CHALMERS

Utveckling av Smart Logger

Ett system för stopptidsmätning och maskindiagnos

Examensarbete inom Högskoleingenjörsprogrammet i datateknik

RICHARD NORLING
KRISTOFFER PETERSSON

Institutionen för data- och informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2014

Utvecklingen av Smart Logger
Ett system för stopptidsmätning och maskindiagnos

Richard Norling, Kristoffer Petersson

© RICHARD NORLING, KRISTOFFER PETERSSON, 2014

Institutionen för data- och informationsteknik
Chalmers tekniska högskola
412 96 Göteborg
Tel: 031-772 1000
Fax: 031-772 3663

Institutionen för data- och informationsteknik
Göteborg, 2014

Sammanfattning

Rapporten beskriver arbetet kring utvecklingen av en loggenhet för ett system som utför stopptidsmätning och maskindiagnos i säkerhetssyfte. Loggenheten som ska utvecklas är en central del i ett systemet som kallas Smart. Till denna enhet kopplas externa enheter som pulsgivare som används för att mäta maskinrörelser och stopp-utrustning som ljusridåer, nödstopp m.m. Innan en mätning ställer man in en uppsättning av villkor som krävs för att starta och avsluta mätningen. Under mätningen samplar enheten all data från givare. Efter mätningen visualiseras den i en graf, den datan som visas är t.ex. position, hastighet och värde på en analogingång. Det befintliga Smart-systemet kräver en uppkoppling mot en PC där applikationen Smart Manager styr loggenheten och visualiserar informationen. Alla inställningar görs i Smart Manager eftersom att loggenheten saknar användarinput och display. Den nya loggenheten ska kunna utföra mätningar utan en PC, den ska vara utrustad med en touch-skärm för input där alla inställningar lätt kan knappas in, enheten ska även kunna visualisera datan i en graf med begränsade zoom-möjligheter.

Arbetet har resulterad i en prototyp bestående av ett utvecklingskort från MikroElektronika som heter ”mikromedia for dsPIC33EP” samt en applikation. Prototypen kan användas för att mäta rörelser från en pulsgivare, beräkna hastigheten hos rörelserna, kontrollera olika villkor, visualisera datan i en graf och sedan spara mätningen på ett SD-kort, allt fungerar utan en uppkoppling till en PC.

Abstract

This thesis describes the research and development of a logger unit for a system that measures stopping characteristics and movement of machines for safety purposes. The logger unit is a central part of the system called Smart. This unit can be connected to external devices such as rotary encoders for measuring movement and stopping equipment including light beams, emergency buttons and more. Before a measurement can be made it is required to set up conditions that will trigger the start and end of the measurement. The log unit samples data from sensors during the measurement. After the measurement is done the data is visualized in a graph, the represented data can for example be position, velocity and analogue values from the AD. The currently existing Smart system requires a connection with a PC where an application called Smart Manager controls the logger unit and visualizes the collected data. All settings for the measurement are set up in Smart Manager since the logger unit lacks any form of user input and display. The new logger unit that is to be developed shall perform the measurements without the need for a PC, it should also be equipped with a touch screen where all the configurations can easily be set, and after the measurement is done the unit should be able to visualize the data in a graph with limited zoom possibilities.

The work has resulted in a prototype consisting of a development board from MikroElektronika called *"mikromedia for dsPIC33EP"* and an application. The prototype can be used to measure movement from rotary encoders, calculate the velocity of the movements, verify measurement conditions, visualize the collected data and then save it the a SD-card. Everything can be done without a connection to a PC.

Förord

Detta examensarbete har utförts under våren 2014 som en del av högskoleingenjörsprogrammet för datateknik vi Chalmers Tekniska Högskola (CTH). Arbetet motsvarade 15 högskolepoäng.

Vi skulle vilja tacka våra handledare, Ingvar Gillholm och Daniel Strigård för deras stöd under projektet. Utan dem hade vi inte haft möjlighet att utföra detta examensarbete på ABB Jokab Safety i Kungsbacka. Utöver dessa två vill vi även rikta ett stort tack till samtliga anställda på företaget.

Vi vill även tacka vår handledare på Chalmers, Peter Lundin för hans otroliga hjälpsamhet och vägledning.

Göteborg 12 juni 2014

Richard Norling & Kristoffer Petersson

Ordlista

- **CPU**
  Central Processing Unit är den hårdvara i datorn som utför instruktionerna i ett datorprogram.

- **Fas**
  Består av en uppsättning villkor för en mätning, innehåller information om vilka utsignaler som ska aktiveras när villkoren uppfylls.

- **Fasavancering**
  Övergången från en fas till en annan fas i Smart Logger. Detta sker när villkoren för en fas är uppfyllda.

- **Inkrementell pulsgivare**
  Ett intstument som används för att mära rörelser och hastighet. Måste utgå gtån en nollposition om spänningen bryts, till skillnad från absoluta pulsgivare.

- **MCU**
  Microcontroller Unit är en liten dator på en integrerad krets som innehåller bl.a. en CPU, minne och diverse periferienheter.

- **Pulsgivare**
  Ett instrument som används för att mära rörelser och hastighet

- **QEI-modul**
  En periferienhet på särskilda microcontrollers (dsPIC) som erbjuder ett interface för att läsa av inkrementella pulsgivare.

- **SPI**
  Serial Peripheral Interface är en databuss för synkron seriekommunikation som arbetar i full duplex.

Innehållsförteckning

1. Inledning 1
   1.1 Bakgrund 1
   1.2 Syfte 2
   1.3 Mål 2
   1.4 Avgränsningar 2
2. Metod 3
3. Teknisk bakgrund 4
   3.1 Det befintliga Smart systemet 4
   3.2 Material 5
      3.2.1 Utvecklingskortet ”mikromedia for dsPIC33EP” 5
   3.3 Utvecklingsmiljöer 6
   3.4 Pulsgivare 6
      3.4.1 Inkrementella pulsgivare 6
      3.4.2 Absoluta pulsgivare 7
   3.5 Filsystem 8
      3.5.1 FAT32 8
4. Genomförande 10
   4.1 Marknadsundersökning 10
   4.2 Kravspecifikation 10
   4.3 Implementering av filhanterare 11
      4.3.1 Begränsning av funktionalitet 11
      4.3.2 Utformning 11
      4.3.3 Struktur 13
   4.4 Mätning och inställningar 14
      4.4.1 Översiktsskärmen 14
      4.4.2 Fasinställning 15
   4.5 Kalibrering 17
   4.6 Mätning 18
   4.7 Visualisering av mätdata 20
      4.7.1 Utformning av grafen 20
      4.7.2 Skalning av mätdata 22
   4.8 Lagring av mätdata 22

4.8.1 Lagring i FAT32 23
5. Resultat 24
5.1 Prototypen 24
6. Slutsats 26
6.1 Resumé 26
6.2 Kritisk diskussion 27
6.3 Vidareutveckling 27
7. Referenser 29
Bilaga A. Sammanfattning av muntlig kravspecifikation för nya Smart Logger 30
Bilaga B. Fil med mätdata 31
Bilaga C. Planeringsrapport, Ganttschema 32

1. Inledning

1.1 Bakgrund

ABB [1] är ett av världens största företag och är verksamma inom elkraft, robot- och automationsteknik. ABB koncernen, med sitt huvudkontor i Zürich, Schweiz är aktiva i ungefär 100 länder och har sammanlagt runt 150 000 anställda.

ABB är idag uppbyggt av fem divisioner. Den division examensarbetet skall utföras inom är ”Lågspänningsprodukter” och företaget heter ABB Jokab Safety. Denna del av företaget utvecklar produkter för både hem och industri. Några av de produkter de utvecklar är lågspänningsbrytare, kabeltillbehör och kabelsystem.

ABB Jokab Safety var under en längre tid ett fristående företag men blev för några år sedan en del av ABB koncernen. De har totalt fem kontor runt om i Sverige och där vi kommer vara verksamma är i Kungsbacka, strax söder om Göteborg. ABB Jokab Safety utvecklar produkter och lösningar inom maskinsäkerhet.

I industrier finns det krav och standarder på hur man ska hantera olika typer av säkerhetsaspekter för maskiner. För att skydda operatören av en viss maskin måste särskild skyddsutrustning placeras på lämpliga platser omkring maskinen. En typ av skyddsutrustning kan vara detektering om operatören närmar sig maskinen vid drift. Denna skyddsutrustning måste då skicka en stoppsignal till maskinen så att den hinner stanna innan operatören riskerar att skadas. För att kunna bestämma var denna skyddsutrustning ska vara placerad krävs att man vet hur snabbt man kan stoppa en viss maskin från det att skyddsutrustningen skickat sin stoppsignal.

Smart [2] är ett verktyg för att göra stopptidsberäkning och analys på hur maskiner beter sig vid drift, stoppfas etc. Hela Smart systemet är uppbyggt av flera delar bl.a. en loggenhet, ett mjukvaruprogram för visualisering av mätdata, diverse givare för att mäta hastigheter och krafter samt stoppenheter för att generera stoppsignaler.

I dagsläget är det ett krav att loggenheten är kopplad mot en PC för att kunna använda systemet. Detta eftersom man via ett PC-program kallat Smart Manager styr själva enheten. I Smart Manager sätter man upp en mätning genom att ange ett antal villkor som bestämmer när mätningen ska starta och stoppa. Mätning laddas över till loggenheten som i sin tur ansvarar för inhämtningen av data från de inkopplade givarna. Loggenheten sparar inte någon data, utan vidarebefordrar datan till Smart Manager.

1.2 Syfte

Syftet med detta arbete är att undersöka om man kan utveckla en ny loggenhet för smartsystemet på ett sätt som ger en pålitligare och mer användarvänlig produkt för stopptidsmätning och maskindiagnos.

1.3 Mål

Målet med detta examensarbete är att utveckla en ny loggenhet för ABB Jokab Safetys Smart-system så att den uppfyller kundernas nuvarande krav. Den nya produkten ska kunna utföra, spara och visualisera stopptidsmätningar för industrimaskiner utan ett PC-system inkopplat, vilket idag är ett krav.

1.4 Avgränsningar

Projektet kommer enbart att bestå av utvecklingen av mjukvara. Hårdvaran kommer att bestämmas inom ramen för projektet men först efter det att en marknadsundersökning bland ABB:s befintliga kunder som använder Smart-systemet har utförts.

Projektet kommer inte att undersöka huruvida andra externa enheter kan användas till den nya prototypen, utan kommer att använda befintliga externa enheter.

Vi kommer inte att undersöka andra utvecklingsmiljöer än de som rekommenderats av ABB Jokab Safety.

2. Metod

Utvecklingsmetoden som är tänkt att användas under projektet har inspirerats av Scrum[3] och Kanban[4] där fokus ligger på att använda sig av en iterativ utvecklingsprocess. Under projektet kommer dagliga statusrapporter att ske, detta tillsammans med utnyttjandet av en whiteboard för att visualisera de processer som ska göras, är aktiva samt klara borde ge en bra överblick över vad som arbetas med den aktuella veckan. Metoden kommer att göra det enkelt att organisera projektet för att kunna arbeta på ett så effektivt sätt som möjligt.

Hela projektet utförs vid ABB Jokab Safetys kontor i Kungsbacka. Det kommer att finnas tillgång till hjälp ifrån mjukvaruutvecklare på kontoret vilket underlättar om utvecklingen kör fast. En annan anledning till att projektet utförts på deras kontor är att det finns tillgång till utrustning som är nödvändig för att kunna testa den prototyp som utvecklas.

Inledningsvis kommer det att tas fram en planeringsrapport som man sedan, under projektets gång, kan använda som riktlinje för att bestämma vilka delar som det ska arbetas med.

3. Teknisk bakgrund

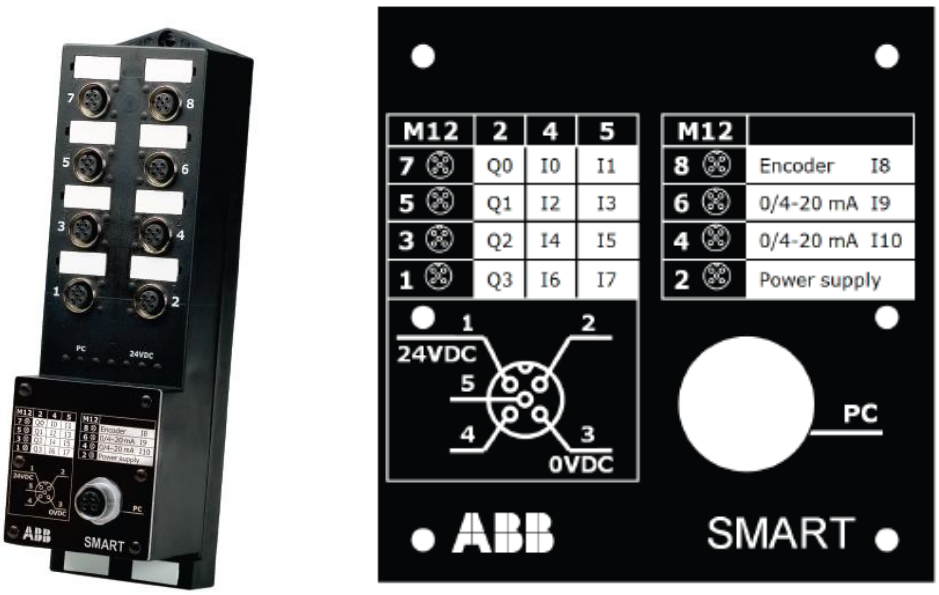
Under detta kapitel ges information som är nödvändig för att, på ett enkelt sätt, kunna förstå rapportens senare delar. Kapitlet går igenom bibliotek, material och tekniker där fördjupad kunskap behövts inhämtats för att kunna skapa vissa delar av den prototyp som utvecklats. Det ger även en förståelse till varför vissa val har gjorts och det diskuteras och jämförs för- och nackdelar med dessa val.

3.1 Det befintliga Smart systemet

För mätning av position och hastighet används främst ABB Jokab Safetys egna givare SM5 1250 och SM5 2500. Dessa enheter kopplas mot loggenhetens encoderingång som behandlar positionsdata som med hjälp av en QEI-modul. Systemet har ett konstant samplingsintervall på 1ms.

Den befintliga loggenheten har totalt 7 st I/O-portar av kontakttypen M12. Fyra av dessa är digitala och har två ingångar och en utgång vardera. Utöver dessa finns en encoderingång och två analogingångar för möjligheten att ansluta analoga givare.

De digitala I/O-portarna kan användas till att skicka signaler till maskinsystemet som man testar, men även skicka signaler till annan stopputrustning som hjälper till att diagnosera systemet. Som Figur.3.1 visar betecknas utgångarna Q0-Q3 och ingångarna I0-I7. För att styra Smart Logger används PC-programmet Smart Manager, där alla inställningarna och mätningar konfigureras. En USB-kabel används för kommunikationen mellan Smart Manager och Smart Logger.



Figur 3.1. Smart logger och dess kapsling samt anslutningar

3.2 Material

Materialet som använts i projektet har tillhandahållits av ABB Jokab Safety. Att just dessa produkter använts beror på att de är kompatibla med den befintliga produkten och är också tänkta att användas tillsammans med den nya produkten.

- *SM5 1250/2500*
  Inkrementell pulsgivare för att mäta position och hastighet på rörliga maskindelar. Produkt från ABB Jokab Safety.

- *SM11 Flag Unit*
  Flaggenhet för att ge stoppsignal till skyddsutrustning. Produkt från ABB Jokab Safety.

3.2.1 Utvecklingskortet ”mikromedia for dsPIC33EP”

En mikrocontroller (MCU) är en liten dator som är tillverkad på ett och samma kretskort. Den består av CPU, programminne, arbetsminne samt ett antal in- och utportar för att kunna kommunicera med andra enheter. En MCU används oftast i små inbyggda system för att styra och reglera diverse produkter. Produkter innehåller mikrocontrollers är t.ex. fjärrkontroller, hushållsapparater, kontrollsystem av motorer.

Det finns många olika typer av mikrocontrollers, vilken som bör användas beror på vad den ska utföra för typ av uppgift. För detta projekt valde ABB Jokab Safety att använda sig av MikroElektronikas [5] utvecklingskort *”mikromedia for dsPIC33EP”* för att skapa en prototyp. Detta utvecklingskort har en *dsPIC33EP512MU810* från MicroChip som MCU. Denna har en 16 bitars kärna som är baserad på den välkända Harvard-arkitekturen. På MCU:n finns även en rad integrerade moduler för utveckling av diverse olika funktioner. Värt att nämna är de två AD-modulerna, timer-modulerna samt ”Quadrature Encoder Interface”-modulen (QEI).

Nedan listas de moduler på utvecklingskortet som har använts under projektet.

- **Serial Flash Memory**
  Ett seriellt flashminne på 1MB som man kommunicera med via SPI.

- **microSD**
  microSD kort för lagring av större filer. Kan kommuniceras med via SPI.

- **miniUSB**
  För kommunikation med andra USB enheter, stödjer USB 2.0 OTG

- **TFT with Touch Panel**
  320x240 pixlars färgskärm med stöd för touch.

- **Quadrature Encoder Interface, QEI**

3.3 Utvecklingsmiljöer

Under projektet har två utvecklingsmiljöer använts. Dessa rekommenderades av handledarna på ABB Jokab Safety då de är kompatibla med det utvecklingskort som använts.

- **microC**
  microC är en ANSI C kompilator med tillhörande IDE. Till microC finns en hel del standard mjukvarubibliotek samt diverse annan funktionalitet för att göra utvecklingsprocessen så smidig som möjlig. Genom en hemsida som heter LibStock[6] kan man även enkelt ladda ned externa bibliotek som sedan kan användas.

- **Visual TFT**
  Visual TFT är en fristående applikation för utveckling av grafiska interface för TFT displayer. Denna utvecklingsmiljö generarar kod för de skapade grafiska komponenterna. Denna kod kan sedan enkelt laddas in till microC.

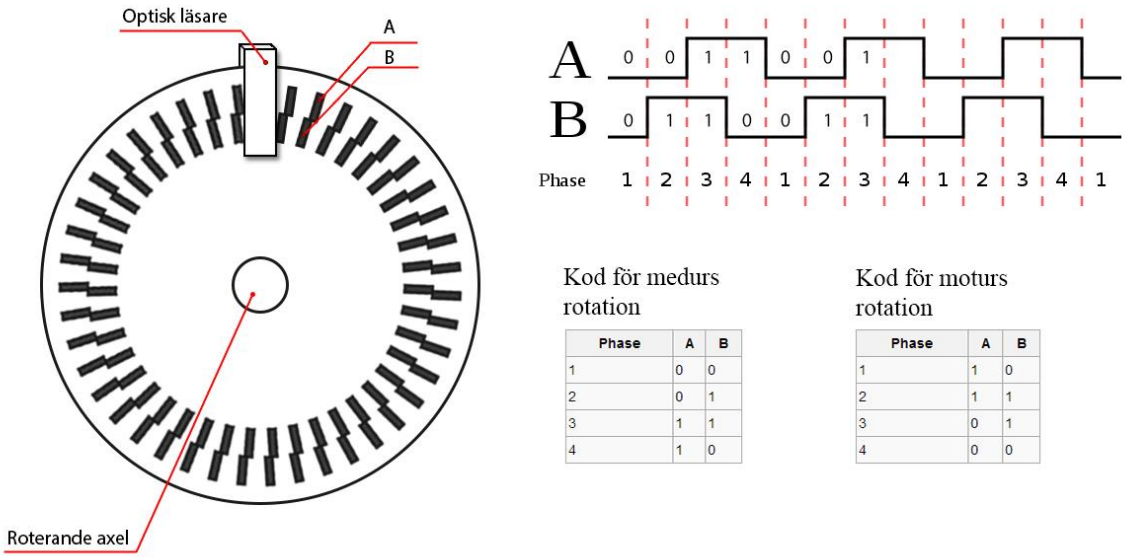
3.4 Pulsgivare

Pulsgivare är en typ av komponent som läser av position från rörliga maskindelar de fästs vid. Detta är nödvändigt för maskiner där precision är en viktig egenskap. Återkopplingen från givaren blir vital för snabba och exakta justeringar och av rörliga maskindelar. Det finns två typer av pulsgivare, inkrementella och absoluta. Pulsgivarna *SM5 1250/2500* som kommer att användas i detta arbete är av typen inkrementella pulsgivare. Dessa givare fästs vid maskiner med hjälp av en vajer och värderna som skickas från givarna indikerar hur långt vajern är utdragen.

3.4.1 Inkrementella pulsgivare

Den inkrementella pulsgivaren saknar kunskap kring vilken exakt position som den befinner sig i. Den skickar ut pulser vid en förändring hos den rörliga maskindelen den är fäst vid, relativt till positionen innan. Pulsgivaren ställs in för att generera ett visst antal pulser per millimeter. Mängden pulser är det som bestämmer pulsgivarens upplösning, högre upplösning medför en bättre precision. Med den inkrementella pulsgivaren måste man i mjukvara räkna antal pulser som skickas från givaren och sedan spara detta för att spåra axelns position.

Inuti pulsgivaren sitter en särskild skiva som är fastmonterad kring en roterande axel. Hos pulsgivarna *SM5 1250/2500* så är det det utdragningen av fästvajern som bestämmer rotation på axeln och därmed rotationen hos skivan. Figur 3.2 visar hur skivan kan se ut, bilden visar även hur 2 st ringar är markerade, de brukar betecknas A och B. Markeringarna för A-ringen och B-ringen avläses med hjälp av en optisk läsare, dessa bildar sedan ett pulståg enlig Figur 3.2. Dessa pulståg skickas sedan till QEI-modulen där pulserna jämförs mot varandra enligt sanningstabellen i Figur.3.2 för att bestämma vilken riktning axeln rör sig.



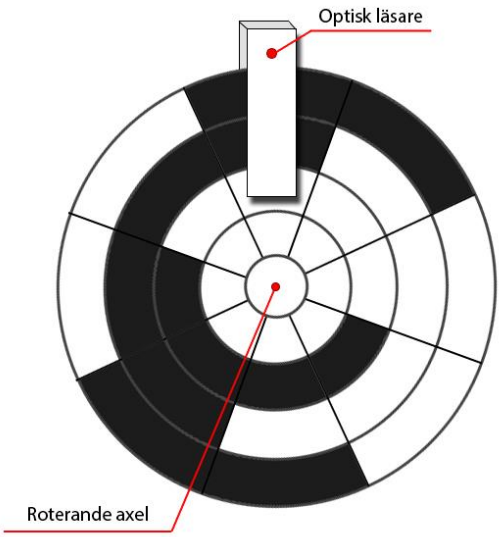
Phase	A	B
1	0	0
2	0	1
3	1	1
4	1	0

Phase	A	B
1	1	0
2	1	1
3	0	1
4	0	0

Figur 3.2. Beskrivning av hur man utläser rörelse från en inkrementell pulsgivare

3.4.2 Absoluta pulsgivare

Utsignalen från en absolut pulsgivare består inte av enskilda pulser som hos en inkrementell, istället skickas den exakta positionen. Vill man mäta och kalkylera rörelser och inte bara position måste det hanteras i mjukvara. Fördelen med en absolut pulsgivare är att den fungerar även när strömmen är avstängd. En omstart av systemet kommer inte att påverka den absoluta pulsgivarens uppfattning om positionen. Figur 3.3 visar ett förenklat exempel på en encoder hos en absolut pulsgivare med 8 olika värden på ett rotationsvarv. Antal värden per rotationsvarv bestämmer upplösningen. En optisk läsare kan sedan läsa vilket segment som visas och sedan bestämma vinkeln hos axeln.



Figur 3.3. Absoluta pulsgivare använder oftast greycode som encoding för positionen

3.5 Filsystem

Ett filsystem används av datorer för att kontrollera hur information lagras och inhämtas. Utan ett filsystem skulle denna information placeras i minnet utan någon som helst struktur och göra det omöjligt för användaren att få ut den information som efterfrågas. Ett filsystem fungerar i enkelhet så att den delar upp datan som lagras i olika enheter som sedan kopplas samman för att bilda en typ av katalogstruktur.

Det finns många olika typer av filsystem, alla har olika bakomliggande logik och struktur. Det som kan skilja de olika filsystemen är t.ex. hur bra prestanda de har (hastighet och minneseffektivitet), vilken nivå av säkerhet dee använder och hur stora filer systemet kan hantera. Det filsystem som använts i detta projekt för att lagra filer är FAT32 [7].

3.5.1 FAT32

FAT32 är ett filsystem som är utvecklat för DOS och Windows i första hand, men även senare versioner av Linux klarar av att hantera detta system. FAT32 är en vidareutveckling av det ursprungliga filsystemet FAT.

FAT32 är uppbyggt av tre olika regioner:

- **Reserverade sektorer**
  Den första reserverade sektorn är ”Boot” sektorn som innehåller information om vilken typ av filsystem som används och pekare till de övriga reserverade sektorerna.

- **FAT regionen**
  I denna region av filsystemet finns en tabell *"File Allocation Table"* som håller reda på vilka kluster i minnet som innehåller filer eller mappar. Filer som är tillräckligt stora kan täcka flera olika kluster och det finns därför även sparat i tabellen var filen fortsätter.

- **Dataregionen**
  I FAT32, till skillnad från det ursprungliga FAT-systemet, är även rot regionen en del av data regionen. I FAT kunde rotkatalogen endast innehålla ett visst antal filer och mappar, i FAT32 finns inte denna restriktion utan den behandlas på samma sätt som andra mappar.

  Dataregionen i sig är den region som innehåller all information om det som är lagrat i alla filer och mappar. Denna del av filsystemet är den största rent storleksmässigt.

En stor fördel med FAT32 jämfört med FAT är att det ökar det totala fria utrymmet på lagringsenheten. I FAT finns det ett tak på hur många kluster en partititon får innehålla. Detta medför att ju större partition som används, desto större blir också klustren. Detta orsakar problem i FAT filsystem då ett kluster endast kan innehålla information från en fil eller en mapp.

För att ge ett exempel kan man tänka sig att man ska lagra en fil med storleken 1kB på en FAT partition av storleken 2GB. Detta medför att denna fil på endast 1kB kommer ta upp utrymme från ett helt kluster på 32kB. Skillnaden i FAT32 är att klusterstorleken för partitioner under 8GB är konstant på 4kB. Använder man FAT32 istället för FAT hade man sparat 28kB per fil, förutsatt att alla filer är av storleken 1kB. Detta sparar i längden en stor del utrymme i lagringsenheten [8].

4. Genomförande

I detta kapitel ges en beskrivning över hur utvecklingen av den nya Smart-enheten har utförts. Det presenteras beskrivningar av de utvecklingsarbeten som gjorts, men även problem och viktiga val som uppstått diskuteras.

4.1 Marknadsundersökning

Innan projektet börjat hade ABB Jokab Safety själva gjort lite informella förfrågningar till olika användare av Smart-systemet angående vilka förbättringar de kunde tänkta sig. Efter en hel del kritik på den befintliga produkten, bestämde de sig för att en ordentlig marknadsundersökning skulle ingå som en del i examensarbetet.

ABB Jokab Safety hade innan projektet startat tagit fram ett formulär med frågor angående vad som kunde förbättrats med den befintliga Smart-enheten. Inom ramen för detta projekt gjordes en marknadsundersökning som pågick i cirka en vecka och innefattade samtal med ABB:s säljorganisation samt återförsäljare och användare av Smart-systemet världen över. Samtalen var väldigt givande och gav en bra bild över hur kunderna uppfattade den befintliga produkten, samt gav bra feedback på nya funktionaliteter.

Undersökningen utfördes främst för att säkerställa att den nya produkten skall möta kundernas krav, men även för att minimera kostnaderna för den befintliga produkten genom att se till att ingen överflödig funktionalitet implementeras. Detta är en viktig del i dagens samhälle då mycket handlar om att hålla nere kostnader och energiförbrukning så mycket som möjligt för att främja ett hållbart samhälle.

4.2 Kravspecifikation

Då marknadsundersökningen väl var genomförd sammanställdes den insamlade informationen. Denna sammanställning gavs sedan till handledarna på ABB Jokab Safety som tog fram en kravspecifikation utifrån detta.

Några av de förändringarna mot den nuvarande produkten är:

- Ska vara en fristående enhet, som inte behöver vara kopplad mot en PC för att kunna användas.
- Den nya Smart-enheten ska endast innehålla en analog ingång (12 bitar) till skillnad mot föregångarens två (10 bitar).
- Kunna ställa in samplingsintervallet för samtliga ingångar i systemet. De föreslagna intervallen är 0.1 ms, 1 ms, 10 ms, 100 ms samt 1000 ms.

4.3 Implementering av filhanterare

Eftersom ett av kraven för prototypen är att kunna använda ett SD-kort för lagring av mätdata togs ett beslut tidigt att en filhanterare var nödvändig. Nedan presenteras hur utvecklingen av denna gått till, samt vilka problem och frågeställningar som har uppstått.

4.3.1 Begränsning av funktionalitet

Eftersom SD-kortet inte är åtkomligt fysiskt för användaren, utan är låst till loggenheten, behövde det implementeras viss funktionalitet för filhantering. Det optimala hade självklart varit att användaren hade haft tillgång till en stor del av den funktionalitet som t.ex. Utforskaren i Windows tillhandahåller, men detta var inte ett rimligt alternativ då displayen på enheten endast är 320x240 pixlar. Med en så liten display skulle det bli förvirrande för användaren om för många alternativ presenterades samtidigt. Den första utmaningen blev att bestämma vilka olika operationer som användaren skulle kunna göra på filsystemet.

För att man enkelt ska kunna strukturera sina mätningar behövs det funktionalitet för att kunna gruppera filer, alltså bestämdes det att skapande, redigerande och borttagning av mappar skulle implementeras. Efter diskussion med handledarna på ABB Jokab Safety togs beslutet att kopiering av mappar inte var nödvändig. De operationer man skulle kunna göra på mappar begränsades till dessa:

- Skapa
- Redigera namn
- Ta bort

De filer som sparas i systemet innehåller endast information om olika mätningar som genomförts. Det kommer alltså inte att finnas några filer som innehåller något annat än detta. Även här valdes att inte implementera kopiering. Operationerna man kan göra på filer begränsades till dessa:

- Skapa (dock inte från filhanteraren)
- Öppna (visar grafer på mätdata)
- Redigera namn
- Ta bort

4.3.2 Utformning

Under utvecklingen av filhanteraren uppstod frågan om att tillåta skapande av mappar i andra mappar, alltså tillåta djupa filstrukturer. I ett vanligt system hade detta kanske setts som en nödvändighet, men eftersom den nya Smart-enheten ska hålla filhanteraren så smidig och enkel som möjligt var detta inte någon självklarhet.

För att ge ett exempel på hur Smart-systemet används i allmänhet, så brukar det gå till så här:

Exempel

---

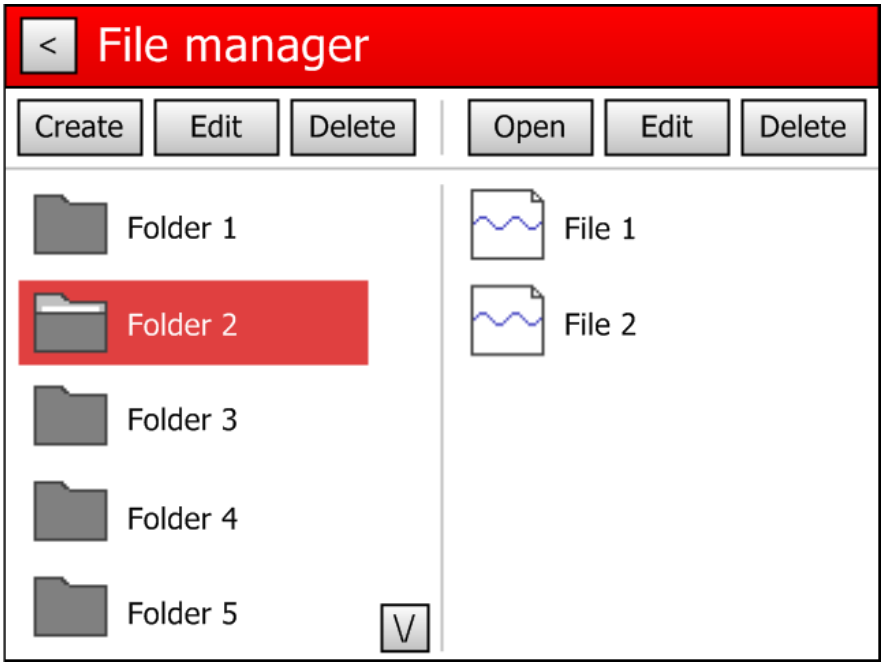
1. En industri behöver göra en årlig kontroll på sina *n* stycken maskiner
2. Använder Smart för att göra mätningar på dessa maskiner.
3. Det brukar göras ca 10 mätningar per maskin. Dessa mätningar grupperas och sparas.
4. Efter att alla mätningar är gjorda slås dess stopptider samman och bildar ett medelvärde.
5. Detta medelvärde tillsammans med analys av positions- och hastighetsgraf ligger sen som grund för hur man placerar skyddsutrustningen.

Med denna information som grund blev valet av utformning på filhanteraren inte så problematiskt som det verkade från början. Som man kan utläsa från exemplet ovan, motsvarar en maskin en mapp, medan en mätning för denna maskin motsvarar en fil som placeras i denna mapp.

Om man ska tillåta djupare filstrukturer än två nivåer växer komplexiteten på systemet, både kodmässigt och rent grafiskt. Eftersom man med ett tvånivåers filsystem klarar av att hantera den typ av mätningar som utförs föll valet på ett sådant filsystem.

Då detta väl var bestämt började arbetet med att designa hur filhanteraren skulle se ut rent grafiskt. Eftersom displayen är så pass liten låg fokus på att skapa ett så enkelt och användarvänligt interface som möjligt. I och med att endast två nivåer skulle användas gjordes valet att visa dessa nivåer på samma skärm. På vänster sida av filhanteraren visas mapparna och på höger sida visas vilka filer som finns i en vald mapp. En begränsning gjordes i form av att endast fem objekt av samma typ kan visas samtidigt. För att få tillgång till de eventuellt resterande objekten implementerades upp och ner knappar som agerar som en scrollfunktion (se Figur 4.1).

Efter att denna grova utformning var gjord kom problemet med var vi skulle placera de olika knapparna för att kunna utföra operationer på mappar och filer. Eftersom de grafiska utvecklingsverktygen inte klarar av att hantera fler än ett lager hade stora problem uppstått om man velat använda sig av en så kallad ”context menu”, alltså en meny man får upp genom att högerklicka på ett objekt.



Figur 4.1. Layout för filhanteraren i Smart Logger

Lösningen på detta problem blev att skapa en ”toolbar” precis under actionbaren där knapparna för dessa operationer kunde vara.

4.3.3 Struktur

Innan kodandet av filhanteraren kunde börja behövdes efterforskingar göras angående hur man lagrar data i ett FAT32-filsystem. Det visade sig att det inte var helt trivialt att göra detta på egen hand, men efter lite sökande bestämdes att hjälpbiblioteket ”*FAT32_Library*” från MikroElektronika [9] skulle användas för att underlätta kommunikationen med lagringsenheten. Genom att använda detta biblotek kunde egna drivrutiner skrivas för att enkelt lagra specifik data för Smart på lagringsenheten.

FAT32_Library använder sig vid skapande, redigering och borttagande av filer och mappar av funktioner som endast behöver ha namnet på objektet som inparameter. Detta gjorde valet av struktur förhållandevis simpelt. Eftersom bara namnet på objektet i fråga behövdes för att kunna utföra diverse operationer på det implementerades inga avancerade strukturer internt för att hålla reda på vilka filer och mappar som skulle visas. Internt används bara två listor som innehåller namnet på mappar som finns tillgängliga, respektive de filer som finns i den aktuella mappen.

För att kunna lagra och visa dessa filer och mappar i filhanteraren behöver man således kunna allokera dynamiskt minne. Detta eftersom man initialt inte vet exakt hur många filer respektive mappar det existerar i filsystemet. En av skillnaderna mellan C och andra programmeringsspråk som Java och C# är att C inte har någon automatisk ”*Garbage*

Collection" vilket innebär att man som programmerare måste ta ansvar för de minnesareor som dynamiskt allokerats.

För att dynamiskt allokera minne i C använder man funktionen *malloc.* Denna funktion tar storleken, i byte, på den minnesarea som skall allokeras och returnerar en void pekare till arean. Ett exempel på dynamisk allokering är följande, där en buffert för 10 int värden skall allokeras:

$$int\ ^{*}buffer;$$
$$buffer = (int *)\ malloc\big(10 * sizeof(int)\big);$$

I C måste man själv frigöra denna minnesarea när man inte längre ska använda den, annars uppstår minnesläckor som kan leda till att programmet kraschar. Ett exempel på hur man frigör allokerat minne ses nedan:

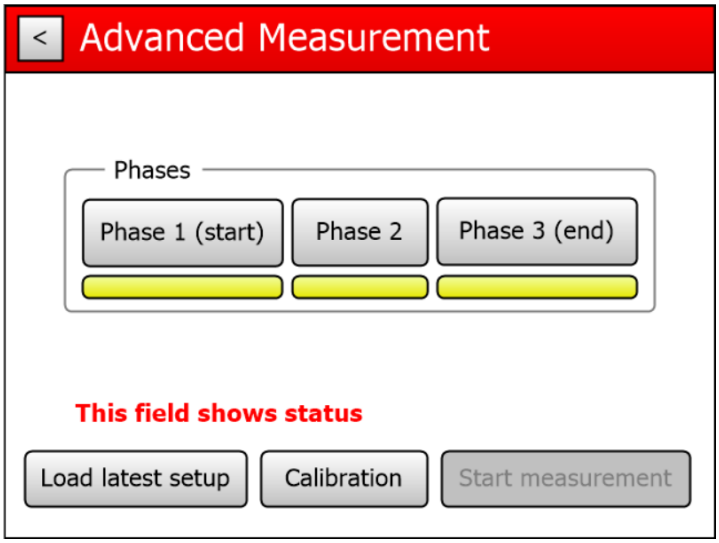
$$free(buffer);$$

4.4 Mätning och inställningar

Nedan beskrivs hur man skapar en mätinställning och hur en mätning utförs.

4.4.1 Översiktsskärmen

En mätinställning består av tre olika faser och en kalibreringskonfiguration. Faserna är uppbyggda av olika villkor och en lista av utportar att aktivera vid dessa uppfyllda fasvillkor. Faserna kan endast aktiveras i ordningen Fas 1, Fas 2 och slutligen Fas 3. Fasavancering kan endast ske genom att den aktiva fasens villkor uppfylls. Viktigt att komma ihåg är att det endast krävs att ett villkor uppfylls inom en fas för att en fasavancering ska ske.



Figur 4.2. överblick för huvudskärmen för en mätningsinställning

Figur 4.2 visar hur användargränssnittet för att ställa in en mätning ser ut. Genom att klicka på de olika faserna (delmomenten) navigeras man till de enskilda fasinställningarna. Indikatorfärgerna nedanför visar nuvarande status för respektive fas. Statusen varierar mellan 'Ej inställd fas (gul), 'Godkänd fas (grön) och 'Felaktig fas (röd). Om en fas saknar villkor kommer den att indikeras som en felaktig fas, eftersom mätningen aldrig kommer att avancera förbi denna fas. En varningstext kommer att visas i statusfältet. En mätning kommer aldrig att kunna startas utan att alla faser är godkända.

Man kan även välja att ladda den senaste godkända mätningsinställningar med knappen "Load latest setup". Då kommer gränssnittet att uppdateras och alla faser att uppdateras med senaste godkända mätinställningen. Mätningsinställningen lagras i flashminnet när man startar en mätning, det innebär att inställningarna även finns kvar efter en omstart av systemet.

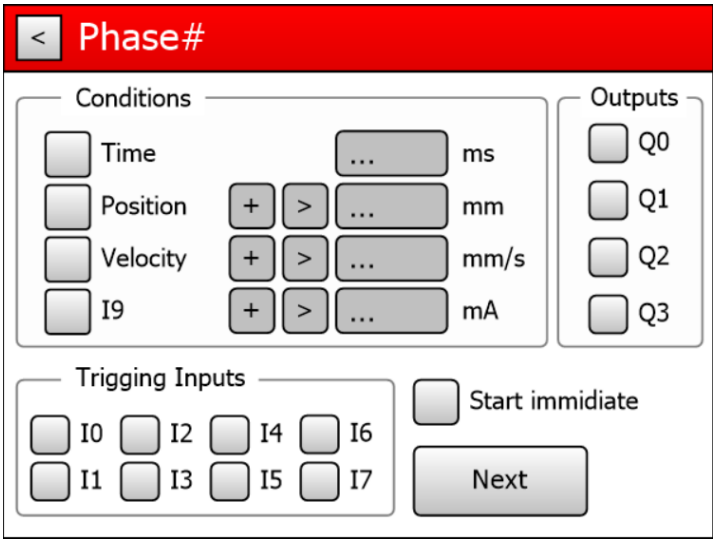
4.4.2 Fasinställning

Startfasen (Fas 1) innehåller villkor som krävs för att en mätning ska starta. Data som hanteras och kontrolleras innan denna fas uppfylls kommer alltså inte att sparas i mätningen.

Fas 2 används vanligtvis endast för att invänta en stoppsignal eller aktivera en utport, men den kan även ställas in precis som de andra faserna.

Slutfasen (Fas 3) är den fas som avslutar mätningen, efter att fasvillkoren har uppfyllts slutar mätningen att sampla värden och initierar istället grafritningen för en visuell presentation av den insamlade data.

Användargränssnittet som visas nedanför används för att ställa in alla villkor och aktiva utsignaler för en fas.



Figur 4.3. Överblicksbild för fasinställningarna

Villkor

Genom att kryssa i de olika kryssrutorna under ”Conditions” aktiverar man respektive villkor och får även möjligheten att ställa in dem. För att ange ett tröskelvärde för exempelvis position, så klickar man i textrutan till höger om ”Position”-etiketten och man kommer till en tangentbordsskärm där inmatning av ett värde sker.

Plustecknet indikerar att en positiv förändringshastighet hos det värdet krävs för att villkoret ska uppfyllas. Genom att klicka plustecknet byts det ut mot ett minustecken, där en negativ förändringshastighet hos värdet krävs för att uppfylla villkoret.

Storlekstecknet är inställt med ”Större än” som standard, detta innebär att alla värden större än det angivna tröskelvärdet kommer att uppfylla villkoret. Genom att klicka på ”Större än”-tecknet byts det ut mot ett ”Mindre än”-tecken, som betyder att alla värden mindre än det angivna värdet kommer att uppfylla villkoret.

Triggande Ingångar

Man kan även använda olika insignaler för att åstadkomma en fasavancering. Genom att kryssa i de olika kryssrutorna under ”Trigging inputs” så aktiveras kontrollen av respektive insignal under en mätning.

Om någon insignal är konstruerad aktivt låg kan man ställa in detta under kalibreringsskärmen som beskrivs i kapitlet *”Kalibrering”*

Utgångar

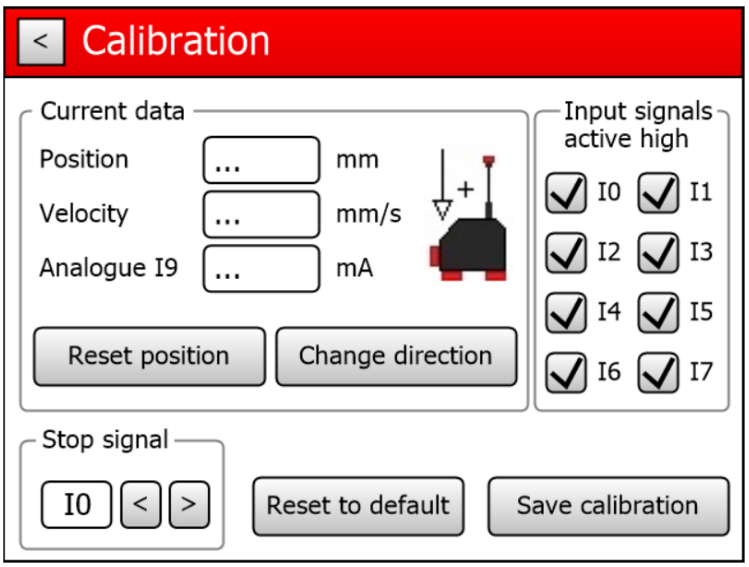
Systemet har fyra olika utgångar (Q0-Q3) som går att aktivera. Utgångarna kommer att aktiveras när en fasavancering sker från den aktiva fasen (OBS! Det krävs endast att ett fasvillkor uppfylls för att åstadkomma en fasavancering). Ett förekommande exempel är att flaggenheten för Smart-systemet (SM11) kopplas in på en switchande utgång och sedan aktiveras vid en fasavancering, detta kan då användas för att skapa en stoppsignal från en ljusridå.

'Start Immidiate'

Kryssrutan 'Start immidiate' avaktiverar alla andra villkor och kommer att genomföra en fasavancering direkt. Gränssnittet kommer att uppdateras och gråmarkera alla andra villkor för att indikera (för användaren) att de andra villkoren inte kommer att testas. Aktiva utsignaler kommer fortfarande att aktiveras som vanligt.

4.5 Kalibrering

Följande stycke beskriver de olika delarna som finns under kalibreringsskärmen. Figur 4.4 visar gränssnittet för kalibreringsskärmen.



Figur 4.4. Överblicksbild för kalibrering av mätningen

”Current data”

Under denna del av gränssnittet kan man se aktuell data från QEI-modulen och den analoga ingången.

Genom att klicka på ”Reset position” nollställer man QEI-modulens positionsräknare så att den räknar relativt till den nya positionen.

”Change direction”-knappen byter riktningen på räknaren, bilden ovanför indikerar vilken dragriktning på vajern som kommer att räkna positionen positivt. Detta skapar fler möjligheter för användaren att fästa pulsgivaren på.

”Stop signal”

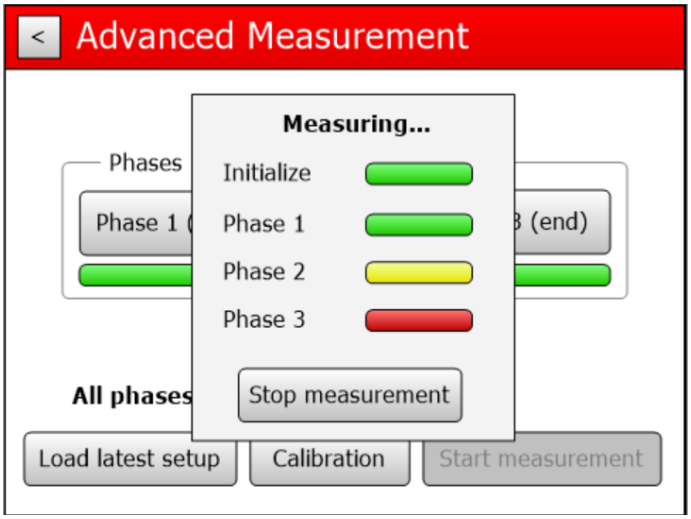
Under detta fältet kan man välja vilken ingång som kommer att agera som stoppsignal för mätningen. Denna signal kommer sedan att visualiseras i grafritningen för att indikera när stoppsignalen angavs, den blir startpunkten för stopptidsmätningen.

”Input signals active high”

Under denna del av gränssnitten har man möjligheten att ställa de insignaler som är aktivt höga. Om en signal är aktivt låg behöver den avkryssas, förutsatt att den används som en triggande ingång i någon av de inställde faserna, annars kommer den att ignoreras.

4.6 Mätning

För att starta en mätning måste alla faser vara godkända och en kalibrering bör vara inställd, sedan trycker man på ”Start measurement”. En popup-ruta visas enligt nedan.



Figur 4.5. Popup skärm som visas under mätningens gång

Figur 4.5 visar hur gränssnittet ser ut under en mätning. En stor förändring från det äldre Smart-systemet och mjukvaran på PC är att man nu får feedback direkt under mätningen. Detta efterfrågades i marknadsundersökningen som utfördes. Om man har ställt in faserna felaktigt kan mätningen fastna under faserna, till exempel genom att man har ställt in omöjliga villkor som aldrig uppfylls. Då kan man enkelt se vilken fas det är som skapar problem och sedan omkonfigurera fasen korrekt. Grön indikator indikerar en avklarad fas. Den aktiva fasen indikeras med det gula fältet, och en icke påbörjad fas har färgen röd.

En initieringsfas har implementerats för att indikera status innan mätning påbörjas. Under initieringsfasen nollställs flash-bufferten för att göra plats för en ny mätning, man startar även en timer som sköter samplingen. Avbrottsintevallet för timern ställs in för att motsvara det valbara samplingsintervallet för mätningen. Inuti avbrottsrutinen samlas och sparas all data samt kontrolleras ifall villkor för en fasavancering är uppfyllda. Varje sampel sparas till flash-bufferten i följd. Mätningen kommer att avslutas ifall flash-bufferten blir fylld.

Tabell 4.1 visar uträkning för maximalt antal sampel och den maximala mätningstiden för varje tillgängligt samplingsintervall.

FLASH_BUFFER_START	*0x010000*
FLASH_BUFFER_END	*0x0FFFD9*
Tillgängliga platser	*(0x0FFD9 – 0x01000) = 0xEFFD9*

Tabell 4.1. Maximalt antal sampel för flashbuffer

Antal tillgängliga bytes: 0xEFFD9 bytes = 983001 bytes

Ett sampel består av 9 bytes, detta ger maximalt 109222st sampel.

Samplingsintervall	***Maximal mätningstid***
0,1 ms	*10,92 sec*
1 ms	*109,22 sec*
10 ms	*18,2 min*
100 ms	*182 min*
1000 ms	*30,4h*

Tabell 4.2. Maximal mätningstid baserat på maximalt antal sampel och olika samplingsintervall

Den insamlade datan består av position och hastighet från QEI-modulen, värden från AD-omvandlaren, aktiva insignaler och värden på aktiva utsignaler. AD-omvandlaren och I/O-portarna var utanför ramen för detta projetekt, för tillfället sparas pseudo-data för dessa variabler

I slutet av mätningen sparas övrig information som är viktig vid grafritningen, informationen består av samplingsintervall, antal sampel, aktuell stoppsignal, min och max värden m.m. När den informationen är sparad i headern är allt klart och systemet börjar initiera grafen.

4.7 Visualisering av mätdata

För att kunna kontrollera ett en mätning gått som man tänkt sig behövdes ett sätt för att verifiera detta. I den första versionen av Smart fick man som användare, efter en mätning var utförd, reda på vilken stopptid maskinen hade, samt vilken hastighet den hade när stoppsignalen gavs. Detta vidareutvecklades sedan till den nuvarande versionen av Smart, där ett mjukvaruprogram (Smart Manager) används för att bestämma stopptid samt analysera positions- och hastighetskurvor för mätningen.

Denna möjlighet, att kunna presentera mätdata i en graf, var en av punkterna på kravspecifikationen. Nedan presenteras hur uppgiften lösts samt de problem som uppstått under utvecklingen.

4.7.1 Utformning av grafen

Efter att ha sökt efter grafverktyg till det utvecklingskortet som används utan resultat insågs att ett eget grafverktyg måste implementeras. Den första frågeställningen var även här, hur

skall utformingen av grafen se ut? Vilka funktioner ska implementeras och vilken data ska visas? Upplösningen på displayen är endast 320x240 pixlar och detta medför stora problem när en stor mängd information ska visas.

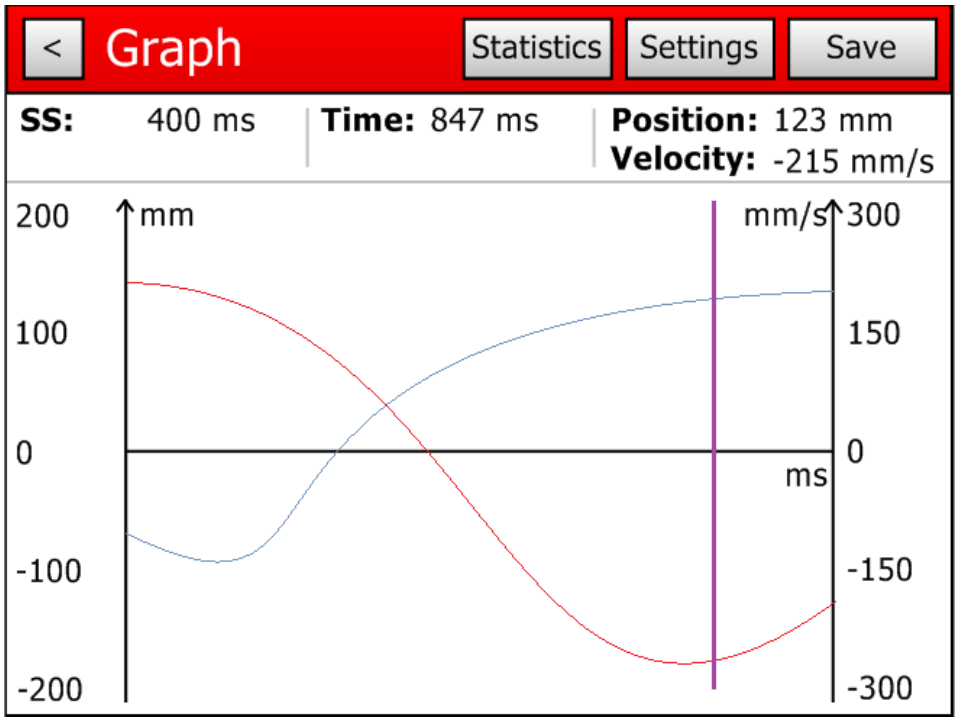
Det första som gjordes var att bestämma vad som skulle visas och var det skulle visas. Eftersom denna del av programmet är den som visar mest information på en och samma skärm valdes att använda actionbaren för att ge användaren tillgång till olika alternativ som ”Inställningar” och ”Spara mätning”, se figur 4.6. Som i fallet med filhanteraren har även här implementerats en så kallad ”toolbar” precis under actionbaren. Skillnaden är att i detta fall används den inte för att ge användaren mer funktionalitet, utan för att visa tids- och positionsdata från grafen. Under toolbaren, i mitten, placerades sedan själva grafen och på dess sidor visas skalor för de olika kurvorna som visas.

Det finns framförallt några saker man tittar extra noga på efter att en mätning är utförd, när stoppsignalen gavs samt hur maskinens position och hastighet påverkades utifrån detta. Genom att analysera dessa faktorer ska användaren snabbt kunna se om mätningen utförts på ett korrekt sätt.

Man behöver alltså kunna urskilja vilken av kurvorna i grafen som representerar vad, man behöver snabbt kunna se när stoppsignalen gavs från systemet. Slutligen behöver det finnas en indikation på när maskinen anses ha stannat.

Nedan presenteras de olika faktorer man enkelt bör kunna urskilja ur grafen.

- Position, blå kurva
- Hastighet, röd kurva
- Stoppsignal, orange flagga (visas ej i Figur.4.6)
- Stopp, grön flagga (visas ej i Figur.4.6)



Figur 4.6. Layout för grafen, den vertikala markören används för att visa värden för en specifik tidpunkt i grafen.

4.7.2 Skalning av mätdata

Vanligtvis sker mätningar med ett samplingsintervall på 1 ms, (kan även vara 0.1 ms, 10 ms, 100 ms och 1000 ms) detta gör att ett stort antal sampel kommer att ingå i varje mätning. Eftersom skärmstorleken inte är särskilt stor är det nödvändigt att skala om mätdatan för att grafen ska få plats på skärmen.

Nedan finns ett exempel som stegvis beskriver hur skalningen av mätdatan sker.

Exempel

Låt säga att en mätning har skett med samplingsintervallet 1 ms och pågått i exakt två sekunder och består av värden mellan -150 och +180. Då kommer det att finnas 2000 sampel i bufferten. Dessa 2000 sampel med data måste paketeras ihop för att få plats på grafytan som är 240x170 pixlar. Alltså måste en justering ske, både i x-och y-led.

X-led:

1. Beräkna hur många sampel som ska motsvara en pixel *(samplesPerPixel)*.
2. Iterera genom alla sampel och slå ihop *samplesPerPixel* antal paket för att bilda ett pixelpaket (*pixelPacket)*.
3. Under itereringen genom alla sampel, spara undan min och max värden.

Y-led:

1. Beräkna, utifrån *min* och *max,* vilka värden grafens y-axel ska ha. Grafen, görs alltid symmetrisk i Y-led. Vår graf hade fått yMin =-200 och yMax=200.
2. Beräkna förhållandet mellan grafens höjd i antal pixlar och differensen mellan *min* och *max (yRatio).*
3. Iterera genom alla *pixelPacket* och multiplicera dem med *yRatio*.

4.8 Lagring av mätdata

Den nya loggenheten är en förenklad version av den gamla, m.a.p analys av mätdata, och tanken är att man fortfarande kommer att använda mjukvaruprogrammet Smart Manager för att göra mer ingående analyser av mätningar. Med tanke på detta behövdes ett sätt att lagra mätningar för att senare kunna importera dem till Smart Manager. Nedan beskrivs hur mätningar lagras i filsystemet och vilka typer av data som lagras.

4.8.1 Lagring i FAT32

En viktig punkt som togs i beaktande när utvecklingen av funktionalitet för lagring av mätdata påbörjades var att den information som lagras måste vara förhållandevis lik den mätdata som idag skickas mellan det gamla Smart-systemets loggenhet och Smart Manager. Detta därför att man på ett enkelt sätt, utan att förändra allt för mycket i Smart Manager, skall kunna importera dessa filer och sedan visa dess grafer.

För att kommunicera med lagringsenheten användes hjälpbiblioteket FAT32_Library. Detta bibliotek har tillgång till funktioner för att skriva data till filsystemet så att det bildar en fil. Men eftersom detta bibliotek är väldigt generellt och endast ger användaren tillgång att skriva data i form av *char* arrayer till systemet bestämdes att egna drivrutiner behövde skapas för att lagra den specifika data som mätningarna innehåller. Nedan ses de två funktioner som används i den nya Smart-loggenheten för att skapa en fil.

- **MMC_WriteHeader(Header *h)**
  Skriver header data till filen. Datan består av bl.a. samplingsintervall, antal sampel etc. Se Bilaga A för fullständig uppbyggnad.

- **MMC_WriteSample(Sample *s)**
  Skriver sampeldata till filen, Datan består av bl.a. positionsvärde, hastighetsvärde och diverse in/ut signaler. Se Bilaga A för fullständig uppbyggnad.

5. Resultat

Syftet med detta projekt var att undersöka huruvida man kan skapa en mer pålitlig och stabilare produkt än den som i dagsläget används som Smartsystemets loggenhet. Målet var att påbörja vidareutvecklingen av denna enhet så att den uppfyller kundernas nuvarande krav.

Syftet med projektet har uppnåtts då vi undersökt och kommit fram till att man kan skapa en pålitligare och stabilare loggenhet till Smartsystemet. Detta är dock inget som fullt ut kunnat färdigställas pga. tidsbrist och att projektets mål endast varit att påbörja vidareutvecklingen av produkten.

För att kunna uppfylla målet om att Smartsystemets loggenhet skulle möta kundernas nuvarande krav, gjordes en marknadsundersökning i början av projektet. Denna marknadsundersökning visade sig vara mycket givande och gav en bra bild över vilka delar kunderna prioriterade. Marknadsundersökning lades fram till de ansvariga för utvecklingen av den nya prototypen som tog fram den kravspecifikation som arbetet utgått ifrån.

5.1 Prototypen

Utveklingen har resulterat i en prototyp bestående av ett utvecklingskort från MikroElektronika som heter *”mikromedia for dsPIC33EP”* samt en applikation. Denna prototyp, som inte behöver vara inkopplad mot en PC för att fungera, klarar av att mäta och visuellt presentera positions- och hastighetsdata genom att använda sig av de tillhörande pulsgivarna SM5/1250 och SM5/2500 från ABB Jokab Safety.

Applikationen besår av fyra sektioner som var för sig ansvarar för en specifik del av applikationen. Dessa sektioner är i princip helt fristående från varandra.

Den första sektionen ansvarar för hur en mätning skall genomföras och vilka villkor som måste uppfyllas för att mätningen ska kunna påbörjas. För att ställa in dessa villkor används tre olika faser, startfasen, mittfasen och slutfasen. De villkor vi implementerat inom ramen för detta projekt är villkor för tid, position och hastighet. När alla faser är ifyllda kan en mätning påbörjas.

Den andra sektionen ansvarar för presentation, statistik och analys av den mätdata som insamlats. Sektionen består av en huvudskärm och två popups. Huvudskärmens uppgift är att presentera mätdatan i form av en graf och ge information om position, hastighet och stoppsignal. Den första av de två popups som tillhör denna sektion visar mer ingående statistik och viss analys över hur mätningen gått. Det presenteras t.ex. min och max värden för position och hastighet samt statistik på hur lång stoppsträcka maskinen hade under mätningen. Den andra popupen är till för att ställa in vilka av graferna och flaggorna som ska visas.

Den tredje sektionen har hand om lagring av mätdata. Den består av en förenklad version av den filhanterare som implementerats samt drivrutiner mot ”FAT32_Library”-biblioteket för att förenkla lagringen av mätdata. Skillnaden mellan denna version av filhanteraren och den som diskuteras i kapitel 4.3 är att man endast har tillgång till mapphantering.

Den fjärde och sista sektionen av applikationen är filhanteraren. Filhanteraren låter användaren skapa, redigera eller ta bort filer och mappar i systemet. Här tillåts inte att man skapar mappar i mappar, utan den implementerade filhanteraren kan bara vara två nivåer djup. Genom att markera en mapp får man tillgång till alla dess filer och kan därifrån göra operationer på dessa. Härifrån man kan också välja att ladda in en gammal fil med mätdata till grafen.

6. Slutsats

Med allt högre krav på säkerhet och pålitlighet för maskiner är det viktigt att det finns bra utrustning för att säkerställa detta. Detta är den största anledningen till att ABB Jokab Safety vill utveckla en ny version av loggenheten till deras Smartsystem. Frågan man kan ställa sig när en produkt som ansvarar för människors säkerhet skall utvecklas är, hur mycket pengar skall ett företag lägga på produktens utveckling för att man ska kunna erbjuda sina anställda maximal trygghet i sin arbetsmiljö? Industrisäkerhet är en mycket aktuell fråga i dagens industrisamhälle, där tekniken utvecklas snabbt och säkerheten kring tekniken måste utvecklas i lika snabb takt för att inte skapa problem. Nya standarder och kriterier formas hela tiden och den kan vara mycket kostsamt för företag att ständigt anpassa sig. Smartsystemet är en bra lösning för industriverksamheter att nästan helt på egen hand se till så att deras verksamhet uppfyller dagens krav för att kunna säkerhetsställa produktionen och därmed deras ekonomi utan större hinder. Smartsystemet är ingen budgetprodukt och därför kan den initiala engångskostnaden för mindre industriföretag slå hårt, men långsiktigt sparar verksamheterna dessa pengar i form av upptid och ersättningar från försäkringar och eventuella stämningar. Företag som använder sig av smartsystemet kommer att bidra till en effektivare och säkrare industrimiljö i vårat samhälle.

6.1 Resumé

Första delen av projektet gick mest ut på att sätta sig in i hur det befintliga Smartsystemet var uppbyggt och hur kunderna använder det i industrierna. Detta gjordes för att få en fördjupad kunskap som vi senare kunde använda i projektet. Under den första veckan, som en del av inlärningsfasen, fick vi göra en marknadsundersökning som innefattade samtal med användare av Smartsystemet runt om i världen. Dessa samtal ledde till en kravspecifikation som vi sedan kunde arbeta efter.

Efter att denna period var klar övergick våra arbetsuppgifter från att lära oss om hur systemet var uppbyggt till att planera projektet. Vi började med att, tillsammans med våra handledare på ABB Jokab Safety, gå igenom vilka delar av kravspecifikationen som skulle fokuseras på inom ramen för detta projekt. Efter att detta var klart började vi implementera de delar vi kommit fram till var de viktigaste. Dessa delar var:

- Göra enkla mätningar.
- Visa graf över mätningar.
- Lagra mätdata i filer.
- Bläddra bland lagrade filer.

Under de kommande veckorna ägnade vi större delen av vår tid till att implementera dessa delar. När dessa delar var implementerade valde vi att inte fortsätta utveckla mer nya större funktionaliteter, utan valde att förbättra de delar vi skapat. Detta gjorde vi därför att underlätta vidareutvecklingen av produkten. Istället för att lämna över en prototyp med

många funktioner som fungerar dugligt är det lättare att fortsätta utveckla en produkt som har få väl fungerande delar.

6.2 Kritisk diskussion

I början av projektet togs det fram en planeringsrapport med en tidsplan för att man enkelt skulle kunna se om man låg bra till tidsmässigt eller ej. Denna planeringsrapport som såg bra ut från början visade sig under arbetets gång inte vara särskilt värdefull då den innefattade fel typ av information. I Ganttschemat, som kan hittas i appendix C, ser man att vi försökte dela upp arbetet utifrån olika typer av områden, Grafisk Layout ena veckan för att veckan efter jobba med signalbehandling osv. Men nu i efterhand inser vi att om vi skulle vetat det vi vet nu hade planeringsrapporten varit utformad på ett helt annat sätt. Till en början hade inte de olika tidsperioderna varit indelade i olika typer av uppgifter utan hade istället varit uppdelade i olika funktionaliteter som finns i programmet. Applikationen består idag av fyra stycken sektioner (Mätning, Visning, Lagring och Filhantering), dessa hade varit mycket lämpligare att använda.

Under projektets gång har vi också insett vilket stort misstag det varit att inte använda någon form av versionshantering för att hålla koden uppdaterad. Eftersom vi bara varit två stycken som arbetat med detta projekt tänkte vi från början att vi håller oss till olika uppgifter så att vi inte arbetar på samma ställe i koden och att vi på så sätt skulle slippa sätta upp någon form av versionshantering. Detta var väldigt naivt då vi redan tidigt i projektet insåg att detta inte var hållbart. Men istället för att lägga ner några dagar på att få igång t.ex. GIT valde vi istället att fortsätta koda på varsitt håll för att cirka en gång i veckan sammanfoga dessa delar.

6.3 Vidareutveckling

Eftersom detta projektets syfte och mål var att påbörja vidareutvecklingen av det befintliga Smartsystemet finns fortfarande mycket kvar att göra innan det är klart. Det som fokuserades på inom ramen för projektet var de mest nödvändiga delarna, som att göra en mätning, lagra en mätning, filhantering osv. Dessa delar valdes ut i samråd med de ansvariga på ABB Jokab Safety så att de senare har en bra grund att fortsätta utvecklingen ifrån. Några av de delar som ännu inte är klara är:

- Göra mätningar baserade på fler villkor(i.e. analog insignal, digitala insignaler) och inte bara tid, position och hastighet.
- I dagsläget kan man göra s.k. avancerade mätningar, en funktionalitet för att göra enklare mätningar finns kvar att implementera.
- Spara mätinställningar.
- Realtidsklocka är inte implementerad.

- Språkhantering.
- Exportering av filer till andra system.

Det finns mycket kvar att implementera innan man har en färdig produkt att lansera på marknaden, men det vi åstadkommit under projektet har skapat en bra grund att vidareutveckla ifrån.

7. Referenser

[1] ABB
http://www.abb.com/
[Tillgänglig: Maj 30, 2014]

[2] Smart
http://www.abb.se/product/seitp329/56f55ef0cbbd0eadc1257982003a9db1.aspx?productLanguage=se&country=SE
[Tillgänglig: Maj 30, 2014]

[3] Scrum
https://www.scrum.org/
[Tillgänglig: Maj 28, 2014]

[4] Kanban
http://kanbanblog.com/
[Tillgänglig: Maj 28, 2014]

[5] MikroElektronika
http://www.mikroe.com/
[Tillgänglig: Maj 28, 2014]

[6] Libstock
http://www.libstock.com/
[Tillgänglig: Maj 30, 2014]

[7] File Allocation Table
http://en.wikipedia.org/wiki/File_Allocation_Table
[Tillgänglig: Maj 05, 2014]

[8] Brien M. Posey, ”Choosing between file systems ”
http://technet.microsoft.com/en-us/library/cc750354.aspx
[Tillgänglig: Maj 05, 2014]

[9] Libstock, FAT32 Library
http://www.libstock.com/projects/view/108/fat32-library
[Tillgänglig: Maj 23, 2014]

Bilaga A. Sammanfattning av muntlig kravspecifikation för nya Smart Logger

Hårdvara

In och utgångar:	2-4 utgångar, 4-8 ingångar.
Encoderingång.	1 st, minst 500kHz, Quadratur (inkrementell)
Samplingsintervall	0,1ms, 1ms, 10ms, 100ms, 1000ms, (ev. 60000ms).
Analogingång.	1 st, 12 bitar, mjukvarukalibrerad, 0-20mA.
Drifttid.	Minst 8 timmar kontinuerlig drift. Batterihantering (laddstatus och strömsparläge mm.) Realtidsklocka (minst 1 månads drifttid). Laddtid (2-4h).
Kalibrering.	AD-omvandlare, touchskärm, batteri, realtidsklockan.

Mjukvara

Enkelt läge (som den gamla mätaren JSSM1), d.v.s. fasta villkor.

Använda SD-kort för lagring av mätdata. FAT32 som filsystem.

Internt flashminne som buffert för en mätning.

Lättbegripligt användarinterface.

Visa graf efter utförd mätning med begränsade zoom-möjligheter.

Möjlig bläddring och visning av sparade mätningar.

Inmatning av information skall kunna ske med fingrar och stylus-penna.

Numeriskt och QWERTY-tangentbord.

Klara hantering av realtidsklocka. Ställs in vid sparande av mätning vid behov.

Funktionalitet liknade Smart Manager.

Kommunikationsmöjligheter

USB för lagring av data och programvaruuppdatering.

Ev. Trådlöst (Bluetooth2) för fjärrstyrd mätning och dataöverföring

Bilaga B. Fil med mätdata

File: Består av en Header och *n* Values

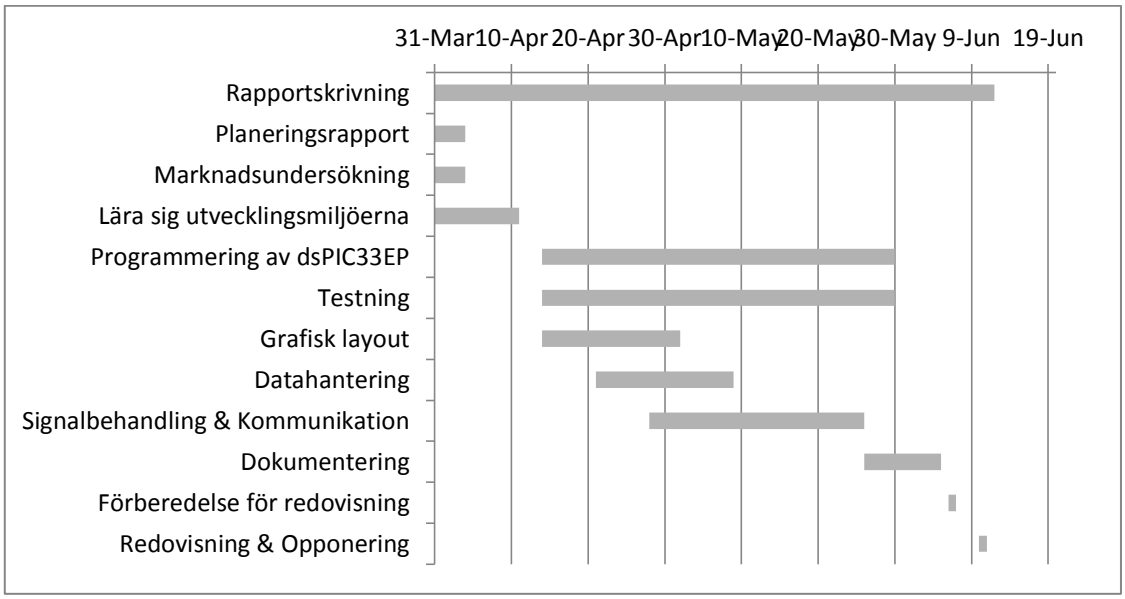
Header{

sampleIntervall	:	16 bits
nrSamples	:	32 bits
stopSignal	:	8 bits
stopSignalTime	:	32 bits
stop2mm	:	32 bits
stop0mms	:	32 bits
stop5mms	:	32 bits
stop10mms	:	32 bits
maxPosition	:	32 bits
minPosition	:	32 bits
maxVelocity	:	32 bits
minVelocity	:	32 bits

}

Values {

position	:	24 bits
velocity	:	16 bits
i9	:	12 bits
i7	:	1 bit
i6	:	1 bit
i5	:	1 bit
i4	:	1 bit
i3	:	1 bit
i2	:	1 bit
i1	:	1 bit
i0	:	1 bit
q3	:	1 bit
q2	:	1 bit
q1	:	1 bit
q0	:	1 bit
checksum	:	1 bit

}

Bilaga C. Planeringsrapport, Ganttschema