades med TIA Portal V11 SP2.
- För att kunna skriva programkod valdes rätt CPU via ”Add new device”. Vid val av ”unspecified” så väljs rätt CPU automatiskt av programmet.
- IP adress och subnät för CPU ställdes in genom inställningar i CPU under projektet. Där las ett nytt subnät in i ”Ethernet addresses” menyn. Den IP adress som tilldelades CPU är samma adress som används vid anslutning till webbsidan.
- Variabelnamn tilldelades för att binda samman namn och logiska operatorer samt för att underlätta felsökning då de är kommenterade.

I ”PLC tags” > ”Default tag table” korrigeras dessa som i figur 1.

Default tag table						
	Name	Data type	Address	Retain	Visibl...	Acces...
1	Start	Bool	%I0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
2	Stop	Bool	%I0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
3	Reset_Knapp	Bool	%I0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
4	Motorskydd	Bool	%I0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
5	Motor1	Bool	%Q0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
6	Tag_1	Bool	%Q0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
7	AWP	DInt	%MD2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
8	AWPStart	Bool	%M100.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
9	AWPStop	Bool	%M100.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
10	Tag_2	Byte	%MB1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
11	Tag_3	Bool	%M1.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
12	Alarm_minne_ok	Bool	%M150.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
13	FirstScan	Bool	%M1.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
14	AlwaysTRUE	Bool	%M1.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
15	AlwaysFALSE	Bool	%M1.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
16	Clock_Byte	Byte	%MB0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
17	Clock_10Hz	Bool	%M0.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
18	Clock_5Hz	Bool	%M0.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
19	Clock_2.5Hz	Bool	%M0.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
20	Clock_2Hz	Bool	%M0.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
21	Clock_1.25Hz	Bool	%M0.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
22	Clock_1Hz	Bool	%M0.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
23	Clock_0.625Hz	Bool	%M0.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
24	Clock_0.5Hz	Bool	%M0.7	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
25	Tag_4	Bool	%M170.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
26	Tag_5	Bool	%M170.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
27	Tag_6	Bool	%M179.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
28	AWPMotorskydd	Bool	%M179.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
29	AWPMemory	Bool	%M15.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
30	AWPMotor1	Bool	%M16.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
31	AWPReset_Knapp	Bool	%M17.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
32	Tag_7	Bool	%M180.1	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
33	Tag_8	Bool	%M180.3	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
34	Tag_9	Bool	%M180.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
35	Start_remote	Bool	%M180.4	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
36	Stop_remote	Bool	%M180.5	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
37	Tag_10	Bool	%M181.0	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
38	Tag_11	Bool	%M181.2	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
39	Tag_12	Bool	%M180.6	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

Figur 1. Tabell för variabelnamn i TIA Portal V11

- Variabelnamnen i figur 1 användes av HTML sidorna. Hur HTML sidorna skapas beskrivs i kapitel [3.4.1](#).

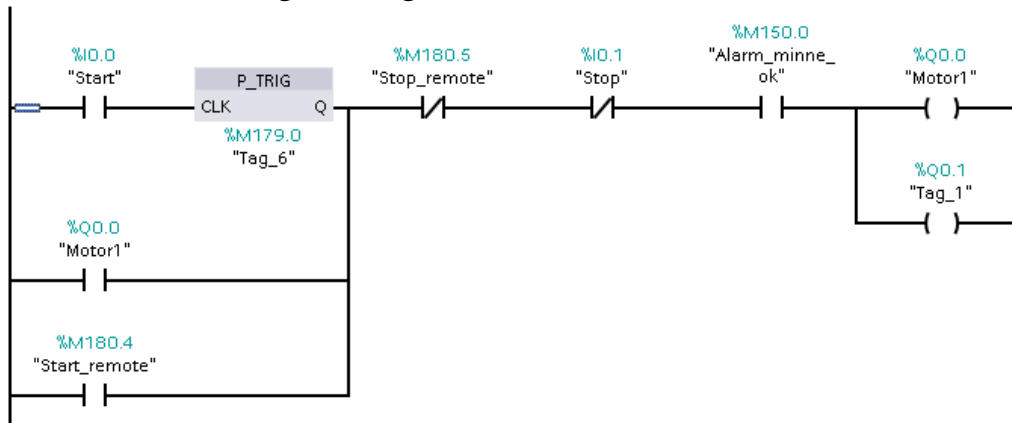
3.1.2 Inställningar för webbservern

CPU > ”Properties” > ”Web server”

- Webbservern och automatisk uppdatering aktiverades.
- I vilken katalog HTML filerna fanns angavs samt valdes ”index.html” som startsida. Projektet får sitt namn Motor via ”Application name”.
- Slutligen genererades ”Web Control DB” ([3.5.1](#)) genom ”Generate blocks”.
- I CPU > ”Properties” > ”Protection” sattes ett skrivskydd med användarnamn och lösenord.

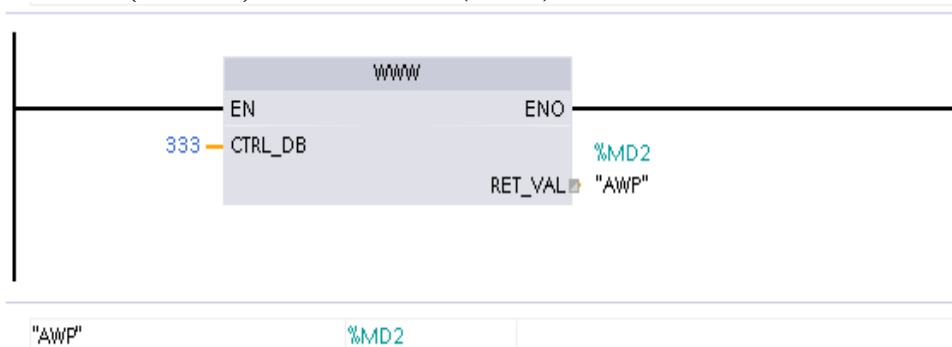
3.1.3 Skapa, kompilera och ladda över program till CPU

- Under ”Program blocks” i CPU skrevs en del av PLC koden för motorn i Ladder som framgår av figur 2.



Figur 2. Programkod för motor (OB1) i TIA Portal V11

- ”WWW” (SFC99) sattes i OB1 ([3.5.1](#)).



Figur 3. ”WWW” instruktion i TIA Portal V11

- När programmet var skrivet kompilerades den och laddades ner till CPU genom snabbkommandona Ctrl+B (kompilera) och Ctrl+L (ladda ner till CPU).

3.2 S7-300

Step 7 och SIMATIC Manager är plattformarna för S7-314C-2 PN/DP vilken användes för att implementera programmet[17].

Detta kapitel kan användas som en lathund till hur webbservern aktiveras, för djupare förståelse på utvalda punkter hänvisas läsare till kapitel [3.5](#).

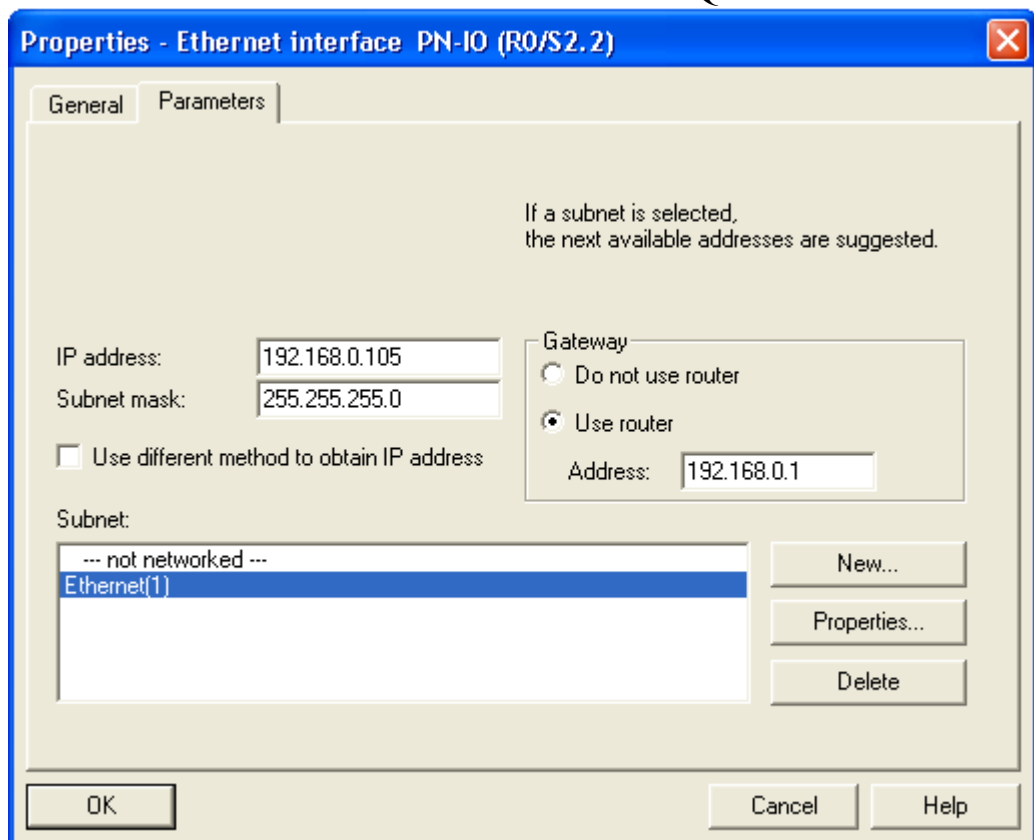
3.2.1 Konfigurera S7-300

- PLC kopplades till datorn med en Ethernet kabel och ett nytt projekt skapades med SIMATIC Manager. Projektnamnet sattes till ”webmotor” via ”File” > ”New”.

- En ny station tillades via "Insert" > "Station" > "SIMATIC 300 Station".
- För att konfigurera hårdvaran på CPU dubbelklickades "SIMATIC 300" och därefter dubbelklickades "Hardware".
- Under "Options" > "Set PG/PC Interface" valdes "TCP/IP wireless" för trådlös anslutning.

3.2.2 Konfigurera hårdvaran

- På högra sidomenyn öppnades "SIMATIC 300", "RACK-300" och "Rail" dubbelklickades.
- Ett nätaggregat valdes i "PS-300" > "PS 307 2A".
- Rätt CPU(CPU314C-2 PN/DP) valdes under samma meny. "CPU-300" > "CPU314C-2 PN/DP" > "6ES7 314-6EH04-QAB0" > "V3.3".

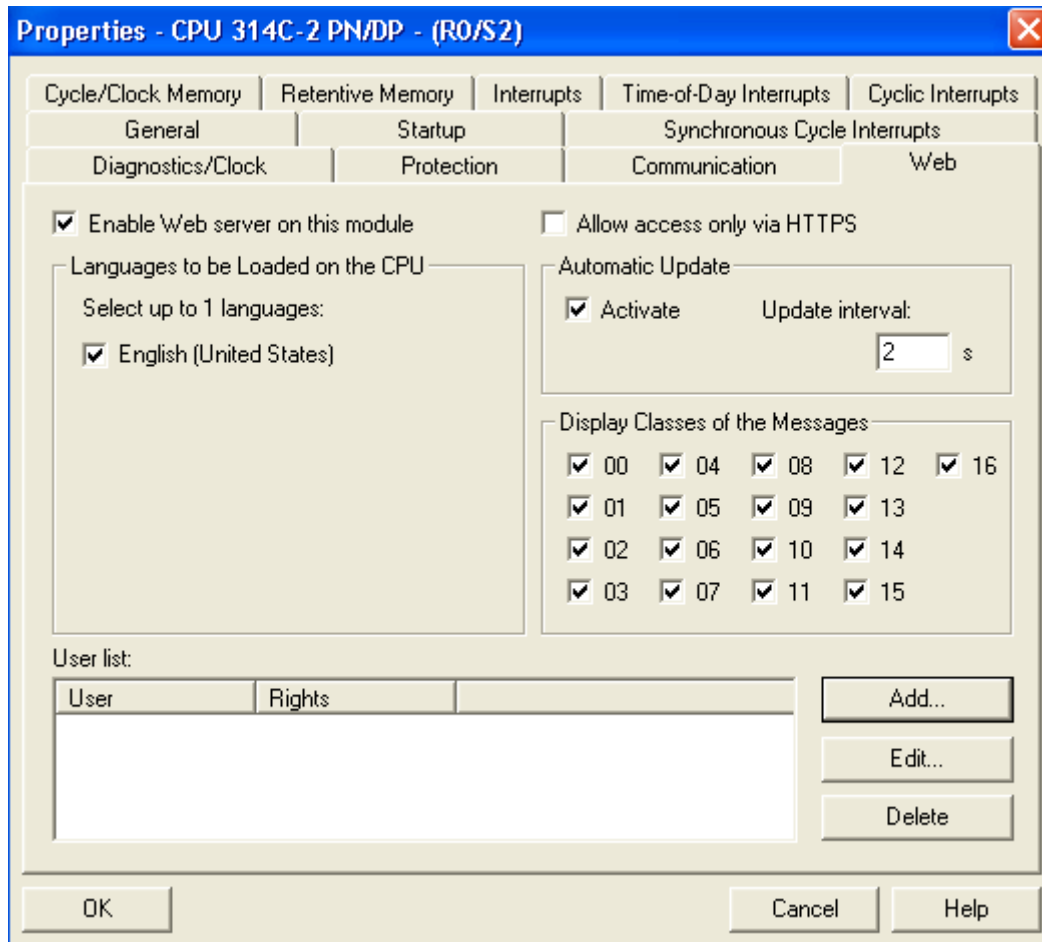


Figur 4. SIMATIC Manager > Hardware configuration > Ethernet interface

- Som figur 4 visar så bockades "Use router" för och ett nytt Subnät lades till via "New" knappen. Därefter bekräftades alla dialogrutor med "OK"

3.2.3 Konfigurera webbservern

- För att aktivera webbservern, automatisk uppdatering och i vilket språk webbservern ska visas i så dubbelklickades det på CPU och sen valdes webbfliken där dessa bockades för.



Figur 5. Webbfliken i CPU inställningar

- I webbfliken las en ny användare till med läs- och skrivrättigheter till CPU via webbservern med ”Add”-knappen.

Figur 6. Lägg till användare

- När inställningarna ställts in såsparades och kompilerades de i CPU.

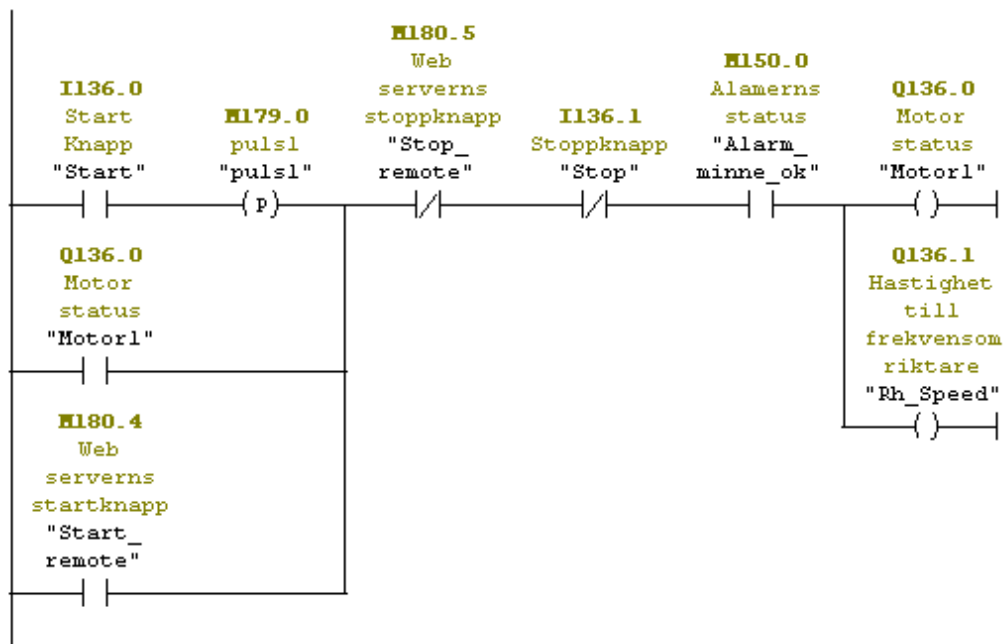
3.2.4 Symboler och S7- program

- I ”Symbols” under ”S7-Program” finns variabelnamnenligt figur 7.

	Status	Symbol	Address	Data type	Comment
1		Alarm_minne_ok	M 150.0	BOOL	Alamerns status
2		AWP_web	FC 1	FC 1	
3		AWPMemory	M 115.0	BOOL	
4		AWPMotor1	M 16.0	BOOL	
5		AWPMotorskydd	M 179.1	BOOL	
6		AWPReset_Knapp	M 117.0	BOOL	Web serverns resetknapp
7		AWPStart	M 100.0	BOOL	
8		AWPStop	M 100.1	BOOL	
9		Motor1	Q 136.0	BOOL	Motor status
10		Motorskydd	I 136.2	BOOL	
11		puls1	M 179.0	BOOL	puls1
12		puls2	M 180.6	BOOL	
13		puls3	M 181.0	BOOL	
14		puls4	M 181.2	BOOL	
15		Reset_knapp	I 136.3	BOOL	
16		Rh_Speed	Q 136.1	BOOL	Hastighet till frekvensomriktare
17		Start	I 136.0	BOOL	Start Knapp
18		Start_remote	M 180.4	BOOL	Web serverns startknapp
19		Stop	I 136.1	BOOL	Stoppknapp
20		Stop_remote	M 180.5	BOOL	Web serverns stoppknapp
21		tag_4	M 170.0	BOOL	
22		Tag_5	M 170.1	BOOL	
23		WEBControl_DB	DB 333	DB 333	
24		WebRequest	SFC 99	SFC 99	
25					

Figur 7. Tabell för variabelnamn i SIMATIC Manager

- Under ”S7-Program” > ”Blocks” lades ett block, OB1, med Ladder till. En del av koden framgår av figur 8.



Figur8.S7-300 SIMATIC Manager. En del av koden i OB1

- Två nyablock lades till, i ladder respektive STL. ”Blocks” > ”Insert new object” > ”function”. I den enadrogs ett SFC99 block ut ur standard biblioteket i vänstra sidomenyn och det blocket fick namnet SFC99. Det andra kallades FC1 och figur 9 visar dess kod.

```

Comment:

CALL "WebRequest"          SFC99
  CTRL_DB:="WEBControl_DB"  DB333
  RET_VAL:=#SFCState_333    #SFCState_333
NOP 0

Network 2: Title:
Comment:

L   #SFCState_333          #SFCState_333
AW  W#16#8010

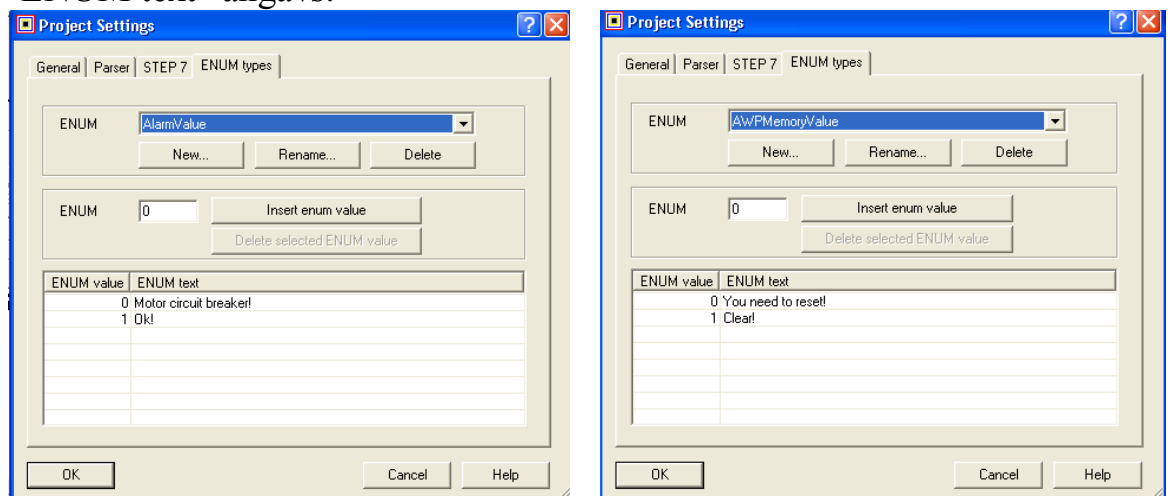
BE

```

Figur 9. SIMATIC Manager Block FC1

3.2.5 Länka samman HTML-sidorna med S7-programmet

- För att binda HTML-sidorna med programmet användes ett tillägg som kallas "S7-Web2PLC". Den fanns under "Block folder" > "S7-Web2PLC". "S7-Web2PLC" krävde separat installation, se kapitel [4.5.2](#)
- Ett nytt projekt skapades genom "File" > "New Project" och döptes till "Example".
- Variabelnamnen i figur 7 exporterades med "Export symbols".
- Den specifika mappen där HTML – sidorna sparades angavs i "File" > "change project settings" under general fliken. Där angavs också vilken HTML-fil som angavs som startsida. Applikationsnamnet sattes till "ex".
- Alarmvärdena sattes i Enum fliken. Två enum värden skapades genom "New" > "AlarmValue" och "AWPMemoryVaule". Alarmvärdena har två tillstånd var. De lades till genom "Insert enum value" där även "ENUM text" angavs.



Figur 10. S7-Web2PLC Enum typer

- DB333 och DB334 genererades genom "Generate DB Source". Därefter kompilerades de och laddades ned till CPU.
- Genom att markera "SIMATIC 300(1)" och sedan klicka på "Download" symbolen så laddades hela projektet ned i CPU.

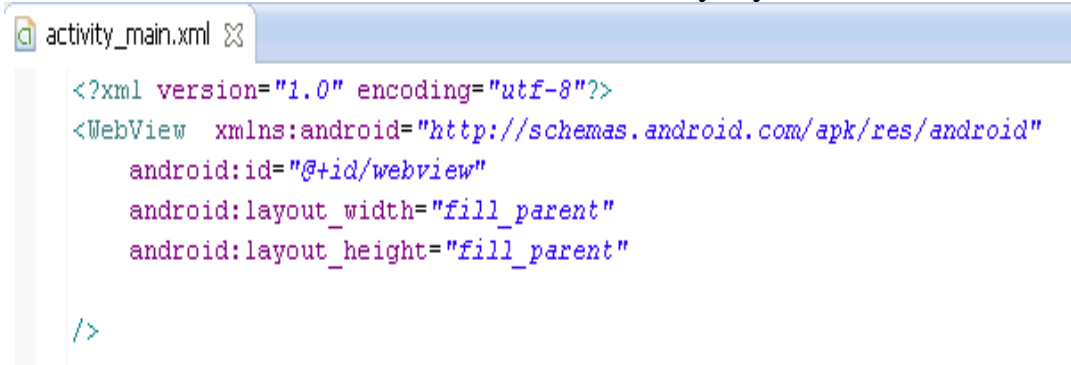
3.3 Android Applikationen

Vid utveckling av applikationer för Androidtelefoner användes följande program; Eclipse med ADT plugin, Android SDK Tools och Android Platform-tools [4]. Dessa fanns som ett färdigt paket på Androids egna hemsida och var gratisatt ladda ner.

Detta kapitel kan användas som en lathund till hur en applikation skapas för anslutning till webbservern som nämns i kapitel [3.1](#) och [3.2](#). För djupare förståelse hänvisas läsare till kapitel [3.5](#).

3.3.1 Tillvägagångssätt

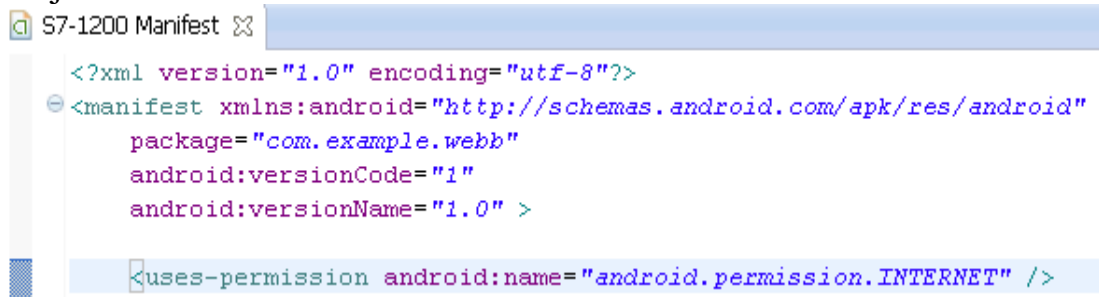
- Via "File" > "New" > "Android Application Project" skapades ett nytt projekt.
- Ett element av webview deklarerades i "activity layout".

A screenshot of the Eclipse IDE showing the XML code for activity_main.xml. The code defines a WebView element with attributes for id, layout_width, and layout_height, all set to fill_parent.

```
activity_main.xml ✕  
<?xml version="1.0" encoding="utf-8"?>  
<WebView xmlns:android="http://schemas.android.com/apk/res/android"  
    android:id="@+id/webview"  
    android:layout_width="fill_parent"  
    android:layout_height="fill_parent"  
>  
</>
```

Figur11. Activity_main.xml Eclipse

- Ett webView element skapades i "MainActivity.java". Med funktionen `loadUrl("http://...")` så laddades den önskade HTML-sidan och med `setJavaScriptEnabled(true)` så kunde HTML - sidan som webView ansluts till använda javaScript.
- Projektet fick tillåtelse att använda internet i AndroidManifest.xml.

A screenshot of the Eclipse IDE showing the XML code for AndroidManifest.xml. The code declares the package name, version code, and version name, and includes a permission for internet access.

```
S7-1200 Manifest ✕  
<?xml version="1.0" encoding="utf-8"?>  
<manifest xmlns:android="http://schemas.android.com/apk/res/android"  
    package="com.example.webb"  
    android:versionCode="1"  
    android:versionName="1.0" >  
  
    <uses-permission android:name="android.permission.INTERNET" />
```

Figur 12. AndroidManifest.xml internetaccess med "android.permission.INTERNET"

- En ny klass MyWebViewClient som är en subclass till WebViewClient skapas och anropas "MainActivity.java".
- En emulator skapades i "Window" > "Android Virtual Device Manager". Med "New" knappen lades en ny emulator till.
- Applikationen testades genom att högerklicka på projektet, "Run as" > "Android Application". Då kompilerades projektet och emulatorn startade.

3.4 User defined webpage

För detta projekt skapades tre stycken HTML-sidor i Notepad++ för styrning av den programmerade motorn i PLC. Sidorna som kodades döptes till index.html, loginPage.html och controlPage.html.

Kodexemplen är markerade i kursiv stil där "taggar" är fet markerade.

3.4.1 Generering av HTML-sidor

- Via "Arkiv" > "Ny" skapades och sparades en ny sida som en HTML fil. Tre stycken filer gjordes på samma sätt.
- För att kunna komma åt PLC "tags" deklarerades följande rader ovanför <html>"taggen":


```
<!-- AWP_In_Variable Name="AWPStart" -->
<!-- AWP_In_Variable Name="AWPStop" -->
<!-- AWP_In_Variable Name="AWPMemory" -->
<!-- AWP_In_Variable Name="AWPMotor1" -->
<!-- AWP_In_Variable Name="AWPReset_Knapp" -->
<!-- AWP_In_Variable Name="AWPMotorskydd" -->
```
- Med hjälp av "AWP commands" kunde önskat "tag" skrivas ut på en specifik plats i HTML sidan. Ett exempel på det är:


```
<p> Motor value: </p><p>:="Motor1":</p>
```

Via ett formulär i HTML kunde en knapp kodas som utförde önskad funktionalitet. Nedan följer ett utdrag från controlPage.html där Start-knappens kodning framgår:

```
<form method="post" action="">
<input value="Start" type="submit" style=" width: 100%">
<input name="AWPStart" value="1" type="hidden">
```

```
<input name=""AWPStop"" value="0" type="hidden">
<input name=""AWPReset_Knapp"" value="0" type="hidden">
</form>
```

Alla HTML filerna, CSS filen samt bilderna sparas i en och samma mapp.

3.5 Fördjupning av Inställningar

Detta kapitel förklarar vad särskilda delmoment i tillvägagångssättet har för funktion. Här förklaras mer detaljerat varför utvalda punkter utförs.

3.5.1 S7-1200

Vid generering av "Web control DB" skapas DB blocken och där lagras all styrinformation från HTML sidorna. Därefter anropas WWW instruktionen med DB blocken som inparameter. DB blocken kan hittas i programblock mappen i projekts navigationsträd där den kan modifieras [16].

För att "User defined web pages" skall vara tillgängliga krävs en "WWW" instruktion. WWW instruktionen måste därför inkluderas och exekveras i OB1 som alltid anropas. Varje gång det blir en ändring i webbservern så ändras ett DB block vilket medför att i någon utav DB blocken då kommer den ändrade blocket att vara en inparameter till "WWW" instruktion och ändrar DB värde [10].

Webbservern kan endast tilldela värden till minnesceller i programkoden. "AWPStart" och "AWPStop" är därför minnesceller som får sina värden från webbservern. Dessa kodas seriellt med "Start_remote" respektive "Stop_remote" med en pulsgenerator. I figur 8 kodas "Start_remote" parallellt med "IO.0Start" för att motorn skulle kunna startas både från webbservern och manuellt. "Stop_remote" kodas seriellt med "IO.1Stop" för motorn ska kunna stoppas antingen från webbservern eller manuellt.

3.5.2 S7-300

CPU tilldelas en IP adress från routern som har det subnätet som läggs till med "New" knappen. För att det skall vara möjligt att komma åt webbservern så måste enheten som vill kommunicera ligga under samma subnät. I figur 4 är subnätet 255.255.255.0.

Variabelnamn tilldelas för att binda samman namn och logiska operatorer samt för att underlätta felsökning då de är kommenterade. Tabellen ger en överskådlig blick över programmet vilket gör det lättare för utomstående att förstå koden.

För att kunna skapa "User Defined Webpage" krävs en separat installation av "S7-Web2PLC" som ligger under STEP 7 DVD V5.5 i CD 2 under "Optional Components". Paketet är integrerat med CPU och gör det möjligt att binda samman HTML sidorna med variablerna i CPU . Den skapar DB block som innehåller koden från HTML – sidorna. Med SFC 99 synkroniseras webbapplikationen med CPU. Ett "Web control DB" innehåller all väsentlig information om webbapplikationen. Det maximala antalet DB block är 16 inklusive "Web control DB" [11].

3.5.3 Android

WebView är en klass i Android SDK som visar en vy av webbsidor. Klassen har till grund att visa en webbsidas innehåll i en aktivitet. Den använder Androids egna WebKit för att visa webbsidor och där finns funktioner som gör det möjligt att anpassa sidan till mobiltelefonen. Användaren upplever inte applikationen som en webbläsare då inga adresser eller laddningsfält förekommer [12].

loadURL("http://...") är en funktion som anropas så fort applikationen startar vilket startar upp den URL sidan som står skriven mellan citattecknen.

3.5.4 HTML

"Meta tags" används för att anpassa HTML sidorna till skärmen på mobiltelefon. Koden nedan användes för att anpassa sidan till mindre skärmar.

```
<meta name="HandheldFriendly" content="true" >  
<meta name="format-detection" content="telephone=no" />  
<meta name="viewport" content="user-scalable=no, initial-scale=1,  
maximum-scale=1, minimum-scale=1, width=device-width, height=device-  
height, target-densitydpi=device-dpi" />
```

"Automation Web Programming", AWP, har hand om syntaxen mellan HTML sidorna och webbservern. Det är Siemens egna programmerade språk som binder CPU "taggarna" med webbsidorna. Inom AWP finns det olika funktioner som används för kommunikation. Läsa, skriva, och sätta enum värden är tre exempel. "AWP commands" kan placeras var som helst i HTML dokumentet men för tydlighetens skull så placeras de i början av HTML sidan. Det är viktigt att tänka på att det är "tag" namnet som används vid anrop och inte adressen.

Ett exempel på en AWP ”command” som ska läsas kan deklarerars på följande sätt:

```
<!-- AWP_In_Variable Name="AWPMotor1" -->
```

3.6 Källkritik

All information som använts till PLC inställningar och funktioner finns att tillgå på Siemens hemsida. Information för mobilapplikationen hämtades från Androids officiella hemsida. Det medför att denna rapport är trovärdig.

4 Analys

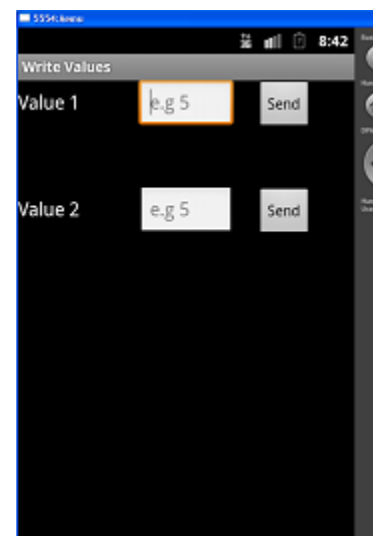
I detta kapitel beskrivs förstudier och lösningsförslag till att kommunicera med S7-300/1200. Den metod som valdes beskrivs i kapitel 3 [Metod](#).

4.1 Förstudie

En metod för att styra S7-1200 trådlöst via en Ipad hittades i vilken Siemens utnyttjar deras egen webserver. Därmed valdes den metoden när kommunikationen mellan S7-1200 och Android upprättades. Manualer i pdf-format fanns på Siemens hemsida att tillgå för att aktivera webbservern.

Det var inte lika tydligt för oss att S7-300 hade en integrerad webserver som kunde användas på samma sätt som på S7-1200. Efter att ha pratat med Siemens Sverige så blev det inte tydligare och då fick andra metoder undersökas för tillämpning. Sökning på nätet gav resultatet att med ett OPC program som översätter data så kan en dator användas som mellanled mellan PLC och mobil-telefon. Hur billig PC skulle kunna användas? Den billigaste vi prövade var Linuxbaserade Raspberry PI. Det gick att skapa en databas i den billiga PC som Android kunde skicka

värden till men det fanns inga OPC program som funkade för Linux. En vanlig Windows PC valdes och ett stort antal OPC program valdes bort på grund av högt licenspris.



Figur13. Prototyp i emulatorn på Android applikation till S7-300 som skickar värden till en server

4.2 Alternativa metoder

ÅF Industry har licens till WinCC Flexible vilket var anledningen till varför detta program var intressant för oss. Den innehåller bl. a. en funktion som kallas "Sm@rtAccess" vilket möjliggör för en extern enhet att komma åt "tags" i en HMI-panel via Ethernet. Licensnyckeln för denna funktion var det billigaste alternativet då ÅF redan har licensen för själva programmet. Kommunikationen i denna metod sker över SOAP[18].

Matrikon OPC "Siemens S7 PLC OPC Server for Siemens PLCs" möjliggör Ethernet kommunikation för lättare åtkomst för PLC data på distans. Denna OPC server är Windows baserad program vilket gör att man måste köra via en

PC med Windows operativsystem för att kunna komma åt PLC data. MatrikonOPC erbjuder flera användargränssnitt möjligheter så som HMI, Databashantering, Webbapplikation m.m. Detta alternativ valdes bort för att MatrikonOPC kräver för högt licenspris för sin OPC server [13].

PhoneGap är ett applikationsspråk som utnyttjar standardbaserad webbt teknik och är gratis att använda. På grund av den tekniken blir PhoneGap operativoberoende plattform vilket menas att man kan utveckla en applikation till alla operativsystem som finns idag som till exempel IOS, Android och Windows Mobile. PhoneGap försöker ta bort onödiga trösklar som finns i dagens applikationsutveckling som olika programmeringsspråk som används för de olika operativsystemen. PhoneGap kodar applikationen i HTML5, JavaScript samt CSS3 [14].

5 Resultat

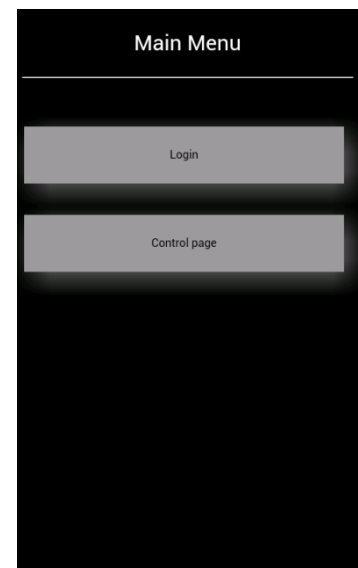
I detta kapitel sammanställs alla resultat som uppnåtts i kapitel 3 [Metod](#).

5.1 Webbservern

Det fungerade att aktivera webbservern i S7-1200 för att få fram de önskade resultaten vilka var att mobiltelefonen kunde ansluta till webbsidan. Android applikationen i mobiltelefonen kunde genom enbestämd IP adress komma åt de anpassade HTML sidorna "User defined webpage" vid start av applikationen. Med vald metod fick arbetet en låg utvecklingskostnad då bara redan innehavande licenser utnyttjades.

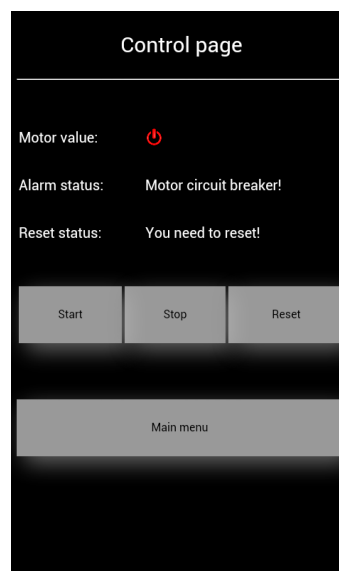
5.2 Applikationen

För att kunna styra motorn måste användaren logga in och sedan klicka på "Control Page". Vill användaren inte styra eller om denne saknar rättigheter så är det möjligt att läsa värden från motorn i "Control page" utan lösenord. "Control page" ser likadan ut om oavsett om användaren är inloggad eller inte, skillnaden är alltså vilka rättigheter denne har. Vid inloggning så klickas "Login" och användarnamn medlösenord skrivs enligt figur 18 i kapitel 6 [Diskussion](#). För att kunna styra motorn klickar användaren på "Control page" där applikationen kan starta, stoppa och återställa med "Reset" ifall alarmet har utlöst. Sidan läser även av värden från motorn där grön figur sidan om "motor value" indikerar på att motorn är igång och röd på att den står still. Ett alarmvärde och reset status finns att läsa av som skriver ut om ett alarm har lösts ut och om alarmet behöver återställas.



Figur 14. första sidan av S7-1200 applikationen vid start

I figur 15 står motorn still då alarmet "Motor circuit breaker!" har utlösts. Om användaren vill starta motorn måste först alarmet åtgärdas och sedan återställas för att sedan kunna starta motorn igen.



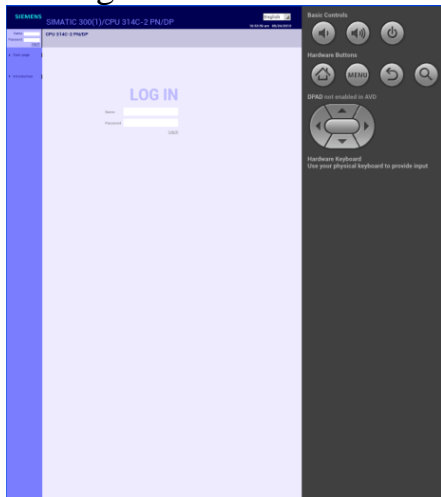
Figur 15. Control page för S7-1200.

Motorn står still, "Motor value" är röd

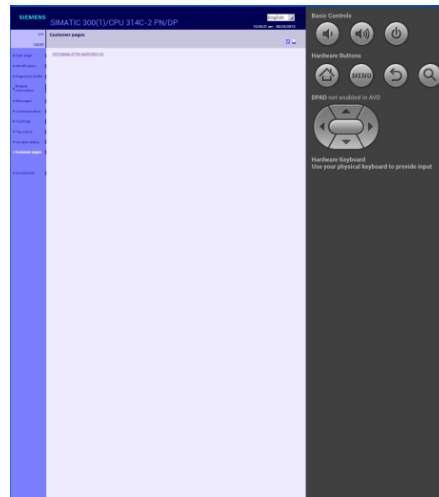
6 Diskussion

En Android baserad mobiltelefon kan kommunicera med en Siemens PLC S7-1200. Mobiltelefonens operativsystem har ingen betydelse så länge telefonen har en webbläsare som kan läsa HTML sidor. Det är då möjligt att både skriva och läsa av nya värden från enheterna som är kopplade till S7-1200. I figur 15 ”Control page” styrdes motorn av digitala värden så som ’1’ och ’0’ men motorn kan även ta emot och skicka analoga värden som ’15’ och ’200’ för motorns varv hastighet .

Samma applikation kunde användas för S7-300 som för S7-1200. Skillnaden var att på S7-300 applikationen så öppnas PLCns egna hemsida där användaren först måste logga in på Siemens webbserver för att sedan klicka sig vidare till de anpassade webbsidorna ”User defined webpage” via en länk som figur 17 visar.



Figur 16 Emulatorns första sida av S7-300 applikationen vid start



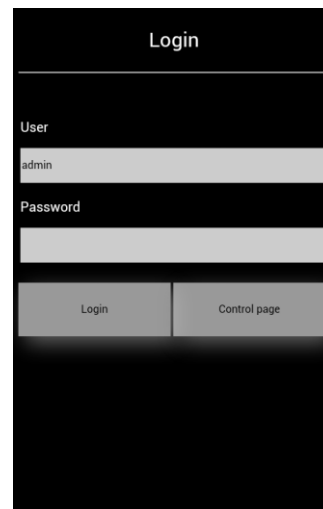
Figur 17 Länk till de anpassade webbsidorna

Applikationen som skapades för Android kan även skapas för IOS – enheter men kräver lite kunskap om ”Objective C” vilket är det språket som Apple kodar sina applikationer med [15].

Siemens har en "FormLogin" fil som kontrollerar inloggningen i webbservern. Den filen kunde inte korrigeras av säkerhetsskäl från Siemens vilket ledde till att inloggningen inte skedde på det sätt som vi hade tänkt oss. "FormLogin" kontrollerar ifall inloggningen lyckades och stannar kvar på sidan. Vi försökte få den att kontrollera inloggningen och sedan skicka oss vidare till "Control page" utan att lyckas.

En sådan applikation som detta examensarbete omfattar får inte ha höga utvecklingskostnader, detta var ÅF Industry tydliga med i början av projektet. De alternativa metoderna valdes bort på grund av att de hade höga licenskostnader och kunden skulle betala för utvecklingskostnaden och inte för tredjeparts licenser vid köp av tjänsten. Därför valdes metoden för webbservern som beskrivs i kapitel 4.

Motorn hade kunnat kommas åt via routerns publika IP adress, dvs. utanför hemnätverket, men IT avdelningen på ÅF ansåg att det var riskfullt att tilldela routern ett sådant så det testades inte. Möjligheten finns och det är upp till varje företag att bedöma riskerna. Ett alternativ är att koppla PLC till en sändare med ett abonnemang och via den IP adressen nå webbservern. För att få en konstant adress kan företaget köpa en domän som de namnger och ansluter till när de vill komma åt webbservern. Räcker det att komma åt PLC från hemnätverket så bestäms en statisk lokal IP adress som applikationen laddar upp.



Figur 18. "Login page" för S7-1200.

7 Slutsats

Med applikationen som rapporten beskriver kan en maskinoperatör ha koll på flera enheter samtidigt med mobiltelefonen. Operatören kan få alarm innan något går fel och stoppa maskinen på distans för att sedan starta den igen när värdena har gått tillbaka till det normala. Chefen på produktionen kan genom några knapptryck på mobilen kontrollera så alla stationer är aktiva.

Varje applikation måste anpassas efter ett specifikt PLC program. Därför kan inte applikationen prissättas och läggas ut på ”Google Play” för kunder att ladda ner. Om en kund vill ha en applikation så måste den skrivas på nytt enligt metoden på kapitel 4. Ett pris kan inte heller sättas på applikationen då PLC program varierar i storlek och som med HTML – sidorna så anpassas priset efter PLC programmet.

Metoden med webbservern för S7-1200 blev den billigaste för ÅF Industry då webbservern aktiveras genom TIA Portal vilken ÅF har licens till. IS7-300 användes också webbservern på grund av för höga licenspriser även om metoder med bättre layout fanns.

7.1 Framtida utvecklingsmöjligheter

För framtida projekt kan denna rapport användas för skapandet av applikationer till enheter via Siemens PLC. Avancerade PLC program kan styra flera enheter. Med egen definierade webbsidor som dessutom är mobilanpassade till en PLC så kan operatören styra enheter trådlöst via mobilen utan att vara i samma lokala nät, t.ex. hemifrån. Detta kräver större kunskap inom IT – säkerhet och autentisering för att inte riskera intrång och manipulering av enheter.

8 Referenser

Åtkomstdatum för samtliga http – referenser är[2013-06-12].

8.1 Webblänkar

1. <http://www.afconsult.com/sv/Om-AF/Om-AF/>
2. <http://frapps.se/2012/05/22/smartphoneanvandande-i-sverige/>
3. <http://www.eclipse.org/org/>
4. <http://developer.android.com/sdk/index.html>
5. <http://notepad-plus-plus.org/>
6. <http://www.w3schools.com/html/>
7. http://www.w3schools.com/css/css_intro.asp
8. <http://www.java.com/en/about/>
9. <http://www.w3schools.com/js/>
10. <http://www.automation.siemens.com/mdm/default.aspx?Language=en&ShowMsg=false&DocVersionId=41524141835&GuiLanguage=en&cssearchengine=NEW>
11. <http://support.automation.siemens.com/WW/llisapi.dll?func=cslib.csinfo&objId=22781365&load=content&lang=en&siteid=cseus&aktprim=0&objaction=csview&extranet=standard&viewreg=WW>
12. <http://developer.android.com/reference/android/webkit/WebView.html>
13. <http://www.matrikonopc.com/opc-drivers/opc-siemens-s7-plc/base-driver-details.aspx>
14. <http://phonegap.com/about/>
15. <http://developer.apple.com/library/mac/#documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html>

8.2 PDF manualer

16. Manual för S7-1200, kan hittas direkt med pdf namnet eller via den här länken:

http://cache.automation.siemens.com/dnl/zA/zAxNDQ3AAAA_58862931_Tools/58862931_S7-1200_Webserver_DOKU_v10_en.pdf

[Hämtad: 2013-06-17]

58862931_S7-1200_Webserver_DOKU_v10_en.pdf

17. Manual för S7-300, kan hittas direkt med pdf namnet eller via den här länken:

http://cache.automation.siemens.com/dnl/TM/TM0ODIzMQAA_44212999_Tools/44212999_AWP_PN_CPU_V10_e.pdf

[Hämtad: 2013-06-17]

44212999_AWP_PN_CPU_V10_e.pdf

18. Manual för "Sm@rtAccess". Den kan hittas via programmet WinCC Flexible genom att trycka på "about" och sedan klicka på den pdf filen som handlar om "Sm@rtAccess".

WCF07WebFeaturesenUS.pdf