CHALMERS

Demonstrationsrigg för Systemutvecklingsverktyget QMAX

Viskositetsmätning via Flexsensor, Arduino & QMAX

Examensarbete inom Data- och Informationsteknik

Oskar Runesson
Richard Gourieh

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2019

EXAMENSARBETE 19:06

Demonstrationsrigg för Systemutvecklingsverktyget QMAX

Viskositetsmätning via Flexsensor, Arduino & QMAX

OSKAR RUNESSON
RICHARD GOURIEH

Institutionen för Data- och Informationsteknik
CHALMERS TEKNISKA HÖGSKOLA
GÖTEBORGS UNIVERSITET
Göteborg 2019

Demonstrationsrigg för systemutvecklingsverktyget QMAX
Viskositetsmätning via Flexsensor, Arduino & QMAX
OSKAR RUNESSON
RICHARD GOURIEH

Handledare: Sakib Sistek
Examinator: Peter Lundin

EXAMENSARBETE 2019:06
Institutionen för Data- och Informationsteknik
Chalmers Tekniska Högskola / Göteborgs Universitet
412 96 Göteborg
Telefon: 031-772 1000

Institutionen för Data- och Informationsteknik
Göteborg 2019

SAMMANFATTNING

Konsultföretaget ADDQ äger rättigheterna till en mjukvaruprodukt vid namn QMAX. QMAX är en mjukvara skapad för att underlätta vid R&D-projekt, där man har möjlighet att på ett enkelt sätt samla sensorvärden, styra elmotorer och skapa grafer för dessa. För att få fart på försäljningen på QMAX har ADDQ beslutat att ett demonstrerings-system skall konstrueras. Demonstrationssystemet skall bestå av en automatiserad elvisp som kan visas upp på möten med framtida kunder och potentiellt öka chanserna för köp. Tidigare har National Instruments CompactRIO eller PXI System använts som hårdvaruinterface mot de system som utvecklats. Ett sådant hårdvaru-interface kostar mellan 50- och 100 000 SEK.

Rapporten beskriver ett projekt för att se på möjligheterna att implementera ett billigare alternativ som ersätter CompactRIO systemet för att sköta sensoravläsning och motorstyrning av ett demonstrationssystem. Det utvecklade systemet använder sig istället av en Arduino Due som hårdvaruinterface för QMAX. Genom användandet av en flexsensor kan systemet göra enklare viskositetsmätningar på vätskan i systemet.

Nyckelord: QMAX, Arduino, CompactRIO

ABSTRACT

Consulting company ADDQ owns the rights to a software product called QMAX. QMAX is a software created for R&D, where you have the opportunity to in a simple way collect sensor values, control electric motors and create graphs of these. To boost sales of QMAX, ADDQ have decided that a demonstration system for QMAX shall be constructed. The system was decided to be designed as an automated cream whipper and can in the future be presented at meetings with future customers and hypothetically increase the chances of future purchases from these customers. Historically, National Instruments CompactRIO or PXI Systems have been used as hardware interface towards the constructed systems. These systems costs in the range of 50- to 100 000 SEK.

This thesis discloses the possibilities to implement a cheaper alternative which replaces the CompactRIO system for sensor reading and motor control in a demonstration system. The developed system instead uses an Arduino Due as its hardware interface for QMAX. Through the use of a flexsensor simpler viscosity readings can be made on fluids in the system.

Keywords: QMAX, Arduino, CompactRIO

FÖRORD

Den här rapporten är en dokumentation för examensarbetet i högskoleingenjör programmet inom mekatronik. Rapporten är det slutgiltiga momentet inom utbildningen.

Detta examensarbete har utförts på ADDQ Göteborg under perioden februari 2019 till juni 2019. En dagbok användes för dokumentation och för att se hur man låg till tidsmässigt i projektplaneringen.

Vi vill tacka vår handledare Sakib Sistek för stödet och den fantastiska handledningen under projektets gång. Vi vill även rikta ett tack till examinator Peter Lundin för sina kloka tips och råd på rapporten, samt till Göran Hult för bra insikter vid val av elektriska komponenter.

Ett tack riktas även till ADDQ för möjligheten som gavs att kunna utföra projektet på, och till produktchefen och vår handledare på företaget Styrbjörn Ekman, som med sitt samarbete har gett oss ett fint bra underlag för hur man skall arbeta, och för att alltid finnas där till hands. Vi vill även tacka Alixander Ansari som var vår fot in i företaget, för hans råd och tips, och att han alltid kunde erbjuda en hjälpande hand. Sist men inte minst skall ett stort tack riktas till företagets VD Fredrik Abrahamsson för att ha fått oss att känna oss välkomna sen första dagen.

Oskar Runesson & Richard Gourieh, Göteborg, Juni 2019

Innehållsförteckning

1

Inledning

ADDQ är ett konsultföretag som startades år 2005 och har nu 30 anställda i Göteborg och 50 anställda i Stockholm. Företaget specialiserar sig inom kvalitetssäkring där dom erbjuder systemutvecklande företag tjänster i form av vägledning, mätning, utveckling och anpassning för att säkerställa av system som utvecklats av företag. De kvalificerade mät- och testsystem som erbjuds, hjälper olika företag att kontrollera och säkerställa utvecklings-processer. Systemet som ADDQ använder för dessa tjänster heter QMAX.

QMAX är ett mjukvarusystem som hanterar mätningar och tester samt automatisering av olika processer. Systemet är baserat på mjukvaran LabVIEW. QMAX ger användaren en möjlighet att visualisera mätdata genom grafer. Man kan även styra hårdvara, som till exempel motorer och ställdon.

1.2 Bakgrund

ADDQ har sett svårigheter att få potentiella kunder att se värdet av QMAX. Testsystem som QMAX är svåra att förstå fördelen med. QMAX är ett digitalt verktyg som ger visualiseringar av värden/tillstånd för sensorer och motorer. Systemet är uppbyggt för att interagera med någon form av testsystem och utan ett sådant kan det vara svårt att förstå vad QMAX egentligen gör.

Vanligtvis paras QMAX ihop med National Instruments hårdvara för sensoravläsning och motorstyrning. Dessa är dyra och förhållandevis klumpiga, men fungerar väl ihop med QMAX. För att kunna möjliggöra enklare och billigare testsystem är det, enligt företaget, intressant att se huruvida QMAX kan arbeta och kommunicera med ett billigare och kompaktare hårdvaruinterface.

1.2 Syfte

För att ge kunder möjligheten att lättare förstå QMAX ska man försöka utveckla en fungerande demonstrationsrigg för QMAX.

För att ge möjligheten till billigare system i framtiden ska man även undersöka möjligheten att kombinera QMAX med ett billigare hårdvaruinterface än de som National Instruments erbjuder.

1.3 Mål

Projektets mål är att konstruera en fungerande demonstrationsrigg som styrs via QMAX. För att lyckas med detta krävs:

1. Ta fram en specifikation för demonstrationsriggen.
2. Besluta om vilken hårdvara som skall användas och montera ihop dessa.
3. Testa hårdvara mot QMAX.
4. Utveckla mjukvara för att styra och avläsa ingående komponenter.
5. Utveckla ett interface mellan hårdvara och QMAX.

1.4 Avgränsningar

• Vi skall ej utveckla hårdvara.

• Vi skall ej jobba med utvecklingsverktyg för QMAX.

• Vi skall ej utveckla några moduler i QMAX utan använda existerande.

2

Teknisk Bakgrund

2.1 Pulsbreddsmodulering

Pulsbreddmodulering (PWM) är en metod som används för att styra elektriska system, vanligtvis elmotorer. Genom att variera pulsbredden på en fyrkantsspänning med hög frekvens kan man styra olika delar av ett system som exempelvis varvtalet på en motor. Tack vare den höga frekvensen uppfattas spänningen som kontinuerlig.

För att styra PWM signalen behövs två parametrar. Frekvensen (pulsrepetitionsfrekvensen) är tiden mellan spänningspulserna och är konstant. Frekvensen väljs av användaren och bör vara så pass hög att apparaten som styrs uppfattar medelvärdet av signalen som en konstant. Pulsbredden är den del av spänningspulsen som är hög. För att öka eller minska effekten sätts därför pulsbredden till större respektive mindre andel av periodtiden.

I detta projekt används PWM tillsammans med en transistor för att styra varvtalet på motorn genom att öka eller minska effekten från batteriet.

2.2 LabVIEW

LabVIEW (Laboratory Virtual Instrument Engineering Workbench) är en grafisk programmeringsmiljö utvecklad av National Instruments [1]. LabVIEW tillämpar programmerings paradigmen ”G” som är en typ av dataflödesprogrammering. Detta betyder att ett program byggs upp med funktioner och variabler som i vilken programmeringsmiljö som helst, men då programmet exekveras kommer data att börja ”flöda”. Detta medför att flera funktioner kan behöva att utföras samtidigt. Därför har LabVIEW möjligheten att utföra flera funktioner parallellt. Då en funktions villkor är uppfyllda kommer denna att utföras, och dess utdata kommer skickas vidare [2].

LabVIEWs program är uppbyggda på flera subrutiner som kallas Virtual Instruments (VI). Ett VI innehåller en frontpanel och ett blockdiagram. Blockdiagrammet är själva koden, där hela programmet är uppbyggt. På frontpanelen finns kontroller och indikatorer som representerar det bakomliggande blockdiagrammet. Via kontrollerna kan man ändra variabler i programmet och på så sätt få olika resultat som representeras av indikatorerna.

2.3 QMAX

QMAX är en mjukvara skapad för forskning och utveckling. Den är baserad på LabVIEW och möjliggör datainsamling, loggning, testsekvensering samt testcellsautomatisering. QMAX gör det enkelt att samla in data och övervaka teststationer. QMAX är en skalbar applikation där kunder kan få sina tjänster anpassade efter behov [3].

Figur 2.1: En testsekvens i QMAX. Från [3]. Återgiven med tillstånd.

Ett QMAX system består av QMAX HMI (Human-Machine Interface) och QMAX Engine. QMAX HMI är den visuella delen av QMAX. Detta är vad en kund primärt använder sig av då det är här man kan ge sitt system kommandon och se sina mätdata i form av grafer och diagram. Det är även här man har möjlighet att skapa testsekvenser.

QMAX Engine är det som driver QMAX och körs alltid i bakgrunden i alla QMAX system. Det är QMAX Engine som sköter datainsamlingen och sekvensexekveringen. Det är Engine som behöver initieras med det system som ska övervakas. I de flesta system konfigureras Engine på en PC för att sedan installeras på ett realtidssystem, till exempel National Instruments CompactRIO eller PXI. För mindre applikationer kan den även köras på en PC.

2.4 ARDUINO

Arduino är ett projekt som startade upp på Ivrea Interaction Design Institute för att skapa programmerbara mikrokontrollerkort med ett okomplicerat programmeringsspråk samt ge studenter en enkel och billig lösning att lära sig programmeringsgrunder på [4].

Hårdvaran består av ett antal olika integrerade mikrokontrollers med stöd för digitala och analoga in och utgångar. Det finns en installerad Bootloader på mikrokontrollern som möjliggör nedladdning av utvecklat program till av chipets flashminne. För kopplingen mellan korten och PC använder sig Arduino av USB. Arduino har även en programmeringsmiljö kallad Arduino IDE. Både Arduinos hårdvara och mjukvara är open-source, vilket ger användare möjligheten att modifiera och konstruera egna versioner av deras kort.

2.5 Arduino IDE

Arduino IDE är Arduinos egna programmeringsmjukvara. IDE står för ”Integrated Development Environment” och är skrivet i JAVA och möjliggör programmering och kompilering av C och C++ kod [5]. IDE gör det även möjligt för användaren att ladda upp kod till kompatibel hårdvara. Primärt Arduinos egna processorkort, men även hårdvara med samma eller liknande konfiguration som Arduinos egna kort. Arduinos programmeringsspråk är baserat på projektet Wiring och IDE är baserat på projektet Processing [4]. Språket är enkelt konstruerat för att vara anpassat för nybörjare och Arduino tillhandahåller ett extensivt bibliotek med funktioner som gör programmering enkelt och intressant.

Figur 2.2: Ett nytt program i Arduino IDE

2.6 Arduino Due

Arduino Due är ett mikrokontrollerkort baserat på Atmels SAM3X8E CPU chip. SAM3X8E är en 32-bit ARM processor med en klockhastighet på 84 MHz och ett flash minne på 512 KBytes [6]. Arduino Due presterar därför avsevärt mycket bättre än andra Arduino kort. SAM3X8E opererar på 3.3V istället för 5V som de flesta andra processorer i Arduinos uppsättning. Därför finns vissa restriktioner i vilka tillbehör som fungerar ihop med kortet.

Arduino Dues in- och utgångsspecifikation är som följande [7]:

- 54 Digitala in- och utgångar, varav 12 har PWM kapabilitet.
- 12 Analoga ingångar med 12-bitars upplösning.
- 2 Analoga utgångar med 12-bitars upplösning
- 1 5V spänningsförsörjningsutgång med en maxström på 800 mA
- 1 3.3V spänningsförsörjningsutgång med en maxström på 800 mA
- 1 sexstifts SPI

2.7 Raspberry Pi

Raspberry Pi är en enkortsdator från Raspberry Pi Foundation. Kortet drivs av en ARM-processor. Datorn har inget inbyggt minne utan ett externt minneskort används istället som hårddisk för fillagring. Raspberry Pi är avsedd att köra operativsystemet Linux och det avsedda tillämpade operativsystemet för kortet är Raspbian, baserat på just Linux [8].

Raspberry Pi tillverkas i flera olika modeller. Vissa modeller använder sig av en USB adapter för internetanslutning medan andra har ethernet anslutningsport för nätverk. Det finns modeller med RAM minne på 256 MB, 512 MB och 1 GB. Processorer finns i form av enkärniga processorer med klockfrekvens på 700 Mhz, fyrkärniga processorer med klockfrekvens på 900Mhz alternativt 1.2Ghz. Vissa modeller har inbyggda Wifi och Bluetooth kort. Samtliga Raspberry Pi har ett 3,5mm video/ljud uttag. Vissa modeller har 26 digitala in/utgångar medan andra modeller har 40 pinnars I/O. Dessa används för egen programmering för styrning av bland annat dioder och reläer.

2.8 CompactRIO

CompactRIO är ett modulärt realtidssystem tillverkat av National Instruments. Ett CompactRIO system innehåller en processor, en Field-Programmable Gate Array (FPGA) modul och en eller flera input/output (I/O) moduler. Då dessa I/O moduler är utbytbara och FPGA är omprogrammerbar finns det möjligheter att omkonfigurera ett CompactRIO system [9]. Dessa system är utvecklade för att möjliggöra insamlande och processande av data. Då det finns I/O moduler för allt från digitala in- och utgångar till trådtöjnings-

givarmoduler kan man relativt enkelt sätta ihop ett system för just de specifikationer som man behöver för sitt projekt.

2.9 Borstad Likströms Motor

En borstad likströmsmotor är uppbyggd med fyra huvuddelar, en rotor, en stator, borstar och kommutatorer [10].

- Rotor är en elektromagnet med en eller flera lindningar som vid spänningstillslag skapar ett magnetiskt fält. Rotorn är fäst på elmotorns axel.
- Stator är en permanentmagnet som skapar ett permanent magnetfält runt rotorn
- Kommutatorerna är segmenterade kopparbrytare som var och en är i kontakt med en lindning.
- Borstarna är fasta kolstift som ligger i kontakt med kommutatorerna.

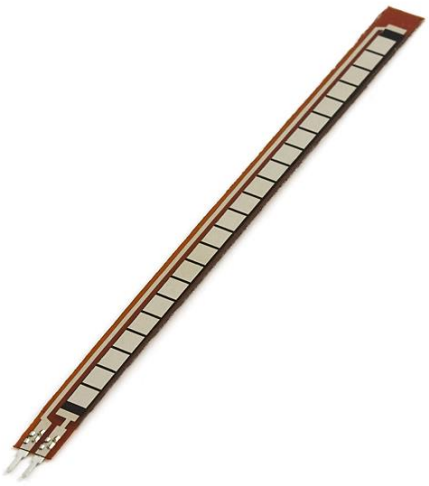
För att varvtalsreglera motorn reglerar man spänningen till motorn.

2.10 Halleffekt sensor

En halleffekt sensor möjliggör mätning av magnetiska fält. Om ström leds genom ett speciellt material, hallsensor, samtidigt som sensorn utsätts för ett magnetiskt fält kommer det att uppstå en spänning över sensorn vinkelrätt mot strömriktningen [11]. Spänningen är proportionell mot magnetfältets fältstyrka. Genom att mäta spänningen kan man avgöra magnetfältets fältstyrka.

2.10 Flexsensor

En flexsensor är en tunn remsa med ledande bläck vars resistans ökar då remsan böjs. Bläcket brukar innehålla kol för att bli ledande. Mellanrummet i kolet ökar då remsan böjs. Detta resulterar i att resistansen ökar i sensorn. För att kunna utläsa resistansen i sensorn anslutes sensorn i serie med en fast resistans varvid spänningsdelningen dem emellan ger sensorns resistans. Flexsensorn är konstruerad för att endast böjas åt ett håll [12].



Figur 2.3: Flexsensor. Från [13], CC-BY.

2.11 Läsgaffel

En läsgaffel är uppbygd med två sammansatta staplar. I den ena stapeln finns det en infraröd LED. I den andra stapeln finns det en fototransistor. En fototransistor fungerar som en transistor fast istället för basen, reagerar fototransistorn på infrarött ljus för att styra strömmen genom transistorn.

När den infraröda signalen passerar obehindrat från LED till transistor, öppnas transistorn och strömmen går genom komponenten. När det kommer något mellan staplarna så kan inte den infraröda signalen passera och det gör att det går någon ström genom fototransistorn.

I vårt fall använder vi läsgaffeln för att göra varvtalsmätningar. På den axel man vill mäta fäster man något för att blockera signalen mellan transistor och diod. Detta medför att läsgaffeln kommer ge en konstant signal till dess att signalen blir blockerad. Räknar man pulserna kan man få ut varvtalet.

Figur 1.4: Läsgaffel. Från [14], CC-BY.

2.12 Termometer DS18B20

DS18B20 är en digital termometer från Maxim Integrated. Den klarar av att mäta temperaturer mellan -50°C till 125°C, med en noggrannhet på en halv grad i spannet -10°C - +85°C [15]. För kommunikation använder sig termometern av Maxims 1-Wirebus. Detta ger användaren möjligheten att även använda termometerns datapin för strömförjning (parasite power). Varje 1-Wireenhet har ett unikt 62 bitars serienummer. Detta ger möjligheten att sätta flera termometrar på samma datalinje.

3

Metod

3.1 Förstudie och planering

Projektet kommer inledas med en förstudie för att ta fram förslag till riggen. Här kommer liknande projekt undersökas för att få inspiration för att få fram flertalet alternativ för riggen. Genom diskussioner med ADDQ kommer det beslutas om en slutgiltig specifikation för riggen.

För att få en klar bild över vad som ska göras kommer en planeringsfas genomföras. Där kommer så många arbetsmoment som möjligt identifieras och föras in i ett Gantt-schema. För att få en bra uppfattning över tidsåtgången för de olika momenten skall de jämföras med tidigare projekt.

3.2 Programmering

För att kunna genomföra projektet kommer det krävas en stor del programmering. Hårdvaruinterfacet kommer att kräva programmering för att läsa de sensorer som kommer att användas och även för att kommunicera med QMAX. Sedan kommer även QMAX att behöva programmeras för att co-operera med hårdvaruinterfacet.

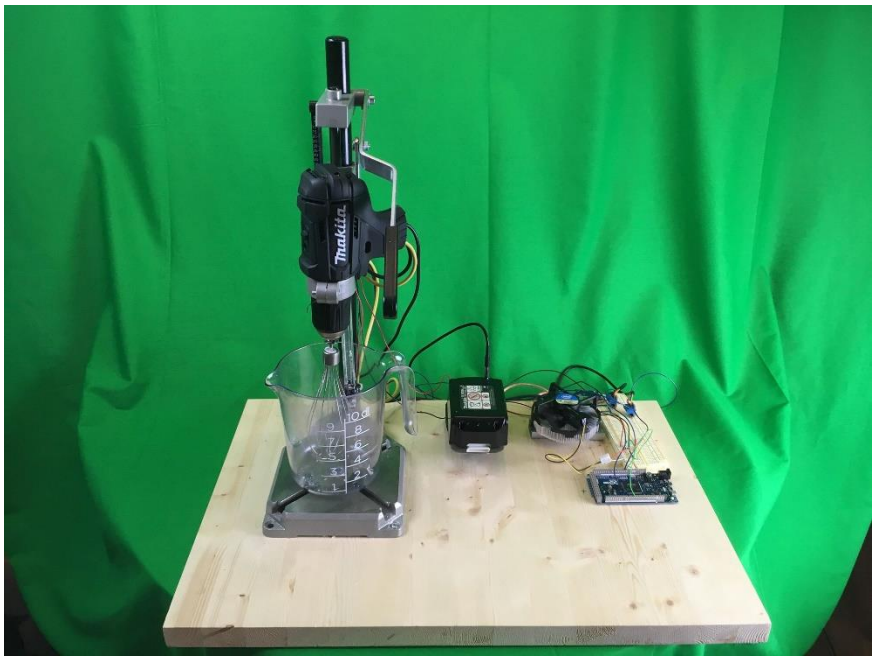
3.4 Tester av delsystem

Tester kommer att utföras för att se så att sensorer fungerar som tänkt. De kommer även utföras för att undersöka huruvida riggen opererar med den kapacitet som krävs. Laborationer ger ett direkt svar på om det som gjorts fungerar i praktiken så som det är tänkt.

4

Systemutveckling

Under detta projekt har vi tagit fram en demonstrationsrigg för systemutvecklings-verktyget QMAX.



Figur 4.1: Demonstrationsriggen

Riggen konstruerades i form av en batteridriven elvisp, med möjlighet att utföra viskositetsmätningar på fluiden som vispas. Systemet kan även mäta strömåtgång, varvtalet på motorn samt temperatur på motor och batteri. Då QMAX körs på en PC kan programmet i sig inte läsa av sensorvärden. Som interface mellan QMAX och hårdvaran användes en Arduino Due. Till Arduinon är alla sensorer kopplade. Flexsensor för viskositetsmätning, temperaturgivare, läsgaffel för varvtalsmätning, halleffekt sensor för mätning av strömmen samt transistor för motorstyrning. Per förfrågan från QMAX sammanställer Arduinon sensorvärden och skickar dessa till QMAX. För att ändra motorhastighet skickar QMAX en textsträng till Arduinon som i sin tur ändrar PWM signalen till transistorn. Detta kapitel handlar om hur vi tog fram denna rigg.

4.1 Förstudie och beslut om rigg

För att utveckla en demonstationsrigg som visade QMAXs kapacitet inleddes en förstudie. Vi började att undersöka de applikationer som QMAX användes till i dagsläget, samt tidigare demonstrationsriggar som ADDQ hade utvecklat till andra projekt. Vi diskuterade även med produktansvarig för att se vad han hade för förväntningar av riggen. Baserat på den insamlade informationen påbörjades arbetet med att ta fram förslag till riggen.

De kriterier som ställdes upp var:

- Ett godtyckligt antal sensorer för att visa upp QMAX möjligheter samt kunna utföra sekvenser av ett styrprogram.
- Ett portabelt system som en eller två personer kan bära med sig. Detta för att enklare kunna ta med riggen till kunder eller till mässor.
- Vara iögonfallande för att skapa intresse hos kunder och gärna innehålla något visuellt för att få folk att stanna.

För att få ytterligare inspiration söktes information på internet på riggar som kunde passa arbetet. Genom diskussioner togs tre alternativ fram. En hydraulpress, batteridriven elvisp och en tre dimensionell CNC maskin. För att få fram ett beslut rankade vi alternativen baserat på de kriterier vi ställt upp.

En hydraulpress erbjuder en spektakulär uppvisning, där man på kort tid kan göra relativt många körningar. Riggen har även möjligheter för flertalet sensorer och kan utföra sekvenser. Dock hade riggen blivit väldigt otymplig i sig, samtidigt som det hade behövts flertalet säkerhetsåtgärder för att undvika personskador. På grund av detta uteslöts denna rigg.

CNC maskinen är även den väldigt iögonfallande. Den hade heller inte nödvändigtvis behövt bli alltför otymplig och hela systemet bygger på förprogrammerade sekvenser. Dock är det relativt få sensorer och QMAX hade därför varit redundant.

Den batteridrivna elvispen uppfyllde alla kriterier. Den hade flertalet sensorer och kunde utföra sekvenser. Det var en relativt lätt rigg och en person kunde med enkelhet få med sig den. Den innefattande även en visuell del där man lätt kunde koppla ihop det som hände framför sig med det som QMAX skulle visa.

Beslut togs således om att konstruera en batteridriven elvisp. Riggen skall styras via QMAX med kapabilitet för att avläsa viskositeten i vätskan, samt strömförbrukning och temperatur i de olika delarna. Mellan QMAX och sensorerna behövs även ett hårdvaruinterface.

4.2 Komponentval

För att uppnå önskat resultat behövs ett antal komponenter. Detta avsnitt kommer handla om hur vi kom fram till de komponenter som skulle användas.

4.2.1 Hårdvaruinterface

För att avläsa sensorvärden behöver QMAX ett hårdvaruinterface. Då QMAX är byggt på LABView och ADDQ har ett samarbete med National Instruments har de tidigare riggarna med QMAX varit byggda med National Instruments hårdvarusystem CompactRIO. På grund av kostnaden på ett sådant system var inte det ett alternativ till denna riggen, istället började vi med att undersöka huruvida deras betydligt billigare myRIO-system kunde passa. Vi undersökte även huruvida vi skulle kunna bygga riggen på en Arduino eller en Raspberry Pi.

Då vi ansåg att myRIO var för dyr och då ett intresse för att se möjligheten att bygga riggar utan National Instruments hårdvara fanns, valdes denna bort. Arduinon var billigast och hade inbyggt stöd för analoga signaler. Raspberryn är ett mer kompetent system där vi hade förhoppningen om att kunna köra QMAX Engine fristående på detta system, dessvärre har den inget inbyggt stöd för analoga signaler. På grund av deras låga kostnad hade vi möjlighet att testa båda systemen. Då vi ej kunde få QMAX Engine att köras på Raspberryn på grund av kompabilitets fel mellan QMAX och Raspberryns ARM processor beslutade vi oss för att använda Arduinon då den även har stöd för fler komponenter.

4.2.2 Sensorer

Sensorerna har som uppgift att avläsa de systemvariabler som är intressanta. Dessa är:

- Viskositeten i vätskan
- Strömförbrukningen från motorn
- Temperaturen på batteri och motorn
- Varvtalet på motorn

4.2.2.1 Viskositet

Viskositeten var den svåraste variabeln att kontrollera. De mätsystem som finns är dyra, otympliga och inte kompatibla till den rigg vi tänkt bygga. Därför fick vi istället titta på variabler som förändras i takt med viskositeten.

Genom diskussioner kom vi fram till två olika variabler med fyra möjliga lösningar för att indirekt mäta viskositeten. Den första variabeln var motståndet i vispen. Då viskositeten ökar kommer det krävas större kraft för att vrida vispen. För att avläsa hur kraften på vispen förändras fann vi två lösningar. Då kraften ökar kommer även momentet på vispen öka. Genom att fästa trådtöjningsgivare på vispen skulle vi teoretiskt sätt kunna avläsa momentet i vispen. Alternativt skulle det ökade kraftbehovet öka strömåtgången i elmotorn och genom att avläsa strömmen skulle vi kunna se hur kraftbehovet förändras.

Den andra variabeln vi kom fram till var motståndet i vätskan. Den mest uppenbara lösningen var att genom att fästa en flexsensor på vispen kunde motståndet i vätskan avläsas. Vi kom även fram till att en flexsensor skulle kunna fästas vid sidan av vispen. När vätskan då träffar sensorn kommer kraften som uppstår vara proportionerlig mot viskositeten och hastigheten. Genom att ha konstant hastighet kommer viskositeten då kunna utläsas.

På grund av deras opraktiskhet valdes de lösningar med sensorer på vispen bort, även om dessa hade varit gynnsamma. Problemet är att sensorerna då hade suttit på en roterande axel och för att undvika trassel krävdes användandet av en släpring. Att få en släpring som klarade våra specifikationer var svårt. Ringen hade även blivit dyr och tagit lång tid att få levererad.

Gruppen beslutade att utveckla viskositetsmätning baserat på dels flexsensorn vid sidan av vispen samt strömförbrukningen på motorn. Den flexsensor som vi beslutade oss för att använda var en 4,5 tums flexsensor från företaget Spectrasymbol då den böjs lättare än en kortare vilket är gynnsamt vid mindre förändringar i viskositet. För strömmonitorering beslutade vi oss för att använda en hall-effect sensor.

4.2.2.2 Temperaturgivare

Som temperaturgivare hade vi två huvudalternativ, typ-K element eller en digital halvledarsensor typ DS18B20. Vi valde att använda DS18B20. Typ-K element är väldigt vanliga och har använts mycket tillsammans med QMAX. De har ett stort temperaturomfång och är förhållandevis enkla att använda. DS18B20 har ett mindre temperaturomfång men fullt tillräckligt för vår konstruktion. De är billigare och enklare att använda än Typ-K elementen. Den stora skillnaden mellan de båda var att med DS18B20 kunde vi koppla flera givare på samma datalinje vilket underlättar om flera temperaturgivare kopplas på.

4.2.2.3 Varvtalsmätning

Till varvtalsmätningen beslutade vi oss för att använda en läsgaffel. Det är en typ av sensor vi arbetat med tidigare och passade även bra in i detta projekt.

4.2.3 Elmotor och batteri

Till vispen behövdes en elmotor, för att göra riggen mer portabel skulle motorn drivas med ett batteri. Vi började med att titta på hur det skulle kunna lösas genom att köpa elmotor och batteri för sig. Problemet med denna lösning var att hitta batterier med tillräckligt hög kapacitet och snabb laddningstid, som även passade till kommersiella elmotorer. Alternativet var färdiga lösningar. En batteridriven skruvdragare innehöll allt det vi sökte.

Den skruvdragare vi beslutade oss för att använda levererades med två batterier med hög kapacitet och korta laddningstider. Detta innebar att man kan köra riggen en heldag genom att byta batterierna när de tar slut. Dessutom finns det ett färdig konstruerat sätt att fästa vispen. Elmotorn drivs på 18 Volt.

4.2.4 Motorstyrning

Elmotorn skulle drivas digitalt och Arduinon har stöd för PWM och därför beslutade vi oss för att använda den tillsammans med en transistor. Motorn drivs med ett 18 Volts batteri. För att ta reda på hur stor ström som transistorn behövde klara kopplade vi bort batteriet och drev motorn via ett spänningsaggregat där vi kunde avläsa strömstyrkan till motorn. Genom att ställa spänningen till 18 V kunde vi utläsa att motorn drog cirka 7 Ampere olastad. För att klara av spikar ville vi därför ha en transistor som klarar minst 20 A och 36 V.

Då Arduinon högst kan leverera 3.3 V behövs en transistor med låg restspänning bas-emitter samt hög strömförstärkningsfaktor för att ej belasta Arduinon för mycket. Den transistor som valdes uppfyllde alla krav och var dimensionerad för 80 Ampere och 55 Volt. Anledningen till att de motorstyrningskort som finns till Arduino valdes bort var att de kort som var allmänt tillgängliga inte klarade av att leverera tillräckligt stor effekt för att styra vår motor.

4.3 Riggkonstruktion

Då man vill ha en portabel rigg valdes en basplatta där alla delar enkelt kunde fästas. På grund av dess låga vikt och enkelhet att arbeta med valdes en basplatta av trä. För att fästa skruvdragaren vertikalt valdes ett pelarborrstativ ut. Detta medförde även att skruvdragaren enkelt kunde bytas ut vid problem. Denna fästes i basplattan via maskinskruv med mutter så den kan demonteras vid platsbrist.

För att ge mer plats till komponenter mellan batteri och motor kapades skruvdragare vid handtaget. Vi kunde därmed behålla den strukturella integriteten i skruvdragarens kåpa för elmotorn samtidigt som vi kunde använda batteristationen effektivare genom

Figur 2.2: Batteriplacering

att fästa den i basplattan. De styrdon som satt i skruvdragaren demonterades. På batteriet fästes en temperatur-givare.

Batteriets pluspol är således kopplad till motorn och motorns negativa pol är därefter kopplad till hall-effekt givaren för strömmätning. Från strömsensorn leds strömmen till kollektorbenet på transistorn. Då tester utfördes på transistorn bildades stora mängder värme då varvtalet reglerades ner. Detta orsakade att lödtennet som användes smälte. För att motverka detta monterades därför transistorn på en kasserad CPU-fläkt. Genom en FLIR-kamera såg vi temperaturen gå från cirka 150° Celsius före fläkten, till cirka 42° Celsius efter det att fläkten installerats. Emitter benet är därefter kopplad till nollan (minuspolen på batteriet). Basen på transistorn är kopplad till Arduinon. För att skydda systemet mot spänningsspikar vid frånslag i transistorn sitter även en frihjulsdiod mellan positiv och negativ pol på motorn.

För att få flexsensorn så nära vispen som möjligt fästes den på ett plaströr. Röret fästes sedan på pelarborrstativet. Positionen på röret erbjöd även en utmärkt plats för läsgaffeln att fästas vid. I höjd med läsgaffeln på vispen fästes en cirkulär 3D printad skiva som tagits fram.

4.4 Programmering av Arduino

Då vissa sensorer och transistorer tog längre tid att levereras än andra hade vi gott om tid att testa och provköra alla sensorer före konstruktionen av riggen påbörjades. Detta hjälpte oss få en djupare förståelse av de sensorer som vi beställt, samt en djupare förståelse av programmeringsspråket i Arduino IDE. Detta gjorde det även möjligt att producera fungerande program för varje sensor samt få fram fungerande kopplingar. När vi sedan kom till ett läge där det var dags att utveckla ett fullständigt program då riggen var färdigställd kunde vi med viss modifikation använda de tidigare programmen vi tagit fram för de olika sensorerna.

För att testa systemet i sin helhet byggdes ett relativt grundläggande program upp. Programmet skulle klara av att läsa alla sensorer och stanna motorn då önskat värde på flexsensorn hade uppnåtts. Det skulle även styra och beräkna varvtalet på motorn. Programmet byggdes som ett försök att vispa grädde och då grädden gick från flytande till fast skulle motorn stängas av.

Programmet är uppdelat i fyra huvuddelar:

- Deklarationer
- Setup
- Loop
- RPM funktioner

I början av programmet, i deklarationerna, är alla bibliotek och globala variabler definierade. Här är bland annat alla portar på Arduinon som används definierade.

I setupfunktionen börjar vi med att ställa in seriekommunikationens Baud Rate. I projektet används 9600 Baud som frekvens då det är den vanligaste frekvensen och de flesta system är kompatibla med den. I denna funktion definieras även huruvida de portar som används är ingångar alternativt utgångar. För att varvtalsmätningen skall fungera används även en avbrottsfunktion som även den definieras under setupen.

```
attachInterrupt(digitalPinToInterrupt(RPM_PIN), RPMCount, FALLING);
```

Figur 4.3: Avbrottsfunktion

De parametrar avbrottsfunktionen använder är den port som ska användas, den funktion som ska anropas och vid vilket fall som avbrottsfunktionen skall utlösas. I vårt fall används den port som är kopplad till läsgaffeln. Då läsgaffelns status ändras från hög till låg anropas funktionen RPMCount genom avbrottsfunktionen som ökar en räknare för varje gång den blir anropad.

Loopfunktionen är den delen av programmet som från dess att den påbörjats avslutas först då Arduinon stängs ner. I denna funktionen sker primärt datainsamling från sensorerna. Genom den seriella monitorn har man möjlighet att dels övervaka variabler men framför allt styra motorn. Motorn styrs via Arduinons PWM-signal. Genom att skriva ett tal 0 - 255 kan man då reglera hastigheten på motorn. Då Arduinon läser av ett viskositetsvärde via flexsensorn som motsvarar det värdet då grädden går från flytande till fast stänger den dessutom av elmotorn.

Utöver de tidigare nämnda funktionerna finns enbart de funktioner som styr varvtalsmätningen. Varav den ena är den tidigare nämnda räknaren som anropas av avbrottsfunktionen. Den andra är funktionen som beräknar varvtalet i varv per minut baserat på antal varv på en viss tid.

Då Arduinon senare enbart skulle fungera som en länk mellan QMAX och sensorerna lades ingen större vikt vid att utveckla detta program. Det var främst för att se att systemet fungerade som tänkt.

4.5 Implementering av QMAX

Efter att programmet i Arduinon fungerade korrekt påbörjades arbetet med att implementera QMAX som styrsystem. Då QMAX tidigare enbart använts ihop med National Instruments hårdvara fanns inget kommunikationsgränssnitt mot Arduino-produkter. Att lösa detta blev därför den första uppgiften.

4.5.1 Kommunikationsgränssnitt

Vår första tanke var att utveckla en komplett drivrutin till QMAX, samt ett standardiserat program till Arduinon där QMAX hade möjlighet att läsa alla portar på Arduinon och sedan kunde användaren genom QMAX välja vilka portar som skall användas. På grund av tidsbrist valdes detta alternativ bort. Istället valde vi att bygga ett mer specifikt

gränssnitt för vår rigg. Då seriell kommunikation fanns implementerad i både Arduinon och QMAX valde vi att basera gränssnittet på det. Implementerade i Arduinos programmeringsspråk finns flertalet funktioner för just seriell kommunikation då detta är det primära sättet för kommunikation med PC. Det var även en av anledningarna till att vi valde seriell kommunikation.

Kommunikationen sker genom en USB-kabel mellan den dator där QMAX körs och Arduinon. Genom kommunikationsporten på datorn skickar QMAX ett tecken. För att läsa tecknet på Arduinon används Serial.read funktionen. Genom att spara ner tecknet

```
serialData = Serial.read();   // Save data from COM Port
```

Figur 4.4: Serial.read funktionen på Arduinon

i en variabel kan Arduinon avgöra huruvida QMAX vill läsa eller skriva data. För att läsa data skickar QMAX ’r’ (för read), och för att skriva data skickar QMAX ett ’w’. Alla tecken som skickas mellan Arduino och QMAX är i teckenkodningen ASCII. Som tidigare nämnt sparar Arduinon detta tecken för att sedan undersöka vilket tecken som mottagits. Om Arduinon mottagit ett ’r’ skickas de aktuella sensorvärdena över till QMAX. För att QMAX ska kunna veta vilka tecken som tillhör vilken sensor skickas datan alltid i samma ordning och varje sensorvärde skiljs med ett ’|’-tecken. Tar Arduinon istället emot ett ’w’ gör den sig redo för att ta emot en sträng med tecken. På grund av begränsningar på Serial.read funktionen, den är enbart kapabel till att läsa in ett tecken åt gången behövdes ett alternativt sett att läsa motorhastighet eftersom vi ville ha större valmöjlighet i det anseendet. Valet föll på en funktion som läser en sträng med data från den seriella porten. QMAX skickar då en sträng med den motorhastighet, ett tal mellan 0 och 255, som användaren vill ha. Då det värde som skickas sätts till PWM-signalen behöver vara en Integer krävs det att Arduinon transformerar strängen från QMAX till en integer och sätter sedan PWM-signalen till detta värde vilket ger den motorhastighet som strängen representerat.

Då vi ville att systemet skulle vara så självgående som möjligt är QMAX programmerat till att automatiskt uppdatera sensorvärdena med en frekvens av 10 Hz när systemet körs. Om man vill byta motorhastighet, väljer man önskad hastighet och sedan sköter QMAX kommunikationen med Arduinon.

4.5.2 Prestandaförbättringar

Efter att systemet stod klart och QMAX var implementerat påbörjades testning. Då systemet körds uppvisar QMAX grafer på sensorvärden. Alla sensorer fungerade som tänkt utom halleffekt-sensorn. Strömvärdet som sensorn mäter tenderade till att antingen vara hög eller låg i intervaller som inte representerade hur systemet kördes. Efter undersökning kom vi fram till att eftersom motorstyrningen var PWM baserad och Arduinon endast läste ett värde på en given tidpunkt var det ganska logiskt att strömsensorn representerade det, istället för det medelvärde vi förväntat oss.

Problemet hade gått att lösa genom att medelvärdesbilda spänningen innan anslutningen till basen, exempelvis med ett lågpassfilter. Då vi var sent i projektet och ej hade

komponenter till detta valde vi istället en mjukvarulösning. Detta gjordes genom att en medelvärdesfunktion, även kallad en smoothingfunktion, programmerades in i Arduinon. Istället för att läsa ett värde från sensorn beräknar den medelvärdet på 1000 värden. Genom användandet av en indexerad array och en total summa kan man beräkna detta. Först subtraheras värdet i den indexerade minnesplatsen från summan, sedan läses ett nytt tal in till den minnesplatsen för att slutligen adderas till summan. Varje gång detta sker ökar indexvärdet. Då indexvärdet nått det antal värden vi vill beräkna medelvärdet på sätts räknaren åter till noll. Sist divideras den totala summan med antalet summerade tal.

```
currentTotal = currentTotal - currentReadings[readIndex];   // Subtraction of indexed value from the sum

currentReadings[readIndex] = analogRead(CURRENT_PIN);       // Read new value to the array

currentTotal = currentTotal + currentReadings[readIndex];   // Add new value to the sum

readIndex++;                                                // Increase the index

if(readIndex >= numReadings) {                              // When index reaches desired amount of readings set Index to 0
  readIndex = 0;
}

currentAverage = currentTotal / numReadings;                // Calculate Average by dividing sum with amount of readings
```

Figur 4.5: Medelvärdesfunktion

Tack vare Arduinons höga klockfrekvens kan detta utföras utan någon påvärkan på prestandan.

5

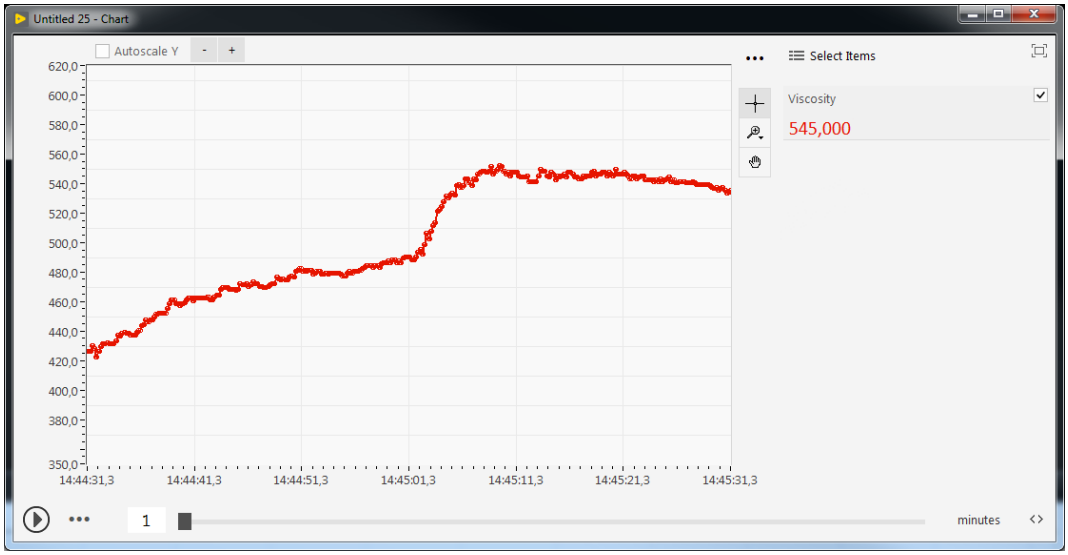
Resultat

En demonstrationsrigg för QMAX i form av en batteridriven elvisp har tagits fram. Genom att para ihop QMAX på en PC med en Arduino för sensoravläsning och PWM-styrning av motorer ger systemet användaren möjligheten att styra och övervaka vispen genom QMAX HMI. En flexsensor används för att utföra enklare viskositetsmätningar och en hall-effekt sensor för att mäta strömförbrukningen av elmotorn.

Efter användarens inmatning till QMAX skickas styrsignaler till Arduinon som i sin tur, beroende på vilken signal som mottagits, antingen svarar med sensorvärden, alternativt förändrar PWM-signalen för att öka eller sänka hastigheten på elmotorn. Grafer och mätare inbyggda i QMAX HMI hjälper användaren att se status på sensorer och motor.

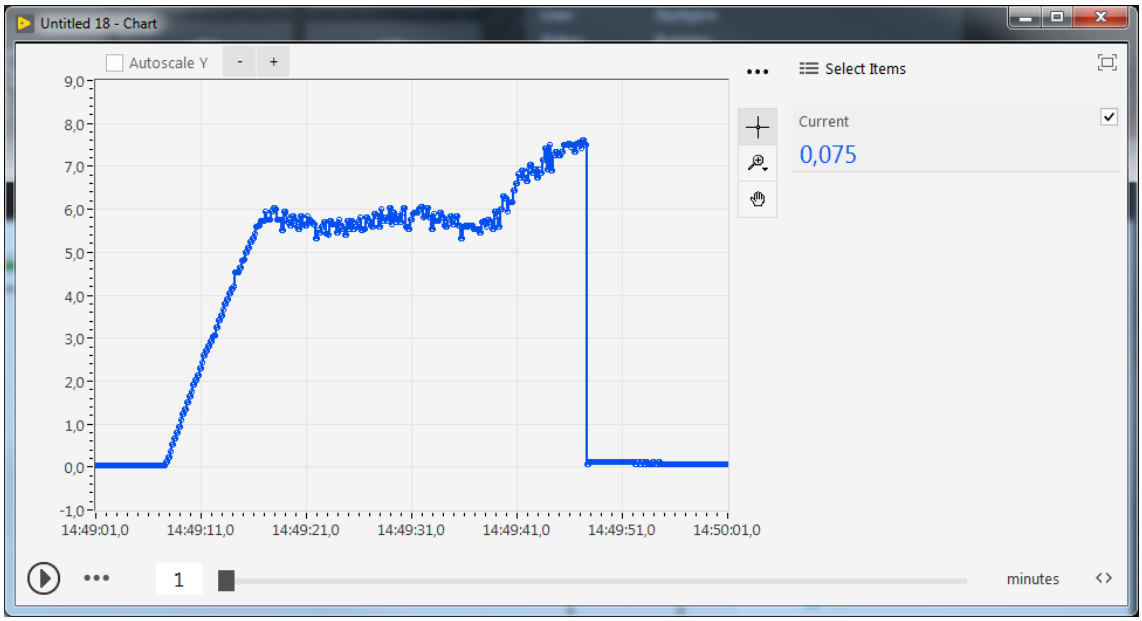
Alla uppsatta mål enligt kapitel 1.3 uppfylls av projektet och det utvecklade systemet. Mål 1 uppfylls genom den framtagna specifikation för riggen med lämpliga komponenter att använda när det gäller funktionalitet till låg kostnad. Mål 2 uppfylls genom att beslutad hårdvara är monterad till en färdig rigg. Mål 3 uppfylls och all hårdvara är framgångsrikt testad tillsammans med QMAX med Arduinon som interface mellan de två delarna. Mål 4 uppfylls och ett kommunikationsinterface mellan Arduino och QMAX är applicerat genom seriell kommunikation. Mål 5 uppfylls och QMAX styr elmotorn samt mottar sensorvärden från Arduinon.

Efter implementationen med QMAX kunde man enkelt få ut tydliga grafer. Då vissa värden överlappar varandra har vi valt att separera dessa, men det finns även möjligheter att samla alla grafer i ett fönster.



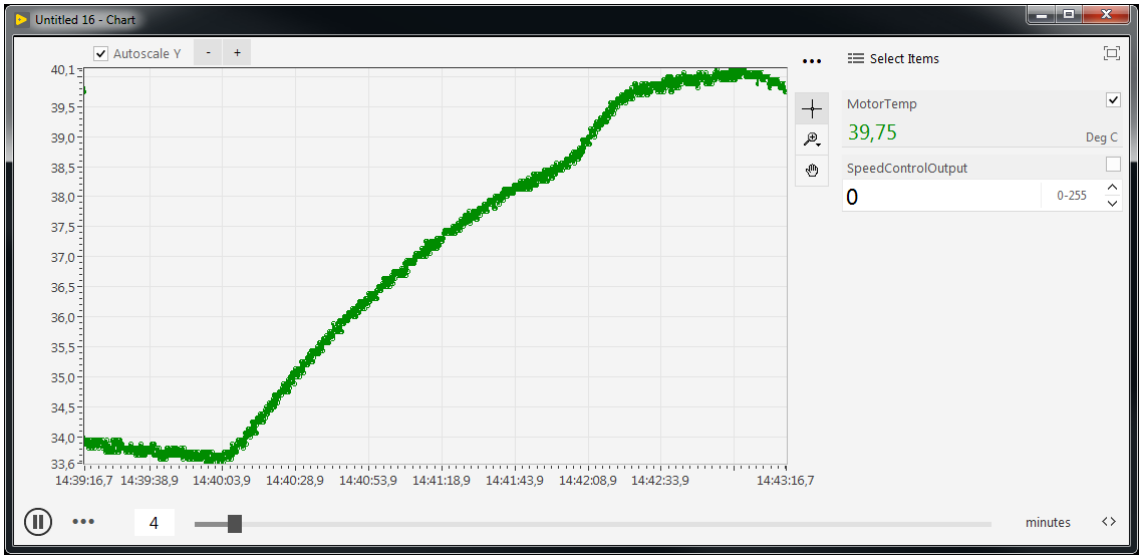
Figur 5.1: Viskositetsvariation under testkörning med grädde. Tiden är representerat på X-axeln medan viskositet är representerad av Y-axeln.

Figur 5.1 visar hur viskositeten i grädde förändras över tid som den blir vispad. Viskositeten är representerat av värdet på flexsensorn som har ett spann mellan 0 - 1023. Något att anmärka är hur fort grädden förändras då den går från flytande till ”fast”. Detta syns i grafen vid 14:45:02 där det sker en stor höjning av viskositeten.



Figur 5.2: Strömkonsumption under testkörning. X-Axeln representerar tid och Y-Axeln representerar strömstorlek genom elmotorn i Ampere.

Figur 5.2 visar hur motorn rampar upp till sitt börvärde för att sedan stabilisera sig. Då grädden blir tjockare blir motståndet för elmotorn större och strömförbrukningen ökar. När QMAX har registrerat att viskositeten i grädden har uppgått till den önskade bryts strömmen, vilket i detta fallet sker cirka 14:49:47.



Figur 5.3: Motortemperatur under testkörning. Temperatur i grader Celsius representeras av Y-axeln och X-axeln representerar tid.

I figur 5.3 kan man se att motortemperaturen ökar stadigt under hela testkörningen. Vad vi kunde se under längre testsekvenser utan belastning vid höga varvtal var att motortemperaturen tenderade att stabiliseras vid cirka 40 grader Celsius. Detta beror på den inbyggda fläkten som är fixerad vid motorns axel.

6
Slutsats

Vi lyckades implementera Arduino Due som en ersättare till CompactRIO Controller och få Arduino tillsammans med QMAX få demoriggen att vispa “perfekt” grädde för framtids kunder av QMAX. Olika grafer och tabeller lyckades fås fram genom QMAX som ett exempel för kunder att se hur programvaran fungerar.

ADDQ sparade 21 256 SEK på att använda sig av en Arduino till testriggen istället för en CompactRIO controller. CompactRIO controller är både större och tyngre. Det Resulterar i att Arduinon även är smidigare att ta med på mässor. Företaget hade som budget att demoriggen skulle kosta runt 10 000kr. Vi lyckades med det då slutkostnaden för riggen blev ca 10 000 SEK.

Kunder kan se väldigt tydliga mätningar i form av grafer och diagram som QMAX framställde. Resultaten var väldigt tydliga och styrningen av konstruktionen var simpel och effektiv. Riggen är flexibel att ta med på möten då den är enkel att bära och väldigt simpel att koppla QMAX till den. Att kunden imponeras med god grädde till en möjlig vald kladdkaka ökar chanserna att den köper QMAX. Det negativa med riggen är att den ej ser industriell ut och att ljudnivån kan vara såpass hög på elmotorn att det kan störa kunden.

7

Kritisk Diskussion

Tar man hänsyn till miljön kunde vi gjort det annorlunda då vi köpte en hel skruvdragare med ett li-ion batteri och modifierade produkten. Det ledde till att material slösades i onödan. Istället kunde vi ha köpt en elmotor för sig och ett batteri för sig och på så sätt åtgärdat det problemet. En positiv metod vi använde i konstruktionen av riggen var att vi använde en CPU fläkt från en trasig och slängd stationär PC, istället för att köpa en ny. Vi kunde ha använd en annan temperaturgivare då vår tar information för långsamt, vilket gör att vi endast kan använda en temperaturgivare (i elmotorn) istället för att ha ytterligare en i batteriet och en i grädden, vilket var tänkt från början.

Om man ska tänka ur ett tidsmässigt perspektiv kunde det ha gjorts annorlunda både från oss och från företaget. Det vi kunde ha gjort för att vara tidseffektivare var att beställa alla komponenter på en gång, istället för att göra beställningarna vid 3 olika tillfällen. Vi förlorade även väldigt mycket tid på att först lära oss Raspberry Pie. All den tiden kunde ha använts effektivare genom att exempelvis konstruera riggen på ett mer industriellt sätt, rent utseendemässigt eller kunna åtgärda problemet med den höga ljudnivån. Tiden kunde även ha gått åt till att implementera en skärm på kopplingsplattan som en extra funktion som visar ifall temperaturen på elmotorn är för hög eller som signalerar när grädden är klar. Från och med 23 januari har vi med produktchefen kommit överens om att det är en automatiserad elvisp som kommer vara testriggen för QMAX. Då hade vi kommit överens om att träffas varje måndag från 14.00-16.00 (2 timmar varje måndag). Våra möten skulle börja gälla från 10 december 2018 till 28 maj 2019. Om man räknar bort jul och nyår och påsk, skulle vi och produktchefen ha träffats 22 gånger (44 timmar). Produktchefen fick ett uppdrag på Essity och Elektrolux som prioriterades. Det ledde till att vi träffades endast 11 gånger (22 timmar). Detta gjorde att det blev misskommunikation via telefon och för lite hänvisning som gjorde att väldigt mycket tid gick i onödan.

Litteraturförteckning

[1] National instrument. ”Labview.” Hämtad 2019-03-26 från: http://www.ni.com/sv-se/shop/labview.html

[2] Researchgate. ”Effective labVIEW Programming.” Hämtad 2019-02-34 från: https://www.researchgate.net/publication/331408353_Effective_LabVIEW_Programming

[3] ADDQ.”R&D”. Hämtad 2019-05-30 från: https://www.addq.se/test-and-measurement/r-and-d/

[4] Arduino. ”Introduction.” Hämtad 2019-05-30 från: https://www.arduino.cc/en/Guide/Introduction

[5] Arduino. ”Environment.” Hämtad 2019-02-11 från: https://www.arduino.cc/en/Guide/Environment

[6] Atmel. ”SAM3X / SAM3A Datasheet.” hämtad 2019-02-16 från: http://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-11057-32-bit-Cortex-M3-Microcontroller-SAM3X-SAM3A_Datasheet.pdf

[7] Arduino. ”Arduino Due.” Hämtad 2019-02-16 från: https://store.arduino.cc/due

[8] Raspberry Pi Foundation. “Setting up your Raspberry Pi.” Hämtad 2019-09-27 från: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up

[9] National Instrument. “Compactrio Controller.” Hämtad 2019-05-11 från: https://www.ni.com/sv-se/shop/select/compactrio-controller

[10] Microship. “Brushed DC Motors Fundamentals.” Hämtad 2019-03-27 från: http://ww1.microchip.com/downloads/en/appnotes/00905a.pdf

[11] E.Ramsden. “Hall Effect Sensors Teory and application second edition.” Upplaga 2. Hillsboro,or,usa: Newness,2006

[12] Electrokit. ”Flexsensor datasheet.” Hämtad 2019-03-29 från: https://www.electrokit.com/uploads/productfile/41010/flex22.pdf

[13] Sparkfun "Flexsensor 4.5," [Elektronisk bild]. Tillgänglig: https://www.sparkfun.com/products/8606, hämtad: 2019-07-29.

[14] Sparkfun "Photo Interrupter - GP1A57HRJ00F," [Elektronisk bild]. Tillgänglig: https://www.sparkfun.com/products/9299, hämtad: 2019-07-29.

[15] Maximintegrated. ”Termometer DS18B20 datasheet.” Hämtad 2019-04-05 från: https://datasheets.maximintegrated.com/en/ds/DS18B20.pdf