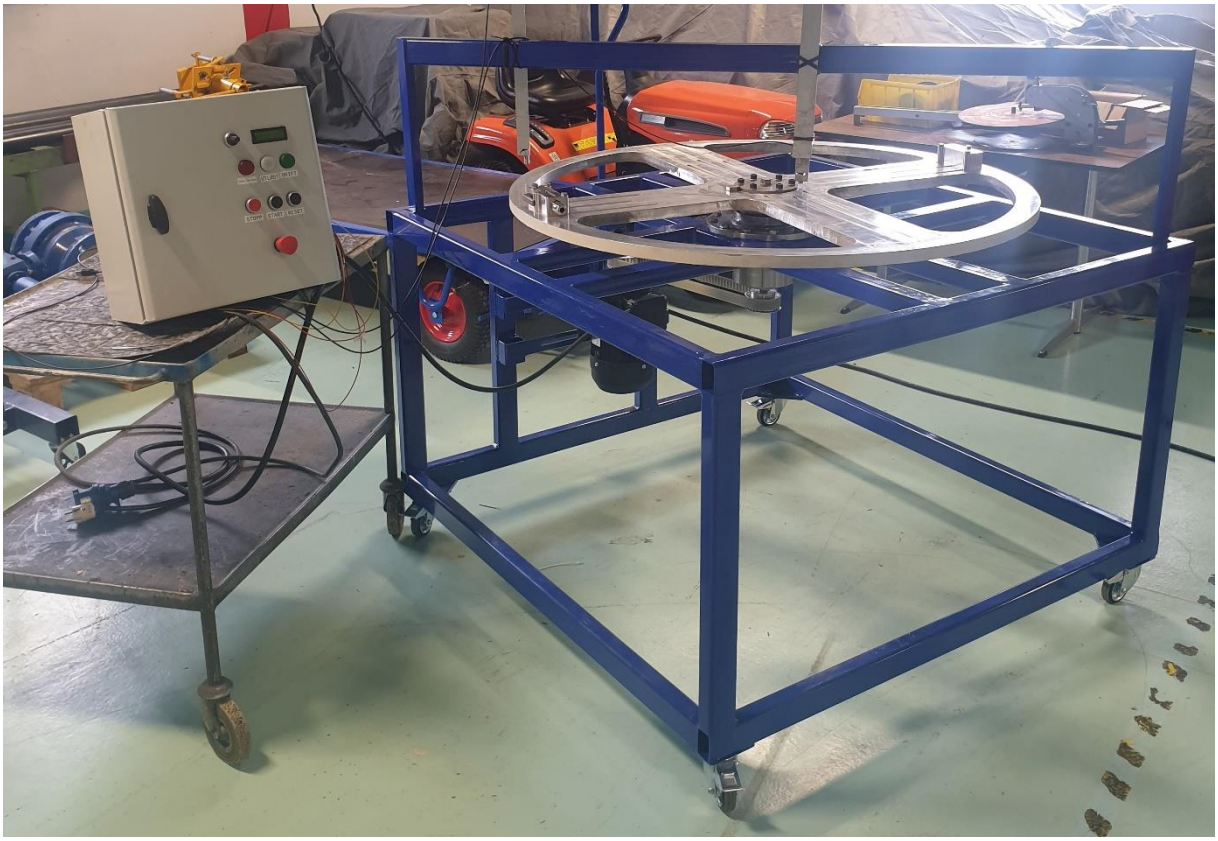
CHALMERS



Styrsystem till testbänk för rusningsvakt

Examensarbete inom högskoleingenjörsprogrammet Mekatronik

JOHAN JANSSON

INSTITUTIONEN FÖR ELEKTROTEKNIK

CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2021
www.chalmers.se

Förord

Denna rapport är skriven i syfte att ge en bra beskrivning på examensarbetet jag som Mekatronikstudent på Chalmers Tekniska Högskola har utfört hos företaget Maskin AS Feral i Magnor, Norge. Hos detta företag har jag konstruerat ett styrsystem för en testbänk som skall användas för att testa en rusningsvakt utvecklad av samma företag. Jag vill tacka Torgrim Finsrud och Lars Åsberg från Maskin AS Feral som gav mig chansen att hjälpa de med detta uppdrag. Jag vill tacka Christoffer Pedersen på Enestor AS som hjälpt mig med mycket av det teknologiska bakom arbetet. Jag vill tacka Viggo Iversen på Maskin AS Feral för hjälp med inköp av varor. Jag vill tacka Børge Andersen och Martin Torstensson på Maskin AS Feral för all hjälp i verkstaden. Till sist vill jag tacka min handledare och examinator Veronica Olesen som hjälpt mig med skrivandet av denna rapport och mycket mer.

Johan Jansson, Maj 2021

Sammanfattning

Ett stort problem som kan förekomma i många vattenkraftverk runt om i världen är rusning av turbiner på grund av att för mycket vatten kan forsa genom dem på grund av till exempel ett allvarligt elfel. Företaget Maskin AS Feral i Magnor, Norge har tagit fram en lösning till detta problem, en rusningsvakt, som konstant övervakar turbinens varvtal. Innan denna enhet kan säljas behöver den testas och för detta har en testbänk utvecklats av företaget. Denna testbänk består av en stor aluminiumskiva som skall symbolisera turbinaxeln som rusningsvakten skall monteras på. Syftet med detta arbete var att ta fram ett styrsystem för testbänken som skall styra en motor som roterar skivan, läsa av varvtalet och indikera på vilket varvtal rusningsvakten har löst ut. Detta gjordes i verkstadshallen på Maskin AS Feral. När systemet var klart var det viktigaste att undersöka noggrannheten på varvtalsräkningen och om rusningsvakten slog ut på rätt varvtal. På grund av feldimensionering av motor kunde inte fullständiga tester göras eftersom skivan endast kunde rotera upp till ca 400 varv/min i stället för den planerade maximala rotationshastigheten 1200 varv/min. På grund av detta problem kunde endast tester med skiva göras på lägre varvtal men tester utan skiva kunde göras på höga varvtal. Skivan roterades upp till 200 varv/min för att undersöka rusningsvaktens utlösningsvarvtal och det konstaterades att varvtalet maximalt kunde skilja ett varv/min. Motor utan skiva roterades upp till lite under 1200 varv/min och en noggrannhet för varvtalsräkningen konstaterades till att det kunde skilja ett till två varv/min. Detta säkerställdes med en pålitlig takometer. Resultaten visar att varvtalsräkningen är pålitlig och pricksäker och rusningsvakten har ett konstant utlösningsvärde som förväntat.

Abstract

A big problem that may occur in many hydropower plants all around the world is that turbines can gain a lot of speed by uncontrollable rushing water by for example a serious electrical fault. The company Maskin AS Feral in Magnor, Norway has created a solution for this problem, an overspeed protector, that constantly monitors the rotational speed of the turbine. But before this unit can be sold it must be tested and a test bench has been constructed for this purpose by the company. This test bench has a big aluminum disc which will emblematize the turbine shaft which the overspeed protector will be attached to. The purpose of this project was to construct a control system for this test bench which will control a motor that rotates the disc. It will calculate the rotational speed and indicate when the protector has been activated and on which rotational speed and this was done in the workshop hall of Maskin AS Feral. When the system was constructed the most important things to investigate were the accuracy of the speed counter and if the protector activated on the correct speed. Because of incorrect dimensioning of the motor complete tests could not be operated because the disc could only rotate up to 400 RPM instead of the planned maximum speed of 1200 RPM. Despite this problem tests of low speed could be done with the disc mounted and tests of high speed could be done with the disc dismounted. The accuracy of the speed counter was measured to differentiate a maximum of 1–2 RPM and the protectors speed of activation could differentiate 1 RPM. This was ensured with a tachometer. These tests ensures that the speed counter and the protector is reliable as expected.

Innehållsförteckning

Förkortningar/Terminologi

NC – Normaly Closed

NO – Normaly Open

PLC – Programmable Logic Controller

DC – Likström

AC – Växelström

VFD – Variable Frequency Drive/Frekvensomriktare

Arduino – Populär enkel programmerbar mikrodator

I/O – Input/Output

LCD-display:

VSS – Jordterminal

VDD – Matningsspänningsterminal

V0 – Spänning för kontrast

RS – Registerval

R/W – Read/Write

E – Enable

DB0-7 – Databussar, för skrivning till display

LED-/+ - Ljusstyrka på display

1. Inledning

Detta är det inledande avsnittet där Bakgrund syfte, avgränsningar och frågeställningarna tas upp.

1.1. Bakgrund

Ett allvarligt problem som tyvärr kan uppstå i vattenkraftverk runt om i världen är att om vatten börjar forsa utan kontroll genom turbinerna på grund av till exempel ett allvarligt elfel med slussarna, så börjar turbinerna rotera upp i farligt höga hastigheter som kan förminska livstiden på lager och mycket annat. För att förhindra detta problem har företaget Maskin AS Feral konstruerat en säkerhetsanordning, en rusningsvakt, som skall övervaka rus av turbin beroende på varvtalet och stoppa problemet i tid om det skulle uppstå. Detta sker genom att om turbinen roterar med ett otillåtet varvtal löser rusningsvakten ut av den centrifugalkraft som uppstår. Rusningsvaktens utlösande område kan ställas in beroende på turbinens axelstorlek och varvtal som den skall aktiveras på.

Innan denna anordning kan sättas i bruk har företaget utvecklat en testbänk med en stor aluminiumskiva. Denna skiva skall simulera en turbinaxel i ett vattenkraftverk för att kunna testa och mäta vaktens funktion och säkerställa utlösningsvarvtalet i en säker miljö. Denna skiva skall rotera med hjälp av en motor och en drivlina konstruerad av två remhjul och en rem.

1.2. Syfte

Syftet med detta arbete är att konstruera ett fullt fungerande styrsystem för en testbänk som skall användas för att testa en säkerhetsanordning för turbiner i vattenkraftverk. Styrsystemet skall användas för att styra testbänken genom att styra en motor som roterar en stor aluminiumskiva, läsa av data med givare, visa varvtal på en LCD-display och stoppa motorn när anordningen har aktiverats eller stoppa motorn manuellt med någon typ av mikrokontroller eller Programmable Logic Controller, PLC.

1.3. Avgränsningar

Rusningsvakten är redan konstruerad av företaget och kommer inte att tas hänsyn till i detta arbete. Testbänken utan styrsystem kommer att byggas upp av företaget och har inte något med detta arbete att göra. Motorn och frekvensomriktaren är redan införskaffade av företaget.

1.4. Precisering av frågeställning

För en testbänk som skall användas för att testa en rusningsvakt som skall utlösas på ett visst varvtal krävs att varvtalsberäkningen är så noggrann som möjligt.

- Kommer utlösningsvarvtalet vara det varvtal som anordningen var inställd för?
- Hur exakt kommer varvtalsräkningen vara för systemet?

2. Teknisk bakgrund

I detta avsnitt tas den tekniska bakgrunden om de mindre självklara komponenterna upp.

2.1. Arduino Mega 2560

En Arduino Mega 2560 är en av många modeller av mikrokontrollerkort gjort av Arduino. Arduino Mega 2560 är ett av de större korten med fler digitala och analoga in- och utgångar än de andra korten som finns. Den har 54 digitala I/O-pins varav 15 kan användas som PWM-utgångar och 16 analoga ingångar. Digitala I/O-pins innebär kontakter som fungerar som både input och output för digitala signaler alltså 0 eller 5 V DC. Analog ingång innebär att denna ingång kan Arduinon använda för att läsa av ett värde mellan 0 och 5 V DC. PWM, Pulse Width Modulation, innebär att med hjälp av digitala signaler skapar en analog signal med hjälp av mycket snabba pulser på 5 V. Den analoga spänningen blir som ett medelvärde av tiden som 5V skickas i jämförelse med 0 V DC. Som exempel fås 2,5 V om 5 V skickas hälften av tiden som 0 V skickas. Det finns pins för jord, +5 V DC, reset för återställning, Vin för att mata spänning till Arduinon, USB-kontakt för programmering och en 2,1 mm DC-Power jack för matning av spänning. Denna Arduino kräver en matningsspänning mellan 7–12 V DC för att fungera optimalt trots att den arbetar med 5 V DC. Arduino mega 2560 är baserad på en ATmega2560 vilket är en 8-bit Atmel mikrokontroller, där Atmel är märket på kontrollern. Arduinons PWM-signaler kan skickas med en upplösning på 8 bitar men den klarar av att läsa analoga signaler med en upplösning på 10 bitar [1].

2.2. Arduino IDE

Arduino IDE är det mjukvaruprogram som används för att programmera en Arduino. I detta program kodas det i ett språk mycket likt C. Det finns många färdiga bibliotek som kan laddas ner och användas av för att förenkla de program som skrivs.

2.3. Induktiv givare

En induktiv givare är en typ av givare som används för att känna av metalliska föremål (mestadels järnbaserade) i dess närhet, ofta inom millimeters räckhåll. Induktiva givare använder sig av magnetiska fält för att känna av föremål i dess absoluta närhet [2]. Det magnetiska fältet skapas av en ström som leds genom en spole inuti givaren. När ett metalliskt föremål kommer inom räckvidd av fältet, framför givaren, ökar induktansen och givaren löser ut [3]. För en induktiv givare med tre olika kablar brukar en kabel vara för matningsspänning, en för jord och en för styrsignalen. När ett metalliskt objekt kommer inom räckvidd för en Normaly Open, NO, givare är signalen i kabeln hög och låg för övrigt. När ett metalliskt objekt kommer inom räckvidd för en Normaly Closed, NC, givare är signalen låg och hög för övrigt.

som kan läsas av, när givaren ”löser ut”. Om givaren är Normaly Open, NO, skickas styrsignalen eftersom givaren ”löser ut” men om givaren är Normaly Closed, NC, bryts styrsignalen när givaren ”löser ut”.

2.4. Asynkronmotor

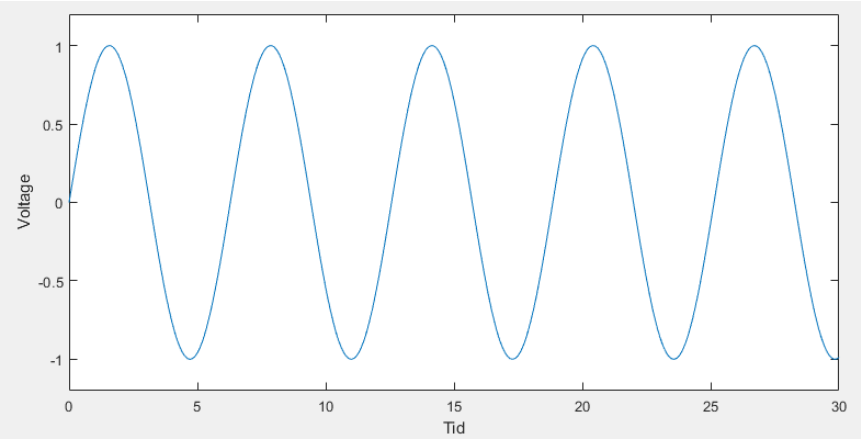
En asynkronmotor är en induktiv motor som drivs på växelström. Det finns asynkronmotorer som drivs på både en fas och tre faser. En asynkron motor består av en stator som är kopplad till en spänningstillförsel och en rotor som är den roterande delen. Strömmen leds genom spolar som statorn består av, denna ström skapar ett elektromagnetiskt fält som får rotorn att rotera. Vid högre strömmar ökar det magnetiska fältet och den blir lättare att rotera rotorn. Den roterande hastigheten på rotorn beror på frekvensen som matas till motorn [4]. En

asynkron motor roterar alltid med lägre rotationshastighet än det synkrona varvtalet och kan ha en effektivitet på upp till 97 % [5].

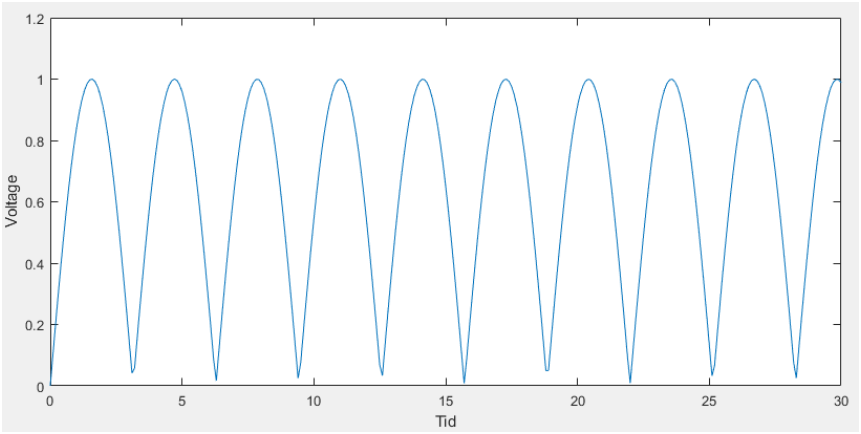
Vid koppling av tre faser till en asynkron motor kan detta ske på flera sätt där de två vanligaste sätten är Y-koppling och D-koppling. Vid en Y-koppling är strömmen i en fas och mellan faserna lika men spänningen mellan faserna är roten ur tre gånger så stor som spänningen mellan en fas och en neutralpunkt. För En D-koppling är strömmen mellan faserna roten ur tre gånger så stor som strömmarna i en fas men spänningen mellan faserna och i mellan en fas och en neutralpunkt är lika [6].

2.5. Frekvensomriktare

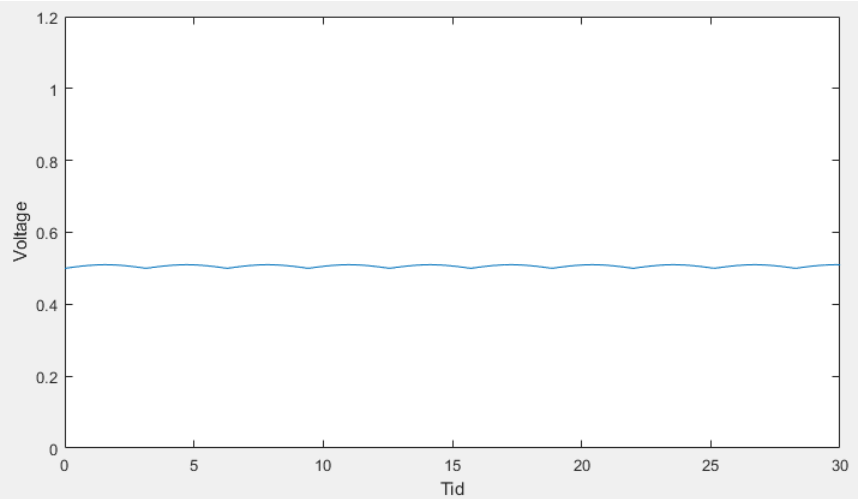
En frekvensomriktare eller VFD, Variable Frequency Drive, är en komponent som ofta används för att styra motorer och pumpar. Den kan användas i flera områden som exempelvis fluidsystem eller bearbetningssystem. Den ändrar rotationshastigheten på en vanlig asynkron motor genom att ändra frekvensen på strömmen till motorn. Detta görs genom att konvertera växelström, AC, till likström, DC, se figur 2.1, 2.2 och 2.3 nedan för hur strömmen ändras till likström. Denna likström omvandlas sedan till en ny vågform, en växelström med önskad spänning och frekvens. Detta tillåter en att styra en motor från 0 varv/min till dess maximala märkta hastighet eller högre, beroende på lasten motorn skall rotera [7]. Frekvensomriktare har stora fördelar jämfört med andra hastighetsstyrningsval, eftersom en VFD är otroligt effektiv, pålitlig, flexibel och användare kan enkelt förbikoppla en VFD för service eller reparation [8].



Figur 2.1: Illustration av växelspänning på en fas in till VFD



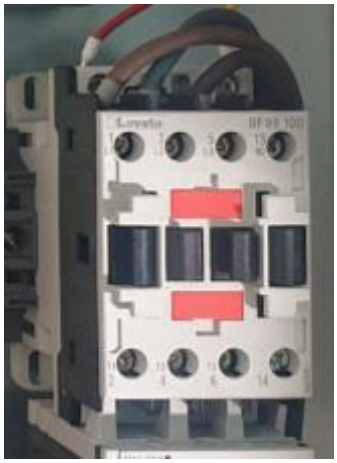
Figur 2.2: Illustration av växelspänning omvandlad till likström med VFD



Figur 2.3: Illustration av utjämnad likspänning innan den omvandlas tillbaka till växelspänning med önskad frekvens och spänning

2.6. Kontaktor

En kontaktor liknar mycket ett relä och är en elektromagnetisk switch som antingen sluts för att leda ström eller öppnas för att stoppa ström från att ledas. Kontaktorn är en mycket viktig säkerhetskomponent som används vid motorstyrning eftersom den stoppar eller leder strömmen som skall till motorn, den funkar kort sagt som en strömbrytare [9]. Beroende på om kontaktorn är tillverkad för en fas eller tre faser kopplas faserna till ingångarna på kontaktorn och lika många kablar kopplas till motorn från kontaktorns utgång. Det finns även in- och utgång för styrspänningen för att sluta kontaktorn. Kontaktorn leder endast ström genom ingångarna till utgångarna om en korrekt styrspänning ges, se figur 2.4 för exempel av kontaktor. Som exempel för en kontaktor gjord för trefas som har en spole som kräver 24 V DC fungerar den sådan att endast om en spänning på 24 V DC läggs på spolen kan spänningen och strömmen på trefasen ledas vidare till motorn, se figur 2.4 nedan. Ibland finns även en NC- eller NO-kontakt som sitter på kontaktorn om en annan styrsignal skall ges när kontaktorn är av eller på, se kontakt 13 och 14 i figur 2.4 nedan.



Figur 2.4: Kontaktor kopplad till trefas där den bruna, svarta och blå kablarna är de tre faserna, röd är +24 V DC spolspänning och gul kabel -24 V DC

2.7. Överströmsrelä

Ett överströmsrelä är en mycket viktig komponent för skydd av en motor. Ett överströmsrelä känner av om för höga strömmar leds till motorn av till exempel för stor påfrestning av motor vilket leder till högre strömförbrukning och detta skadar motorn i längden på grund av överhettning, vilket förkortar livstiden på motorn. Överströmsreläet skall bryta strömmen om den blir för hög. Strömmen till motorn leds genom ett bimetalliskt material som är en hopsättning av två olika metaller med två olika längdutvidgningskoefficienter. När dessa blir varma kommer de böjas eftersom den ena töjs ut mer av värmen än den andra. När strömmen blir för stor böjs materialet och strömmen slutar ledas. Ett modernt överströmsrelä brukar gå att återställa både manuellt och automatiskt [10].

2.8. Kort med optokopplare

Kort med optokopplare är ett redan färdigt kretskort med fyra optokopplare av modellen PC817 som är en populär 4-benad optokopplare som kan användas till allmänna kretsar och är bra till funktioner där en mikrokontroller är inblandad [11]. Optokopplare är en komponent som kan användas som spänningskonverterare. En optokopplare består av en lysdiod och en fototransistor. Detta innebär att spänningarna som finns på var sida av optokopplaren är elektroniskt isolerade från varandra. När en spänning ligger på dioden och det ledes ström känner transistorn av ljuset och transistorn sluts och ström ledes genom den vilket ger en spänning på andra sidan. På detta vis kan en låg spänning konverteras till en hög spänning på ett säkert sätt eftersom dessa är isolerade från varandra [12]. Korten är designade för antingen en inspänning på 24 V DC och en utgångsspänning på 5 V DC eller tvärtom och har en lysdiod för att visualisera när optokopplaren matas med spänning. Dessa kretsar passar bra till att omvandla styrsignaler på 24 V DC från induktiva givare till 5 V DC så att en mikrokontroller som endast kan hantera 5 V DC, som en Arduino, enkelt kan hantera dessa.

2.9. Takometer

En takometer är ett verktyg som används för att mäta rotationshastigheter på olika föremål. Detta är ett verktyg som många ser varje dag, varvmätaren i en bil är ett exempel på en takometer. Det finns två olika typer av takometrar, analoga och digitala, men endast digitala kommer att tas upp eftersom en sådan används i arbetet. Digitala takometrar är vanligtvis instrument som inte kräver fysisk kontakt vid mätning. Mätningen sker genom att läsa av märken på ett roterande objekt [13]. Varvtalet kan antingen beräknas genom att räkna antalet pulser som läses under en viss tid eller beräkna tiden mellan varje läsning. En digital takometer kan delas upp i tre olika typer, optisk, induktiv och magnetisk, här beskrivs optisk och induktiv. En optisk takometer användes sig av optik för att mäta varvtal, antingen med laser eller ljusavgivande diod i kombination med fotodiod eller fototransistor som detektor. Optiska takometrar har även högre noggrannhet än andra digitala takometrar men mätningarna kan störas av smuts och liknande som kan blockera ljuset [13]. En induktiv takometer använder induktans för att mäta varvtal och används ofta inom fordonsindustrin för antisladd-system och ABS-system [13]. I stället för en ljuskälla med detektor för att mäta kan en induktiv givare användas för att läsa på en induktivvänlig punkt på det roterande objektet. Induktiva takometrar är dock mindre noggranna vid höga varvtal till skillnad från de optiska [13].

2.10. Hitachi HD44780

För många år sedan manufakturerade Hitachi en mikrokontroller som var designad för att driva en alfanumerisk LCD-modul. Mikrokontrollern hade ett mycket enkelt gränssnitt som gjorde det mycket enkelt att koppla den till en vanlig mikroprocessor eller mikrokontroller. Denna mikrokontroller har definierat ett gränssnitt som i dagens samhälle blivit något av en standard för denna typ av displayer. En HD44780s gränssnitt består alltid av åtta dataingångar för vad som skall skrivas till skärmen, en RS-ingång, en R/W-ingång och en enable-ingång. En ny generation av drivrutiner existerar nu som har bytt ut Hitachi men fortfarande behållit många av dess funktioner [14].

3. Metod

En systembeskrivning behövde göras för att ge en bra översikt över hela systemet som senare ledde till ett mer noggrant kopplingsschema. Två mycket grundläggande systembeskrivningar gjordes där den ena bestod av en PLC som huvudenhet och den andra beskrivningen bestod av en mikrokontroller som huvudenhet eftersom det inte var bestämt vilket typ av system som skulle användas. Detta gjordes för att få en bra och grundlig översikt över hur hela systemet kunde vara uppbyggt och se vilka andra komponenter som kunde krävas beroende på val av huvudenhet. Den valda systembeskrivningen blev den med mikrokontroller som huvudenhet av ekonomiska skäl. Denna beskrivning utvecklades med fler komponenter som kontaktor, spänningsaggregat, olika typer av omvandlare, LCD-display, överspänningsskydd, VFD, Motor, knappar, lampor, nödstopp, potentiometer och andra komponenter som ansågs nödvändiga för systemet. Den färdiga systembeskrivningen beskrev hela systemet och hur alla komponenter var kopplade till varandra. Detta för att ge företaget en bra översikt för hur systemet skulle vara uppbyggt.

För att bygga upp en del av styrsystemet permanent med Arduinon som en egen krets med reläer, kablar, Arduino och resistorer behövdes en lödstation och lödpenna användas för att löda ihop kretsen och få det hållbart och permanent.

Andra verktyg som skruvmejslar, spärrskaft och blocknycklar användes också för att kunna utföra vissa moment av arbetet.

Under tiden det praktiska arbetet utvecklades utfördes tester för att bekräfta att allt fungerade som det var planerat för vad systemet var kapabelt till att klara vid den tidpunkt testet utfördes. Dessa tester gjordes genom att provköra programmet i olika stadier av arbetet. Spänningar och olika signaler mättes med hjälp av en multi-meter för att säkerställa att alla ingångar och utgångar angav eller läste rätt spänningar, detta för att säkerställ att inga komponenter skulle förstöras av för höga spänningar. I slutet av arbetet genomfördes tester för att mäta att varvtalet som läses och beräknas utav systemet är det faktiska varvtalet eller i alla fall så nära det faktiska varvtalet som möjligt. Detta gjordes genom att köra systemet och rotera skivan och säkerställa varvtalet med hjälp av en takometer.

4. Genomförande

I detta kapitel beskrivs genomförandet av arbetet. Här ingick bland annat val av komponenter som krävdes för systemet. En elektronisk konstruktion av komponenter och programmering utvecklades för systemet. Under arbetsprocessen testades systemet för att bekräfta dess funktion och pålitlighet.

4.1. Val och funktion av komponenter

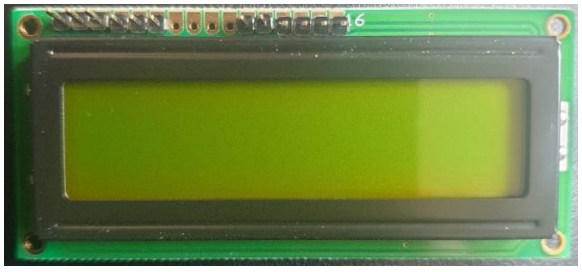
4.1.1. Huvudkontroller

Med hjälp av de två kompletta systembeskrivningar som hade gjorts kunde valet av system göras. Med en PLC som huvudenhet blev priset med nödvändiga komponenter för högt även om en billigare PLC valdes ut i jämförelse med en mikrokontroller. På grund av detta ansågs en mikrokontroller vara en mer passande huvudenhet enligt företaget. Den typ av mikrokontroller som valdes var en Arduino. Nackdelen med att välja en Arduino var dock att den arbetar med 5 V DC medan valda givare, lampor och VFD jobbar med den industriella standarden 24 V DC. För att få detta att fungera behövdes ett flertal DC-DC omvandlare och andra komponenter väljas ut, dessa tas upp senare i delkapitlet.

Arduinon som valdes som huvudenhet var en Arduino Mega 2560 eftersom detta ansågs vara en mycket enkel kontroller att använda, dessutom fanns det en tillhörande prototypsköld för att enkelt kunna bygga en hel krets för systemet. Arduino Mega 2560 valdes också på grund av antalet I/O eftersom denna har 70 I/O i jämförelse med andra modeller, till exempel Arduino Uno med 20 I/O. Detta var ett måste med tanke på att alla I/O som behövdes för alla givare, knappar, lampor, VFD, LCD-display mm. En nackdel med att välja en Arduino var tyvärr upplösningen av bitar när det skrevs eller lästes analoga signaler. Om det skulle skrivas en PWM-signal till något så var upplösningen endast 8 bitar, alltså ett värde mellan 0–255 kunde endast skickas och när en analog signal lästes av var upplösningen 10 bitar alltså ett värde mellan 0–1023. Med detta i åtanke hade det varit en fördel med att välja en PLC eftersom många PLCs kan ha 12 bitars upplösning vilket ger en mer noggrann avläsning eller analog signal.

4.1.2. LCD-display

Som LCD-display valdes en vanlig 16x2 LCD-display. Varför en sådan liten display valdes var för att det inte krävdes något mer avancerat. Displayen har en programvara motsvarande Hitachi HD44780 drivrutiner där 16x2 innebär att displayen har 16 kolumner och två innebär antalet rader som tecken kan skrivas på. Genom att välja en display med den programvaran kunde displayen mycket enkelt programmeras med en Arduino tack vare Arduinons LiquidCrystal-bibliotek med färdiga funktioner för LCD-displayer. Det fanns mycket bättre och mer avancerade alternativ av displayer men ur en ekonomisk synvinkel och hur enkelt det är att programmera en sådan display med en Arduino var den lilla displayen det rätta valet för detta arbete. Displayen kunde användas på två sätt, antingen genom att skriva med fyra bitar, alltså använda kontakterna 11–14 eller skriva med åtta bitar, då används även kontakterna 7–10. I detta arbete användes fyra bitars skrivning. Alla kontakter från 1–16 var i ordning, VSS, VDD, V0, RS, R/W, E, DB0-7, LED- och LED+, detta kan ses högst upp på displayen i figur 4.1 nedan där 1 är längst till vänster och 16 längst till höger.



Figur 4.1: 16x2 LCD-display

LCD-displayen användes för att smidigt och enkelt hjälpa användaren av testbänken genom att berätta vad som skall göras och visa vad som händer. Med hjälp av alla externa enheter kopplade till Arduinon kunde det skrivas ”Close lid” om locket på testbänken inte var stängt, när locket väl var stängt skrevs det ut ”Choose speed:” på första raden och varvtalet i varv/min på andra raden vilket kunde ändras genom att vrida på potentiometern. När varvtalet hade valts, kunde startknappen tryckas ned och det skrevs ”Current speed:” på första raden och nuvarande varvtal i varv/min på andra raden. Det sista som skrevs, om stoppknappen blev nedtryckt eller rusningsvakten löste ut, var ”Speed attained:” på första raden och varvtalet som uppnåddes i varv/min på andra raden. Ifall skivan skulle uppnå det maximalt valda varvtalet innan stoppknappen trycktes ned eller rusningsvakten löste ut skrevs ”Max RPM reached” ut.

4.1.3. Induktiva givare

Det användes tre identiska induktiva givare till tre olika avläsningar i systemet. Den valda induktiva givaren för detta arbete var av typen NO. Den kan använda en matningsspänning i intervallet 10 V och 36 V DC. På grund av industristandard valdes 24 V DC som matningsspänning. Givaren har en arbetshastighet på 800Hz vilket innebär att den klarar av 800 läsningar i sekunden och hade ett maximalt funktionsavstånd på 2 mm. Alla givare har tre kablar, +24 V, jord och en signalkabel till Arduino, se figur 4.2 för givaren. Signalkabeln kopplades först till optokopplaren för att omvandla 24 V till 5 V DC som sedan kopplades till Arduinon. Den första givaren användes för att inspektera om locket över testbänken var stängt, detta av säkerhetsskäl. När givaren läste av att locket var stängt kunde processen fortsättas. Den andra givaren användes för att beräkna varvtalet på skivan. Det fanns en mätpunkt utsatt på aluminiumskivan som givaren kunde läsa av. Den tredje givaren var till för att läsa av när rusningsvakten hade lösts ut och systemet skulle då veta att processen skulle stoppas. Detta bekräftades av givaren genom att när rusningsvakten löste ut klickade en arm ut av centrifugalkraften på vakten och denna arm kunde läsas av givaren.

Figur 4.2: Induktiv givare

4.1.4. VFD

Frekvensomriktaren var redan vald av företaget eftersom det fanns en på lager som inte användes och passade perfekt till arbetet både funktionellt och ekonomiskt sett enligt gjorda beräkningar av företaget. Den valda frekvensomriktaren kan omvandla 240 V AC, 50 Hz på tre faser till 0–230 V AC och 0–500 Hz, för den valda motorn låg frekvensen mellan 0–50 Hz. Omriktaren krävde en omprogrammering enligt instruktionsboken för att kunna styras av Arduinon. Det som ändrades från standardinställningarna på omriktaren var att den skulle starta när en digital styrsignal på 24 V skickades till omriktarens terminal från Arduinon, standard startas omriktaren med hjälp av en knapp på omriktaren. Det programmerades så att omriktaren kunde styras utav en 0–10 V analog styrsignal från Arduinon som standard var satt som manuell styrning från kontrollpanelen på omriktaren. Den maximala frekvensen, basfrekvensen och frekvensen på motorn sattes till 50 Hz för att inte kunna överbelasta motorn, dessa var satta som noll som standard. Ett procentuellt värde för ett inbyggt motorskydd sattes också och detta till 86 % eftersom den maximala strömmen frekvensomriktaren kunde skicka ut var 7 A men den maximala strömmen motorn klarade av att ta emot var 6 A, det vill säga 86 % av den maximala strömmen från frekvensomriktaren

detta var satt till 100 % som standard. Se figur 4.3 nedan för framsidan av frekvensomriktaren.

Figur 4.3: Framsida av frekvensomriktare

4.1.5. Lampor, knappar, potentiometer, nödstopp

För att hjälpa användare av testbänken införskaffades tre knappar, tre lampor, en potentiometer och ett nödstopp. De tre knapparna användes som startknapp, stoppknapp och reset-knapp, alla dessa är återfjädrande knappar eftersom en snabb avläsning var det enda som behövdes för de funktioner som knapparna användes till. En konstant signal från en knapp var inget krav. De tre lamporna användes som driftlampa (grön), en lampa som visade om rusningsvakten hade utlösts (vit) och en lampa för att visualisera om locket var stängt eller processen var stoppad (röd). Potentiometern användes för att ställa in det önskade varvtal som skivan skulle roteras upp till genom att vrida på ratten på potentiometern. Nödstoppet fanns av säkerhetsskäl och den var kopplad så att om den trycks ned skulle den digitala styrströmmen brytas till omriktaren och motorn skulle stoppas.

4.1.6. Motor

Motorn valdes av företaget och valdes baserat på den redan existerande frekvensomriktaren. Då omriktaren endast klarar av att driva en motor på maximalt 1,5 kW och 7 A valdes en trefasmotor på 1,5 kW och 6 A. Motorn D-kopplades för att kunna köras på trefas 230 V AC och kopplades direkt till frekvensomriktaren.

4.1.7. Spänningskonvertering av likspänning

Det användes två olika komponenter för att konvertera likspänning där den ena var två färdiga kretskort med optokopplare för att omvandla 24 V DC till 5 V DC och tvärtom och en signalomvandlare som kunde omvandla en analog spänning mellan 0–5 V DC till en analog spänning mellan 0–10 V DC. Optokopplarna valdes på grund av dess isoleringsförmåga och att de var färdigdesignade för 24 V till 5 V. Kortet som omvandlade 24 V till 5 V användes till att omvandla styrsignalen från givarna så att Arduinon kunde hantera dessa. Kortet som omvandlade 5 V till 24 V användes för att tända lamporna eftersom de krävde en spänning på 24 V för att lysa. När en lampa behövde tändas skickades en signal från Arduinon som omvandlades till 24 V och tände lampan. Signalomvandlaren användes för att omvandla en 0–5 V PWM-signal från Arduinon till en analog 0–10 V styrsignal till frekvensomriktaren. En 0–10 V spänning krävdes för att styra motorn med frekvensomriktaren och en sådan spänning

kunde inte skickas från en Arduino utan att omvandlas. Signalomvandlaren krävde också en matningsspänning på 24 V för att fungera.

4.1.8. Kontaktor och överströmsrelä

En kontaktor behövdes för att kunna styra motorn och inte mata en ström till motorn konstant. Kontaktorn är gjord för trefas och klarar av en spänning på 690 V AC men användes för 230 V AC och krävde en spolspänning på 24 V DC för att sluta kontaktorn. På kontaktorn monterades ett överströmsrelä som tillhörde kontaktorn och kunde lätt monteras på kontaktorn för att skydda motorn i en sådan situation att strömmen överskred den maximalt tillåtna strömmen. På överströmsreläet kunde det enkelt den maximala strömmen väljas genom att vrida på en skruv och välja ett värde mellan 4 A och 6,5 A. Skyddet hade också en stoppknapp för att stoppa tillförseln av ström manuellt och en reset-knapp för att kunna återställa skyddet när det aktiverats.

4.1.9. Relä

Två reläer för montering på kretskort införskaffades som kräver en spolspänning på 5 V DC och kan leda en spänning på 24 V DC. Detta behövdes för att kunna sluta kontaktorn med en signal från Arduinon och leda vidare den digitala styrsignalen till frekvensomriktaren utan att den behövde vara i gång hela tiden. Anledningen till att relä användes till frekvensomriktare och kontaktor var egentligen endast för att företaget rekommenderade detta i början av arbetet.

4.1.10. Spänningsaggregat

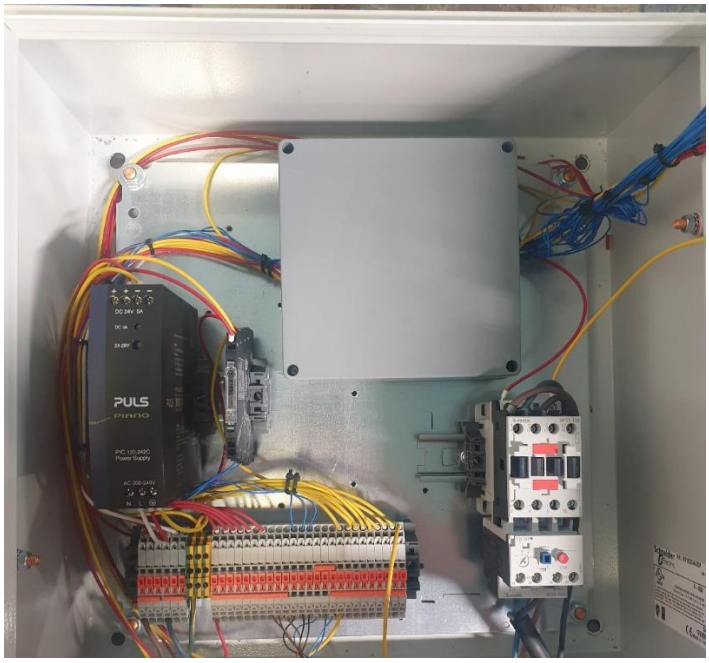
Tre spänningsaggregat valdes ut. Ett 24 V DC-aggregat för att kunna mata spänning till lampor, givare, frekvensomriktare, signaomvandlare och resterande 24 V-komponenter. Ett 12 V DC-aggregat valdes för att ge en matningsspänning till Arduinon som även om den jobbade med 5 V krävde en matningsspänning mellan 7–12 V DC. Sist valdes ett 5 V DC-aggregat för att assistera Arduinon med att mata 5 V till komponenter Arduinon inte klarade av. Dock behövdes aldrig 5 V-aggregatet eftersom Arduinon klarade av att mata 5 V till krävande komponenter på egen hand. Ett problem som skulle kunna ha uppstått om 5 V-aggregatet hade använts var att nollpunkten inte hade vart samma som för Arduinon. Detta skulle kräva att få nollpunkterna för Arduinon och aggregatet att bli gemensamma för att fungera.

4.2. Koppling av styrsystem

4.2.1. Insida skåp

Ett begagnat plåtskåp användes för att kunna montera alla komponenter i förutom frekvensomriktaren som monterades på utsidan. Detta på grund av kylningsproblem i ett stängt utrymme eftersom inget luftflöde finns in eller ut. Detta skulle troligen innebära att omriktaren skulle överhetta i skåpet. I skåpet monterades kopplingsplintar på din-skenor som kablaget kunde kopplas till för att enkelt kunna parallellkoppla +24 V och -24 V från 24 V DC-aggregatet som också monterades på en din-skena, se nedre och vänstra delen av skåpet i figuren nedan. Kopplingsplintarnas kontakter var fjädrande så det var enkelt att bara trycka i kablarna. Tre av kopplingsplintarna var jordplintar som med metallklämmor jordade i skåpet. Till dessa jordplintar kopplades jorden från 230 V AC och sedan vidare till de två DC-aggregaten. En jordplint lämnades fri för möjlig framtida inkoppling av ännu ett DC-aggregat. Kontaktor med överströmsrelä monterades på en separat din-skena för att försöka isolera de höga spänningarna och strömmarna från de andra komponenterna för att minska störningar.

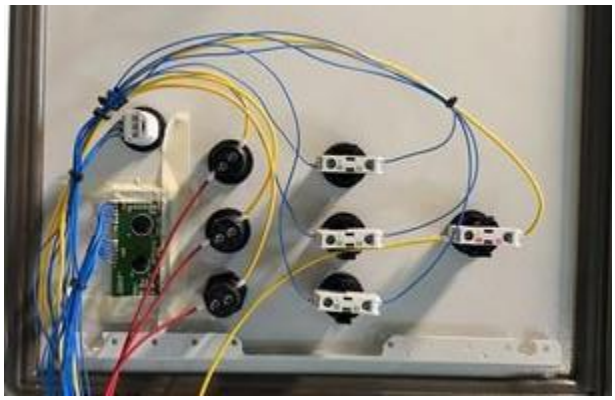
Arduinon, 12 V DC-aggregatet och optokopplarna monterades i en plastlåda som monterades i skåpet, detta för att isolera de lättpåverkade kretsarna från de resterande komponenterna, se upptill och till höger i skåpet i figuren. Storleken på kablaget för 24 V var 0,75 mm^2 och för 5 V användes 0.34 mm^2 och till nästan alla kabeländar användes ändhylsor av isoleringsskäl. I varje ände av din-skenorna sattes även ändplintar som skruvades fast så att inget skulle kunna röra på sig, se ändarna av varje skena i figur 4.4 nedan.



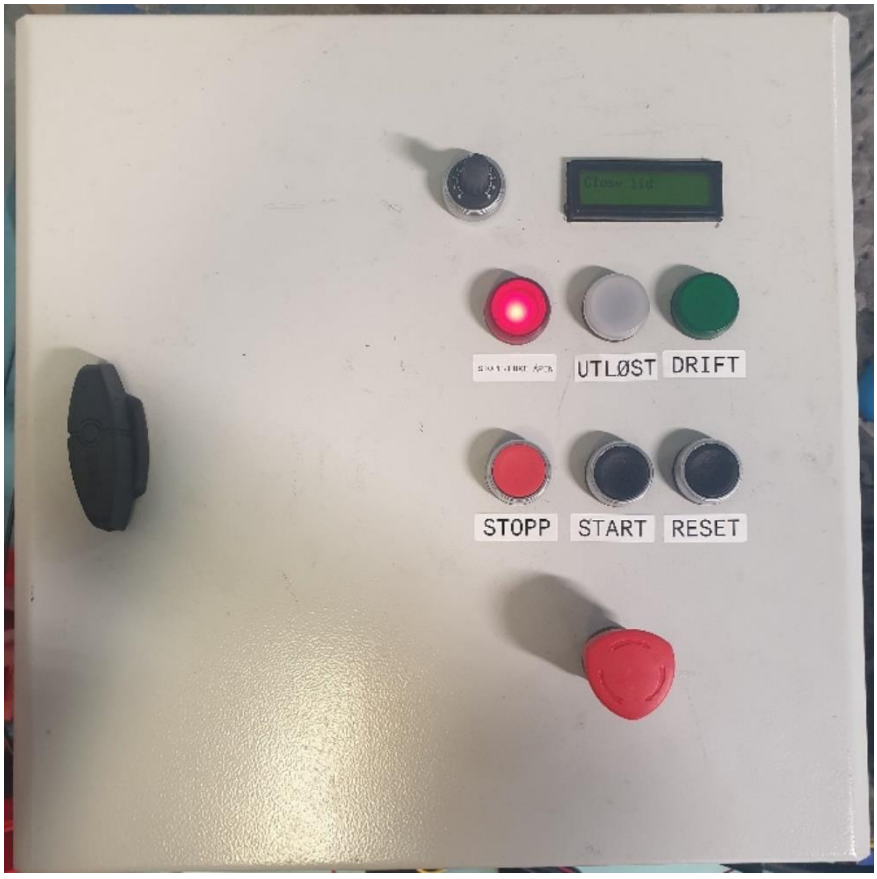
Figur 4.4: Insida av skåp för styrning av testbänk

4.2.2. Utsida skåp

Lamporna, knapparna, nödstoppet, potentiometern och LCD-displayen monterades på framsidan skåpet för att enkelt kunna hantera systemet. Alla komponenter monterades mycket enkelt genom att borra eller skära hål på framsidan och sätta i komponenten och skruva fast dem med hjälp av en tillhörande plastmutter på baksidan eftersom axeln på komponenterna var gängade, förutom displayen som tillfälligt tejpades fast, se figur 4.6 nedan för framsidan av skåpet. Knapparna i sig hade inga elektroniska kopplingar själva utan två olika typer av kontaktblock behövde införskaffas som enkelt monterades på baksidan av knapparna med hjälp av något som kallades för montageclip som klämdes på baksidan av knappen. Kontaktblocken som införskaffades var av typerna NO och NC som slöts eller öppnades när knappen trycktes in. NO användes till de tre vanliga knapparna och en NC användes till nödstoppet. Till dessa kontaktblock anslöts kablarna genom att skruva fast dem.



Figur 4.5: Baksida skåpslucka



Figur 4.6: Framsida av skåp för styrning av testbänk

Frekvensomriktaren monterades aldrig utanpå skåpet utan låg temporärt bredvid skåpet på grund av problemen som uppstod i slutet av arbetet. Till omriktaren anslöts trefaskablarna från kontaktorn och en jordkabel. Kablarna för den analoga styrsignalen från signalomvandlaren och för den digitala styrsignalen från reläet kopplades. Från frekvensomriktaren drogs trefaskablarna och jorden direkt till motorn som redan nämnts, hade D-kopplats.

4.2.3. Testbänk

Två av de tre givarna monterades på testbänken, en för varvräkning och en för bekräftelse av utlöst rusningsvakt. Den tredje givaren kunde inte monteras på grund av att det säkerhetslock som var tänkt för bänken inte hade producerats ännu, så denna givare fanns lös för att ändå

kunna simulera ett stängt lock genom att endast lägga en metallbit framför givaren. Givarna monterades temporärt, för testningens skull, med hjälp av buntband och överblivet metallskrot. Företaget kommer senare konstruera mer permanenta lösningar för montage av givare.

Testbänken var byggd av företaget. En motor var monterad på insidan av ena sidan av bänken och en axeltapp var monterad i mitten av bänken som bestod av en axel pressad i två rullager som aluminiumskivan sedan var monterad på. Ett remhjul var monterat på undersidan axeltappen och ett lika stort remhjul var monterat på motorns axel i samma storlek med en rem mellan dessa, se figur 4.7 nedan för en helhetsbild av testbänken utan lock. Detta gjorde att utväxlingen blev 1:1. Motorn kunde justeras i sidled för att kunna spänna remmen genom at lossa på fyra skruvar och bultar som höll fast motorn.

Figur 4.7: Testbänk utan säkerhetslock

4.3. Programmering

Ett program gjort i Arduino IDE består alltid av en setup-funktion där initieringar eller setup-funktioner tillämpas och denna funktion körs en gång i början. Ett program måste också bestå av en loop-funktion som är main-loopen och denna loopar så länge programmet är i gång. Utanför dessa funktioner kan nya funktioner skapas, bibliotek inkluderas eller globala variabler skapas. I detta avsnitt beskrivs först ett program som användes för test av display och givare och programmet för det klara systemet.

4.3.1. Test av display och givare

Den första delen av koden är till för utskrivning till LCD-displayen med hjälp av de olika funktionerna från LiquidCrystal-biblioteket. Detta med hjälp av de redan existerande funktionerna print(), setCursor(), begin() och clear(). Print() användes för att skriva ut till skärmen, setCursor() användes för att flytta på pekaren där utskrivningen startade, begin() användes för att starta upp skärmen och clear() rensade skärmen, se bilaga 1–7 för hur de används. För att dessa funktioner skulle kunna användas behövdes biblioteket inkluderas i koden genom att använda #include. För att enklare kunna programmera och läsa programmet sattes numren för alla pins kopplade till displayen, som krävde output från Arduinon, till variabler med samma namn för vad de utförde för funktion till displayen, som enable, registerval och mera. En display initierades med namnet för att kunna anropa funktioner till displayen kopplade till utgångarna. Se figur 4.8 nedan för initiering av skrivning till LCD-display.

```
#include <LiquidCrystal.h>
const int rs = 42, en = 44, d4 = 46, d5 = 48, d6 = 50, d7 = 52;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);
```

Figur 4.8: Initiering för användning av LCD-display för Arduino

För att testa en av de induktiva givarna med hjälp av displayen skrevs ett mycket enkelt program som räknade antalet gånger givaren hade lösts ut och visade det talet på displayen. Detta gjordes genom att skapa en global variabel count som ökade med ett varje gång givaren löste ut och denna variabel skrevs ut på displayen på andra raden, se figur 4.9 nedan där ”giv” är en variabel för den digitala ingången för givaren. För att detta skulle fungera behövdes funktionen pinMode() implementeras i setup-funktionen och ange ”giv” som en INPUT_PULLUP. Varför INPUT_PULLUP användes i stället för INPUT var för att få användning av den inbyggda pullup-resistorn i stället för att krångla till det med en extern pullup- eller pulldown-resistor. Detta för att det ska vara säkra värden på 5V eller 0V eftersom värden har en tendens till ”flyta” på grund av att kontakten inte är kopplad till något (LOW har samma mening som FALSE i andra programmeringsspråk), se bilaga 1 för hela programmet.

```
if(digitalRead(giv) == LOW) {
  count++;
  while(digitalRead(giv) == LOW) {
    lcd.setCursor(0, 1);
    lcd.print(count);
  }
}
```

Figur 4.9: Kod för att räkna antalet pulser som mätes av induktiv givare

4.3.2. Slutgiltiga programmet

Precis som för skapande av variabler för utgångar till displayen skapades variabler för alla utgångar och ingångar till all utrustning från och till Arduinon. Detta gjordes utanför alla funktioner. Dessa var variabler för displayen, knapparna, lamporna, potentiometern, givarna, reläerna, styrsignal till VFD och konstant kontrast på skärmen. Övriga mindre självklara globala variabler som skapades var två bools, active och notActive, som var variabler för LOW och HIGH för att förenkla förståelse i koden. En variabel för räkning av varvtal,

variabel för antalet mätpunkter som varvtalsräknaren kan mäta på, ett initieringsvärde för nuvarande varvtal och två variabler för tidräkning i mikrosekunder, se figur 4.10 nedan för hur initiering av alla variabler skedde.

```
// Initiera LCD-display och tillhörande
#include <LiquidCrystal.h>                                      // Inkludera LCD-biblioteket
const int rs = 42, en = 44, d4 = 46, d5 = 48, d6 = 50, d7 = 52;      // Sätt LCD-pins till variablar
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);                      // Aktivera biblioteket

// Initiera pins givare
const int giv1 = 53, giv2 = 51, giv3 = 3;                       // Sätt som variablar

// Initiera pins knapp, LED, potentiometer
const int start = 45, stopp = 2, driftLed = 39, utlostLed = 35,
stoppLed = 33, potmeter = A15;                                  // Sätt som variablar

// Initiera pins relä
const int relayCON = 23, relayVFD = 37;                         // Sätt som variablar

// Initiera pin VFD
const int VFD = 9;                                              // Sätt som variabel

// Initiera variablar
const bool active = LOW;                // Variablar för aktiv eller ej aktiv
const bool notActive = HIGH;
bool RPMSensor = notActive;             // Variabel för varvräkning
int measurePoints = 1;                  // Antal mätpunkter för varvtal
int currentRPM = 0;                     // Nuvarande varvtal

unsigned long newMicros;                // Tider för varvräkning
unsigned long lastMicros = 1;

// Initiera kontrast för LCD
const int kontrast = 5;                 // Variabel för kontrast
```

Figur 4.10: Initiering av variabler och pins för slutprogrammet i Arduino

För att använda alla in- och utgångar och LCD-displayen krävdes det nödvändiga initieringar av dessa i setup-funktionen. Initiering av in-och utgångar för alla pins där lamporna, reläerna och VFD-styrsignalen var utgångar och givarna och knapparna var ingångar. Ingångarna använde de inbyggda pullup-resistorerna och initierades med pinMode(), se det övre stycket av figur 4.11 nedan. Start av skärm gjordes med lcd.begin(), det skrevs ut en starttext till displayen med lcd.print() och en konstant kontrast sattes genom att skriva en PWM-signal till displayens kontrast-ingång med funktionen analogWrite(), se det nedre stycket av figur 4.11 för start av skärm. AnalogWrite() skrev ut en analog signal med hjälp av PWM till vald utgång som ett värde mellan 0–255 där 0 var 0V och 255 var 5V, se sista raden i figur 4.11 nedan (Värdet 75 testades fram).

```
void setup() {
  // Initiera alla pins som INPUT_PULLUP/OUTPUT
  pinMode(giv1, INPUT_PULLUP), pinMode(giv2, INPUT_PULLUP), pinMode(giv3, INPUT_PULLUP);
  pinMode(start, INPUT_PULLUP), pinMode(stopp, INPUT_PULLUP), pinMode(utlostLed, OUTPUT),
  pinMode(driftLed, OUTPUT), pinMode(stoppLed, OUTPUT);
  pinMode(relayCON, OUTPUT), pinMode(relayVFD, OUTPUT);
  pinMode(VFD, OUTPUT);

  // Initiera display, skriv och sätt kontrasten
  lcd.begin(16,2);                              // Starta display
  lcd.print("Close lid");                       // Skriv ut "Close Lid"
  analogWrite(kontrast, 75);                    // Välj kontrast
}
```

Figur 4.11: Setup-funktionens uppbyggnad i slutprogrammet för Arduino

I loop-funktionen skrevs huvudkoden. Nödvändiga lokala variabler infördes, så som den analoga styrsignalen till frekvensomriktaren, ett börvärde motorn skulle rotera upp till, en variabel för värdet potentiometern gav, ett maxtillåtet varvtal och en variabel för att hålla koll på tidigare varvtal, se figur 4.12 nedan för initieringar.

```
int motorSignal = 0;      // Skapa variabel motorSignal = 0
int setPoint;             // Skapa variabel setPoint
int potSpeed;             // Skapa variabel potSpeed
double maxRPM = 1200;     // Skapa variabel maxRPM = 1200
int lastRPM = 0;          // Skapa variabel lastRPM = 0
```

Figur 4.12: Initiering av nödvändiga variabler för start av huvudkod

Det första som skulle hända när programmet startade var att tända lampan för att visualisera om locket var öppet/motorn ej roterade vilket gjordes med digitalWrite() som skrev ut en digital signal till vald utgång. Nästa steg i programmet skulle undersöka om locket var stängt och om locket var stängt gick det vidare till nästa steg. Detta gjordes med en if-sats och funktionen digitalRead() som läser och returnerar den digitala signal den läst av från vald ingång. Om locket var stängt skulle reläet för kontaktorn slutas, vilket matade ström och spänning till frekvensomriktaren. Den tända lampan skulle släckas och ”Choose speed:” skrevs ut på första raden med hjälp av en egenskriven funktion printFirstRow() och hoppade sedan in i en while-loop. While-loopen körs så länge startknappen inte blivit intryckt. I loopen kodades det så ett börvärde behövde väljas genom att värdet från potentiometern som ligger mellan 0–1023 omvandlades till ett värde mellan 0 och det maximalt tillåtna varvtalet. I detta fall var det maximalt tillåtna varvtalet 1200 varv/min. Värdet skrevs sedan ut på skärmen på andra raden med hjälp av en egenskriven funktion printsecondRow(). En egenskriven funktion isLidClosed() implementerades i loopen för att undersöka ifall locket öppnades under processen och skulle i så fall stoppa allt, se figur 4.13 nedan för hur denna del av koden skrevs.

```
digitalWrite(stoppLed, HIGH);          // Tänd stopplampa
if(digitalRead(givl) == active) {
  digitalWrite(relayCON, HIGH);        // Skicka 24V styrström till kontaktor
  digitalWrite(stoppLed, LOW);         // Släck stopplampa
  printFirstRow("Choose speed:");      // Skriv ut "Choose speed"
  while(digitalRead(start) == notActive) {
    potSpeed = chooseSpeed();             // Få värde från potentiometer
    setPoint = potSpeed*maxRPM/1023;      // Omvandla värdet till ett varvtal
    printSecondRow(setPoint);             // Skriv ut börvärdet
    isLidClosed(motorSignal, lastRPM, LOW);    // Kolla om locket är stängt
  }
```

Figur 4.13: Första delen av loop-funktionen av slutprogrammet för Arduino

Om startknappen hade blivit intryckt skulle programmet hoppa ut ur while-loopen och nästa steg av programmet skulle utföras. Här initierades nya variabler, en accelerationstidsvariabel och fyra tidsvariabler för tidräkning för acceleration och utskrift, se de olika variablerna av typen unsigned long och int i figur 4.14 nedan. Nästa steg i programmet var att sluta reläet för styrsignalen till att starta frekvensomriktaren, skriva ”Current Speed:” på första raden och ”0” på andra raden, tända driftlampan och starta tidräkningen för accelerationen och utskriften, se figur 4.14 nedan för hur detta gjordes.

```
int accTime = 5000  ;                 // Sätt en accelerationstid
unsigned long startMillisAcc;         // Skapa tidsvariablar för acceleration och utskrift
unsigned long currentMillisAcc;
unsigned long startMillisPrint;
unsigned long currentMillisPrint;
digitalWrite(relayVFD,HIGH);          // Skicka 24V styrsignal till VFD
printFirstRow("Current speed:");      // Skriv ut "Current speed"
printSecondRow(0);                    // Skriv ut 0 på andra raden
digitalWrite(driftLed, HIGH);         // Tänd driftlampa
startMillisAcc = millis();            // Starta tidräkning för acceleration och utskrift
startMillisPrint = millis();
```

Figur 4.14: Andra delen av loop-funktionen av slutprogrammet för Arduino

Huvudfunktionen av detta program var att styra en motor och läsa av varvtalet skivan snurrade med. Detta gjordes med en enkel while-loop som kördes så länge det nuvarande varvtalet var mindre än börvärdet. Sedan användes funktionerna, isLidClosed(), isStopPressed och isRusActivated() som undersökte om locket var stängt, om stoppknappen blivit nedtryckt och om rusningsvakten hade löst ut och den nuvarande tiden som programmet körts sparades. En if-sats användes för acceleration av motorn som hela tiden övervakade om en viss tid hade gått, om denna tid hade uppnåtts skulle den analoga signalen till frekvensomvandlaren ökas med ett steg och tidräkningen började om. Under tiden detta skedde behövde varvtalet beräknas och en egen funktion countSpeed() implementerades och en ny nuvarande tid för programmet drifttid sparades, se den övre halvan av koden i figur 4.15.

På grund av att funktionerna för utskrift och tillhörande till displayen tog relativt lång tid att utföra i jämförelse med andra funktioner märktes en bugg med systemet. Ibland kunde varvtalet som skrevs ut vara exakt hälften så stort som det riktiga varvtalet. Orsaken till detta är fortfarande okänt. Men som lösning till problemet implementerades en if-sats som gjorde att varvtalet endast skrevs ut varje 200 millisekund och en annan if-sats inuti den första som övervakade om det nuvarande varvtalet var större än det förra. Då skrevs varvtalet ut och det nuvarande varvtalet sparades som det förra, se den nedre halvan av koden i figur 4.15.

```
while(currentRPM < setPoint) {
  isLidClosed(motorSignal, lastRPM, HIGH);        // Kolla om locket är stängt
  isStopPressed(motorSignal, lastRPM);      // Kolla om stoppknappen blivit tryckt
  isRusActivated(motorSignal, lastRPM);     // Kolla om rusningsvakten slagit ut
  currentMillisAcc = millis();              // Kolla nuvarande tid
  if(currentMillisAcc - startMillisAcc >= accTime) {
    motorSignal++;                           // Öka styrsignalen till vFD
    signalVFD(motorSignal);                  // Skicka styrsignal till VFD
    startMillisAcc = currentMillisAcc;       // Sätt ny tid
  }
  countSpeed();                             // Beräkna varvtal
  currentMillisPrint = millis();            // Sätt ny tid
  if(currentMillisPrint - startMillisPrint >= 200){
    if(currentRPM > lastRPM) {
      printSecondRow(currentRPM);       // Skriv ut nuvarande varvtal
      lastRPM = currentRPM;             // Spara det senaste varvtalet
    }
    startMillisPrint = currentMillisPrint;      // Sätt ny tid
  }
}
```

Figur 4.15: Tredje delen av loop-funktionen för slutprogrammet för Arduino

Ifall det maximala varvtalet som ställts in i val av varvtal skulle uppnås skall processen stängas ner och programmet skulle kunna startas om. För detta användes endast egna funktioner. ”Max RPM Reached” skrevs på första raden, varvtalet som uppnåddes på andra raden och sedan hoppade den till funtionen done() som innebar att processen var klar, se figur 4.16. Done() stoppade motorn, tände och släckte nödvändiga lampor och när motorn stannat stängde den av kontaktor och frekvensomriktare, se figur 4.17 nedan för hur den funktionen var uppbyggd.

```
printFirstRow("Max RPM reached");     // Skriv ut "Max RPM reached"
printSecondRow(currentRPM);           // Skriv ut nuvarande varvtal
done(motorSignal, lastRPM, LOW);      // Klar
```

Figur 4.16: Fjärde delen av loop-funktionen för slutprogrammet för Arduino

```
void done(int motorSignal, int lastRPM, bool stoppMotor) {
  printFirstRow("Please wait");          // Skriv ut "Please wait"
  while(currentRPM < lastRPM-10) {
    countSpeed();                        // Räkna varvtalet om det nya varvtalet är mindre än den förra varvtalet - 10
  }
  if(digitalRead(givl) == notActive) {
    printFirstRow("Warning lid open");    // Skriv ut "Warning lid open"
    if(stoppMotor == HIGH) {
      lcd.setCursor(0,1);                  // Flytta pekaren längst ner vänster
      lcd.print("stopping motor");         // Skriv ut "stopping motor"
    }
  } else {
    printFirstRow("Speed attained: ");    // Skriv ut "Speed attained"
    printSecondRow(currentRPM);           // Skriv ut varvtal
  }
  digitalWrite(driftLed, LOW);           // Stäng av driftlampa
  digitalWrite(stoppLed, HIGH);          // Sätt igång stoppLampa
  while(motorSignal > 0) {
    motorSignal = motorSignal - 1;       // Minska styrsignalen till VFD
    delay(500);                          // Vänta 500ms
    signalVFD(motorSignal);              // Skicka signal till VFD
  }
  motorSignal = 0;                       // Sätt styrsignal = 0
  signalVFD(motorSignal);                // Skicka signal till VFD
  digitalWrite(relayCON, LOW);           // Bryt 24V styrström till kontaktor
  digitalWrite(relayVFD, LOW);           // Bryt 24V styrström till VFD
  while(1) {}
}
```

Figur 4.17: Funktionen Done() som anropas för avslutning av programmet

Under arbetets gång skapades många egna funktioner för att underlätta läsning och den allmänna funktionen av koden. De funktioner var printFirstRow() som användes för att skriva ut en text på första raden. PrintSecondRow() användes för att skriva ut ett tal på andra raden. ChooseSpeed() hämtade värdet från potentiometern. SignalVFD() skickade den analoga styrsignalen till frekvensomriktaren. CountSpeed() beräknade varvtalet på skivan genom att mäta tiden av ett varv i nanosekunder och omvandla det till enheten varv/min. Done() var funktionen som anropades när programmet var klart, här saktades motorn ner, reläerna bröts, driftlampan släcktes och stopplampan tändes när motorn hade stannat. IsLidClosed, isStopPressed och isRusActivated har redan förklarats, se bilaga 2–6 för hela programmet.

4.4. Tester

Här beskrivs de slutgiltiga testerna som gjordes med och utan aluminiumskivan för att undersöka att det slutgiltiga programmet fungerade som det skulle.

4.4.1. Test av motor

De allra första testerna som gjordes var att bekräfta om systemet fungerade som det var tänkt genom att välja ett varvtal och endast låta motorn rotera upp till det varvtalet utan remmen monterad. Varvtalet mättes medhjälp av en skruv som skruvades fast i motorns remskiva som en induktiv givare läste av för varje varv som gick.

4.4.2. Test av system med och utan skiva monterad

När det kunde säkerställas att system fungerade som det skulle -kunde skivan och remmen monteras och programmet testades igen genom att ställa in låga varvtal på grund av säkerhetsskäl eftersom inget säkerhetslock ännu fanns som redan nämnt. När det fastställdes att låga varvtal inte var några problem provades det att rotera upp skivan i 1200 varv/min med försiktighetsåtgärder. Detta fungerade dock inte eftersom frekvensomriktarens motorskydd slog ut och processen stoppades vid ca 400 varv/min. Skivan demonterades och ett nytt test gjordes för att reda ut varför motorn stoppades.

Vid de varvtal som processen gick att köra på användes stoppknappen först för att simulera utlösningsmekanismen men sedan monterades även rusningsvakten på skivan med tillhörande motvikt och givaren så att rusningsvakten kunde testas.

5. Resultat

Hela arbetet resulterande i något som inte var helt förväntat. Det verkade som att motorn, som hade valts utav företaget var feldimensionerad, vilket resulterade i att motorn inte orkade rotera aluminiumskivan med en högre rotationshastighet än ca 400 varv/min. Då rusningsvakten skulle kunna lösas ut på ett maxvarvtal av 1200 varv/min kunde fullständiga tester inte genomföras för systemet. Tester för rusningsvakten upp till 300 varv/min gjordes i stället för att klargöra att systemet fungerade som det var konstruerat. Rusningsvakten monterades tillsammans med en motvikt placerad på andra sidan skivan motsvarande samma avstånd som rusningsvakten, men på grund av att det inte fanns något datablad eller liknande på hur rusningsvakten ställdes in för ett visst varvtal på en viss storlek av turbin ännu var det enda sättet att testa rusningsvakten genom att prova sig fram och varvtalet 200 varv/min valdes som mål. Efter några få prövningar var vakten inställd för ungefär 200 varv/min för den största storleken av turbin och tack vare takometern kunde utlösningsvarvtalet bekräftas med god säkerhet. Efter att fem tester gjordes kunde det bekräftas att utlösningsvarvtalen på de fem testerna låg på 199, 199, 200, 201 och 200 vilket tyder på en god säkerhet och en genomsnittlig tolerans på under ett varv/min. Vad som kunde ha påverkat varför varvtalet kunde skilja sig kunde varit noggrannheten av varvtalsräkningen eller mekanismen inuti vakten.

Med hjälp av takometern kunde varvtalet som systemet beräknade bekräftas med en mycket god säkerhet. På grund av metoden som varvtalet beräknades kunde varvtalet bli mindre noggrant vid högre varvtal eftersom tiden mellan gångerna systemet läste av en mätpunkt blev mycket kortare. Efter några få justeringar i koden kunde varvtalsräkningen testas på höga rotationshastigheter genom att demontera aluminiumskivan från den roterande axeln och endast rotera axeln och använda en skruv monterad på axeln som mätpunkt. Testet utfördes i tre stadier med tre tester i varje stadie där varvtalet kördes upp till någonstans mellan 1100 och 1200 varv/min. Sedan simulerades en utlösning av rusningsvakten med hjälp av stoppknappen och allt undersöktes även med takometern. Efter att dessa nio tester hade gjorts analyserades värdena systemet räknat ut i jämförelse med vad takometern visade och det kunde bekräftas att varvtalet systemet visade kunde skilja en till två varv/min.

6. Diskussion

Under arbetets gång uppkom ett flertal problem som blev tvungna att lösas under arbetsprocessen. Det största problemet som uppstod var att motorn inte klarade av att rotera aluminiumskivan till de rotationshastigheter som krävdes. En lösning till detta problem kunde inte ordnas under arbetsgången utan detta är något som kommer ordnas utav företaget senare där de två lösningarna kan vara att antingen uppgradera drivlinan, alltså byta ut motor och VFD, eller att öka storleken på remhjulet fäst i skivan. Det andra alternativet påverkar dock rotationshastigheten på genom att skivan roterar saktare än motorn. Detta innebär att avancerade omberäkningar kommer krävas om en rusningsvakt egentligen skall lösas ut på ett högre varvtal än skivan kan rotera. En fördel med större remhjul är dock att det är ett billigare alternativ än att byta ut motor och VFD eftersom remhjul och rem är mycket billigare än en motor och VFD.

Det finns fler sätt att beräkna varvtal på än att beräkna tiden för ett varv med en mätpunkt och använda tiden för att beräkna varvtalet. I början av arbetet mättes tiden för skivan att rotera ett varv i mikrosekunder i stället för nanosekunder vilket blev ett ganska allvarligt problem vid höga varvtal eftersom tiden det tog för skivan att rotera ett varv tog vid varvtalet 800 varv/min 20,8 µs och vid 820 varv/min 20,3 µs. Detta utgör ett stort problem eftersom en Arduino endast tolkar mikrosekunderna i heltal vilket innebar att 20,8 µs och 20,3 µs avrundades till 20 µs i koden. Detta innebar att varvtalet som visades på skärmen inte uppdaterades förens tiden för ett varv var 19 µs eller rättare sagt 19,9 µs eftersom detta avrundas till 19 µs. Detta innebar att när varvtalet nådde 833 varv/min uppdaterades inte det nya varvtalet förens det nådde 877 varv/min. Detta var ett stort problem eftersom varvtalet som visas på skärmen måste vara så nära det faktiska varvtalet som möjligt. En lösning på detta problem var att mäta tiden i nanosekunder i stället vilket innebar att mättiden ökade med en faktor 1000. Det faktiska varvtalet kunde då visas på displayen med en mycket högre noggrannhet. Ett mindre betydelsefullt problem som återstod med denna metod av beräkning av varvtal var att vid mycket låga varvtal, under 60 varv/min, uppdaterades varvtalet mycket långsamt, alltså för endast varje varv skivan roterade. Detta resulterade i ett dåligt representerat varvtal vid mycket låga varvtal. Dock eftersom rusningsvakten endast skulle lösa ut på högre varvtal har detta ingen påverkan på de viktiga resultaten. Om en lösning skulle implementeras på detta problem skulle den kunna vara att öka antalet mätpunkter som varvtalsgivaren läser av och ta tiden för ett varv dividerat med antalet punkter. Detta skulle innebära att varvtalet uppdateras mycket snabbare men problemet med fler mätpunkter är att tiden mellan punkterna blir kortare och då kunde problemet som diskuterades tidigare återkomma. Ett annat sätt att beräkna varvtalet på är att räkna ett antal punkter på skivan under en viss tid. Problemet med detta är i stället att det krävs fler mätpunkter än en om ett noggrant varvtal vill fås. Varvtalet kan heller inte uppdateras snabbare än tiden som varvtalet beräknas och vid lägre mättid krävs också fler mätpunkter.

Ett annat ganska allvarligt problem som uppstod i slutet av arbetet var att den ursprungliga varvtalsräkningen på varvtal över 400 varv/min blev mindre och mindre noggrann. När testet med demonterad skiva gjordes klargjordes det att upp på över 1000 varv/min kunde det skilja över 20 varv/min än vad det faktiska varvtalet var. Detta var på grund av att tiden som mättes per varv mättes från början i millisekunder och vid så höga varvtal skiljde sig tiden endast med decimaler och detta innebar att varvtalet som visades på skärmen inte var det riktiga varvtalet. Lösningen på detta problem var bara att ändra till att mäta i mikrosekunder. En

annan lösning till varvtalsräkningsproblemet kunde varit att byta ut metoden för att beräkna varvtalet till att mäta punkter under en viss tid i stället för tiden per varv.

Ur ett hållbart perspektiv kan det kortsiktigt anses att miljön påverkas negativt på grund av produktion av de olika komponenterna som krävs till rusningsvakten. Arbetet kan dock anses som mer positivt påverkande på miljön än negativt. Trots utsläpp vid produktion, som kan påverkas positivt beroende på framställning av energin mm, gör denna rusningsvakt mer gott än ont mot miljön. Eftersom denna rusningsvakt skall kunna etableras i vattenkraftverk runt om i världen kommer stora olyckor kunna förhindras och färre delar kommer behövas bytas ut på grund av förslitning. Detta innebär mindre utsläpp vid produktion av komponenter det vill säga lager och andra nödvändiga komponenter. Vattenkraftverken kommer troligen också kunna köras säkrare med tanke på att om ett fel skulle uppstå och rusningsvakten löses ut och vattnet stoppas behövs i grunden endast felet som orsakade ruset fixas. Detta innebär snabbare och mindre reparation och verken kan köras i gång snabbare vilket innebär ökad produktion av miljövänligare el.

Referenser

[1] Arduino, ”Arduino Mega 2560 Rev3”, 2021. [Online]. Tillgänglig: https://store.arduino.cc/arduino-mega-2560-rev3 (Hämtad: 2021-04-10)

[2] Kinney T.A. Proximity sensors Compared: Inductive, Capcative, Photelectric and Ultrasonic [internet]. Machine Design; 2001 [2021-02-13]. Tillgänglig från: https://www.machinedesign.com/automation-iiot/sensors/article/21831577/proximity-sensors-compared-inductive-capacitive-photoelectric-and-ultrasonic

[3] N. Ida, Sensors, *Actuators, and their Interfaces*: *A Multidisciplinary Indtroduction*. I. uppl., Edison , USA: SciTech Publishing, 2014. [Online]. Tillgänglig: https://app.knovel.com/web/view/khtml/show.v/rcid:kpSATIAMI2/cid:kt0110NRA3/viewerType:khtml//root_slug:sensors-actuators-their/url_slug:inductive-proximity-sensors?&kpromoter=federation&page=26&view=collapsed&zoom=1, Hämtad: 2021-04-10.

[4] M.J. Brandt, K.M. Johnson, A.J. Elphinston, D.D. Ratnayaka, *TWORT'S WATER SUPPLY*, VII. uppl., Elsevier , 2016. [Online]. Tillgänglig: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=4673329, Hämtad: 2021-06-03

[5] S. Dunning, L.S. Katz, *Energy calculations & problem solving sourcebook: a practical guide for the Certified Energy Manager Exam.* I. uppl,. Lilburn, USA: The Fairmont Pres, Inc, 2017. [Online]. Tillgänglig: https://app.knovel.com/web/toc.v/cid:kpECPSSAP3/viewerType:toc/root_slug:energy-calculations-problem?kpromoter=federation , Hämtad 2021-06-27

[6] N. Jenkins, J.B. Ekanayake, G. Strbac, *Distributed Generation*. I. uppl,. London, Storbritannien: Institution of Engineering and Technology, 2010. [Online]. Tillgänglig: https://app.knovel.com/web/toc.v/cid:kpDG000002/viewerType:toc//root_slug:distributed-generation-2/url_slug:front-matter?issue_id=kpDG000002&hierarchy= , Hämtad 2021-06-03

[7] S. Doty, W.C. Turner, *Energy magagement handbook*. VIII. Uppl., Lilburn, USA: The Fairmont Pres, Inc, 2013. [Online]. Tillgänglig: https://app.knovel.com/web/view/khtml/show.v/rcid:kpEMHE0003/cid:kt00C19FM1/viewerType:khtml//root_slug:energy-management-handbook/url_slug:adjustable-speed-drives?&kpromoter=federation&page=18&view=collapsed&zoom=1, Hämtad: 2021-06-04.

[8] B.L. Capeheart, W.C. Turner, W.J. Kennedy, *Guide to Energy Management*. IIX. uppl., Lilburn, USA: The Fairmont Press, Inc, 2016. [Online]. Tillgänglig: https://app.knovel.com/web/view/khtml/show.v/rcid:kpGEME0011/cid:kt010WWLY3/viewerType:khtml//root_slug:78-using-variable-frequency-drives-vfds/url_slug:using-variable-frequency?kpromoter=federation&toc-within=VFD&b-toc-cid=kpGEME0011&b-toc-root-slug=&b-toc-url-slug=using-variable-frequency&b-toc-title=Guide%20to%20Energy%20Management%20(8th%20Edition)&b-toc-within=VFD&page=15&view=collapsed&zoom=1&q=VFD, Hämtad: 2021-05-10

[9] J.T. Warner, *The handbook of Lithium-Ion Battery Pack Design: Cemistry Components, Types and Terminology*. I. uppl., Midland, USA: Elsevier, 2015. [Online]. Tillgänglig:

https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=2056916#, Hämtad: 2021-04-12

[10] D.R. Patrick, S.W. Fardo, *Rotating Electrical Machines and Power Systems*. II. uppl., Lilburn, USA: The Fairmont Press, Inc, 1997. [Online]. Tillgänglig: https://app.knovel.com/web/view/pdf/show.v/rcid:kpREMPSE02/cid:kt007DSIB2/viewerType:pdf//root_slug:rotating-electrical-machines/url_slug:motor-overload-protective?&kpromoter=federation, Hämtad: 2021-05-10.

[11] *PC817X Series*: Sharp, 2003. [Online]. Tillgänglig: https://www.farnell.com/datasheets/73758.pdf, Hämtad: 2021-05-18.

[12] T-M.I. Bajenescu, M.I. Bazu, *Component Reliability for Electronic Systems*. I. uppl., Norwood, USA: ARTECH HOUSE, 2009. [Online]. Tillgänglig: https://app.knovel.com/web/view/khtml/show.v/rcid:kpCRES0001/cid:kt011O9ZT1/viewerType:khtml//root_slug:component-reliability/url_slug:component--introduction-4?&kpromoter=federation&page=13&view=collapsed&zoom=1, Hämtad: 2021-06-03

[13] A.S. Morris, R. Langari *Measurements and Instrumentation: Theroy and Apllication*. I. uppl.: Elsevier, 2011. [Online]. Tillgänglig: https://ebookcentral.proquest.com/lib/chalmers/reader.action?docID=4731588, Hämtad: 2021-05-10.

[14] T. Wilmshurst, *Designing Embedded Systems with PIC Microcontrollers: Principals and Applications*. II. uppl., Derby , Storbritannien: Elsvier, 2010. [Online]. Tillgänglig: https://app.knovel.com/web/toc.v/cid:kpDESPICM7/viewerType:toc//root_slug:designing-embedded-systems/url_slug:hd44780-lcd-driver-its?issue_id=kt0092LOS3&hierarchy=, Hämtad: 2021-05-23.

Bilagor

Bilaga 1. Program för test av display och givare

```
// Initiera LCD-display och tillhörande
#include <LiquidCrystal.h>
const int rs = 42, en = 44, d4 = 46, d5 = 48, d6 = 50, d7 = 52;
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);

// Initiera pins givare
const int giv = 2

// Initiera count-variabel för antal registrerade signaler av givare
int count = 0;

void setup() {
  // put your setup code here, to run once:
  pinMode(givl, INPUT_PULLUP)

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
  // Print a message to the LCD.
  lcd.print("Antal reg. giv:");
  lcd.setCursor(0, 1);
  lcd.print(count);
}

void loop() {
  if(digitalRead(giv) == LOW) {
    count++;
    while(digitalRead(giv) == LOW) {
      lcd.setCursor(0, 1);
      lcd.print(count);
    }
  }
}
```

Bilaga 2. Program för testbänk (Del 1)

```
// Initiera LCD-display och tillhörande
#include <LiquidCrystal.h>                                      // Inkludera LCD-biblioteket
const int rs = 42, en = 44, d4 = 46, d5 = 48, d6 = 50, d7 = 52;  // Sätt LCD-pins till variablar
LiquidCrystal lcd(rs, en, d4, d5, d6, d7);                      // Aktivera biblioteket

// Initiera pins givare
const int givl = 53, giv2 = 51, giv3 = 3;                       // Sätt som variablar

// Initiera pins knapp, LED, potentiometer
const int start = 45, stopp = 2, driftLed = 39, utlostLed = 35,
stoppLed = 33, potmeter = A15;                                  // Sätt som variablar

// Initiera pins relä
const int relayCON = 23, relayVFD = 37;                         // Sätt som variablar

// Initiera pin VFD
const int VFD = 9;                                              // Sätt som variabel

// Initiera variablar
const bool active = LOW;                  // Variablar för aktiv eller ej aktiv
const bool notActive = HIGH;
bool RPMSensor = notActive;               // Variabel för varvräkning
int measurePoints = 1;                    // Antal mätpunkter för varvtal
int currentRPM = 0;                       // Nuvarande varvtal

unsigned long newMicros;                  // Tider för varvräkning
unsigned long lastMicros = 1;

// Initiera kontrast för LCD
const int kontrast = 5;                   // Variabel för kontrast

void setup() {
  // Initiera alla pins som INPUT_PULLUP/OUTPUT
  pinMode(givl, INPUT_PULLUP), pinMode(giv2, INPUT_PULLUP), pinMode(giv3, INPUT_PULLUP);
  pinMode(start, INPUT_PULLUP), pinMode(stopp, INPUT_PULLUP), pinMode(utlostLed, OUTPUT),
  pinMode(driftLed, OUTPUT), pinMode(stoppLed, OUTPUT);
  pinMode(relayCON, OUTPUT), pinMode(relayVFD, OUTPUT);
  pinMode(VFD, OUTPUT);

  // Initiera display, skriv och sätt kontrasten
  lcd.begin(16,2);                              // Starta display
  lcd.print("Close lid");                       // Skriv ut "Close Lid"
  analogWrite(kontrast, 75);                    // Välj kontrast
}
```

Bilaga 3. Program för testbänk (Del 2)

```
// Skriver ut text på första raden
void printFirstRow(String string) {
  lcd.clear();              // Töm skärmen
  lcd.setCursor(0, 0);      // Flytta pekaren till högst upp vänster
  lcd.print(string);        // Skriv ut
}

// skriver ut ett tal på andra raden
void printSecondRow(int number) {
  lcd.setCursor(0, 1);                          // Flytta pekaren längs ner vänster
  if(number < 10) {
    lcd.print(number);                          // Skriv ut
    lcd.setCursor(1, 1);                        // Flyta pekaren ett steg till höger
    lcd.print("               ");               // Skriv ut tomrum
  } else if(number < 100 && number >= 10) {
    lcd.print(number);                          // Skriv ut
    lcd.setCursor(2, 1);                        // Flytta pekaren ett steg till höger
    lcd.print("               ");               // Skriv ut tomrum
  } else if(number < 1000 && number >= 100) {
    lcd.print(number);                          // Skriv ut
    lcd.setCursor(3, 1);                        // Flytta pekaren ett steg till höger
    lcd.print("               ");               // Skriv ut tomrum
  }
  lcd.print(number);                            // Skriv ut
}

// Returnerar värdet från potentiometern
int chooseSpeed() {
  return analogRead(potmeter);  // Returnerar värdet givet från potentiometer 0-1023
}

// Skriver styrsignalen till VFD
void signalVFD(int VFDSignal) {
  analogWrite(VFD, VFDSignal);  // Skriv ut styrsignal till VFD
}
```

Bilaga 4. Program för testbänk (Del 3)

```
// Räknar det nuvarande varvtalet
void countSpeed() {
  if(digitalRead(giv2) == active) {
    if(RPMSensor == notActive) {
      newMicros = micros();                          // Ny tid
      currentRPM = 60000000/(newMicros - lastMicros)/measurePoints;  // Räkna ut varvtal
      lastMicros = newMicros;                        // Sätt den nya tiden till den gamla tiden
      RPMSensor = active;                            // Sätter den som klarräknad
    }
  }
  if(digitalRead(giv2) == notActive) {
    RPMSensor = notActive;                           // Sätter den som redo för att räkna på nytt
  }
}

// Det sista systemet gör innan den är klar
void done(int motorSignal, int lastRPM, bool stoppMotor) {
  printFirstRow("Please wait");          // Skriv ut "Please wait"
  while(currentRPM < lastRPM-10) {
    countSpeed();                        // Räkna varvtalet om det nya varvtalet är mindre än den förra varvtalet - 10
  }
  if(digitalRead(giv1) == notActive) {
    printFirstRow("Warning lid open");    // Skriv ut "Warning lid open"
    if(stoppMotor == HIGH) {
      lcd.setCursor(0,1);                   // Flytta pekaren längst ner vänster
      lcd.print("stopping motor");          // Skriv ut "stopping motor"
    }
  } else {
    printFirstRow("Speed attained: ");    // Skriv ut "Speed attained"
    printSecondRow(currentRPM);           // Skriv ut varvtal
  }
  digitalWrite(driftLed, LOW);           // Stäng av driftlampa
  digitalWrite(stoppLed, HIGH);          // Sätt igång stoppLampa
  while(motorSignal > 0) {
    motorSignal = motorSignal - 1;       // Minska styrsignalen till VFD
    delay(500);                          // Vänta 500ms
    signalVFD(motorSignal);              // Skicka signal till VFD
  }
  motorSignal = 0;                       // Sätt styrsignal = 0
  signalVFD(motorSignal);                // Skicka signal till VFD
  digitalWrite(relayCON, LOW);           // Bryt 24V styrström till kontaktor
  digitalWrite(relayVFD, LOW);           // Bryt 24V styrström till VFD
  while(1) {}
}
```

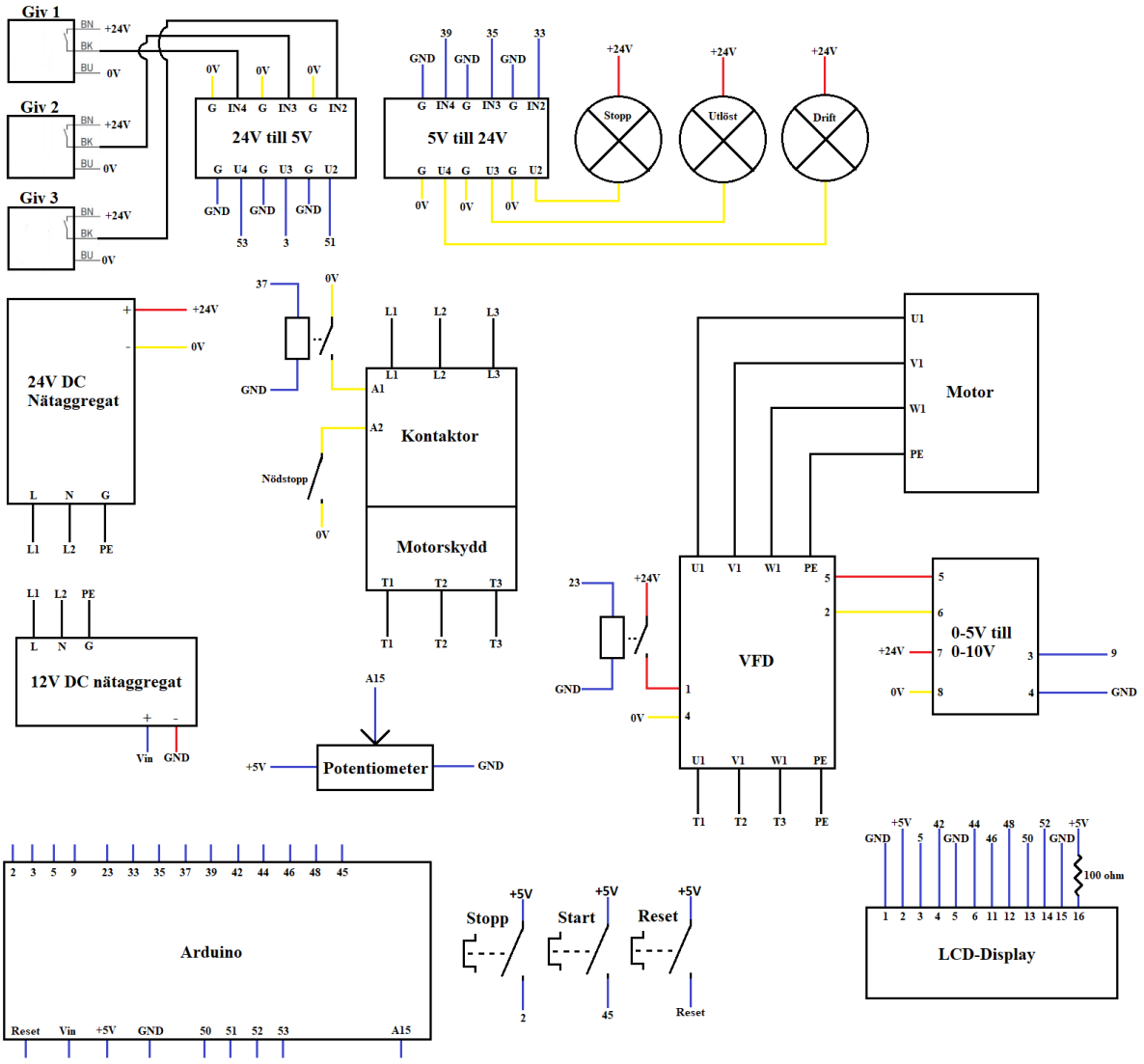
Bilaga 5. Program för testbänk (Del 4)

```
// Kollar om locket är stängt
void isLidClosed(int motorSignal, int lastRPM, bool stoppMotor) {
  if(digitalRead(givl) == notActive) {
    done(motorSignal, lastRPM, stoppMotor);           // Klar
  }
}

// Kollar om stoppknappen blibit nedtryckt
void isStopPressed(int motorSignal, int lastRPM) {
  if(digitalRead(stopp) == active) {
    done(motorSignal, lastRPM, LOW);       // Klar
  }
}

// Kollar om rus blivit aktiverat
void isRusActivated(int motorSignal, int lastRPM) {
  if(digitalRead(giv3) == active) {
    digitalWrite(utlostLed, HIGH);    // Tänd utlöstlampan
    done(motorSignal, lastRPM, LOW);       // Klar
  }
}
```

Bilaga 6. Program för testbänk (Del 5)

```
// Huvudfunktionen
void loop() {
  int motorSignal = 0;      // Skapa variabel motorSignal = 0
  int setPoint;             // Skapa variabel setPoint
  int potSpeed;             // Skapa variabel potSpeed
  double maxRPM = 1200;     // Skapa variabel maxRPM = 1200
  int lastRPM = 0;          // Skapa variabel lastRPM = 0
  digitalWrite(stoppLed, HIGH);         // Tänd stopplampa
  if(digitalRead(givl) == active) {
    digitalWrite(relayCON, HIGH);       // Skicka 24V styrström till kontaktor
    digitalWrite(stoppLed, LOW);        // Släck stopplampa
    printFirstRow("Choose speed:");     // Skriv ut "Choose speed"
    while(digitalRead(start) == notActive) {
      potSpeed = chooseSpeed();             // Få värde från potentiometer
      setPoint = potSpeed*maxRPM/1023;      // Omvandla värdet till ett varvtal
      printSecondRow(setPoint);             // Skriv ut börvärdet
      isLidClosed(motorSignal, lastRPM, LOW);    // Kolla om locket är stängt
    }
    int accTime = 5000  ;               // Sätt en accelerationstid
    unsigned long startMillisAcc;       // Skapa tidsvariablar för acceleration och utskrift
    unsigned long currentMillisAcc;
    unsigned long startMillisPrint;
    unsigned long currentMillisPrint;
    digitalWrite(relayVFD,HIGH);        // Skicka 24V styrsignal till VFD
    printFirstRow("Current speed:");    // Skriv ut "Current speed"
    printSecondRow(0);                  // Skriv ut 0 på andra raden
    digitalWrite(driftLed, HIGH);       // Tänd driftlampa
    startMillisAcc = millis();          // Starta tidräkning för acceleration och utskrift
    startMillisPrint = millis();
    while(currentRPM < setPoint) {
      isLidClosed(motorSignal, lastRPM, HIGH);        // Kolla om locket är stängt
      isStopPressed(motorSignal, lastRPM);      // Kolla om stoppknappen blivit tryckt
      isRusActivated(motorSignal, lastRPM);     // Kolla om rusningsvakten slagit ut
      currentMillisAcc = millis();              // Kolla nuvarande tid
      if(currentMillisAcc - startMillisAcc >= accTime) {
        motorSignal++;                          // Öka styrsignalen till vFD
        signalVFD(motorSignal);                 // Skicka styrsignal till VFD
        startMillisAcc = currentMillisAcc;      // Sätt ny tid
      }
      countSpeed();                             // Beräkna varvtal
      currentMillisPrint = millis();            // Sätt ny tid
      if(currentMillisPrint - startMillisPrint >= 200){
        if(currentRPM > lastRPM) {
          printSecondRow(currentRPM);       // Skriv ut nuvarande varvtal
          lastRPM = currentRPM;             // Spara det senaste varvtalet
        }
        startMillisPrint = currentMillisPrint;      // Sätt ny tid
      }
    }
    printFirstRow("Max RPM reached");     // Skriv ut "Max RPM reached"
    printSecondRow(currentRPM);           // Skriv ut nuvarande varvtal
    done(motorSignal, lastRPM, LOW);      // Klar
  }
}
```

Bilaga 7. Enkelt kopplingsschema av system



INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2021
www.chalmers.se

CHALMERS