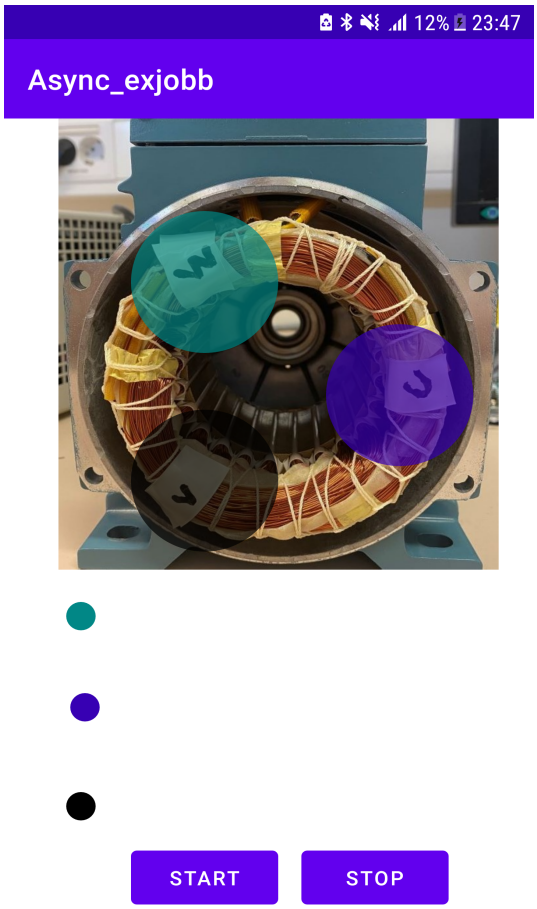


Demonstrator av flöde i en trefasmotor

Examensarbete inom högskoleprogrammet Elektroteknik

Malek Amer Abshir Mohamed INSTITUTIONEN FÖR ELEKTROTEKNIK
CHALMERS TEKNISKA HÖGSKOLA
Göteborg 2023
www.chalmers.se

Examensarbete 2023

Demonstrator av flöde i en trefasmotor

Malek Amer Abshir Mohamed

CHALMERS

Institutionen för Elektroteknik
Chalmers Tekniska Högskola
Göteborg 2023

Demonstrator av flöde i en trefasmotor
Malek Amer, Abshir Mohamed

Handledare: Sakib Sistek, Data och Informationsteknik
Examinator: Stefan Lundberg, Elektroteknik

Examensarbete 2023
Institutionen för Elektroteknik
Chalmers Tekniska Högskola
SE-412 96 Göteborg
Telefon +46 31 772 1000

Skriven i LaTeX
Göteborg 2023

Sammanfattning

Detta examensarbete utfördes i skolan under våren 2023. Syftet med arbetet var att skapa en demonstrator i form av en applikation som skulle visa hur det magnetiska-flödet rör sig i en asynkronmotor.

Denna projektidé är inte alltför populär ute på marknaden och därmed finns det mycket plats för potentiell utveckling i detta fält. Även fast flödesrörelsen i en elmotor har studerats noga kan det vara svårt att illustrera hur det faktiskt ser ut under drift i realtid. Där kommer denna lösning in i bilden. Projektet kan vara ett verktyg för att utveckla den pedagogiska aspekten inom detta specifika ämne. Det gör det även allmänt lättare att studera en motor och dess egenskaper.

I projektet användes halleffektsensorer då de mäter flöde. De kopplades till en Arduino som hanterar mätvärdena och sedan skickar över värdena via Blåtand till applikationen. Applikationen programmerades i Android studio som använder språket Java. Hur flödet skulle demonstreras i applikationen var en beslutsprocess som ändrats under arbetets gång. Detta beror på att saker tar mer tid än planerat och det påverkar vad som hinner implementeras.

Flödet i lindningarna representeras av en cirkel vid varje lindning som ändras i storlek baserat på värdet av flödet. Varje sensor kommer att ligga på varsin lindning för att mäta upp flödesförändringen. Cirklarna används då de tydligt kan visa ökad positivt flöde genom att expandera och minskat negativt flöde genom att minska. Eftersom flödet roterar i motorn kommer den positiva och negativa delen även rotera och detta visar olika storlekar av flöde vid olika tidpunkter. Målet var att visa det på ett sätt som var lättast att förstå med den befintliga tiden. Arbetet resulterade i en väl fungerande applikation med små fördröjningar. Applikationen kan förbättra sin prestanda men även med fler funktioner. Mest tid i arbetet lades på att få över värdena från Arduinon till applikationen samt att separera den stringen som värdena ligger i när de skickas över.

Abstract

This degree project was carried out at school in the spring of 2023. The purpose of this project was to create a demonstrator in the form of an application that would show how the magnetic field moves in a rotating induction motor.

This project idea is not too popular in the market, thus there is much room for potential development in this field. Even though the field movement in an electric motor has been studied carefully, it can be difficult to illustrate what it actually looks like in operation in real-time. That's where this solution comes into the picture. The project can be a tool for developing the educational aspect of this specific subject. It also makes it generally easier to study an electric machine and its characteristics.

The project use hall effect sensors as they measure magnetic field. They were connected to an Arduino that handles the measurements and then sends them via Bluetooth to the application. The application was programmed in Android Studio using the Java language. How the field would be illustrated in the application was a decision process that changed during the work. This was because it took more time than anticipated, which changed what could be implemented.

The magnetic field generated by the windings is presented as a circle at each winding that changes in size based on the value of the field. Each sensor will lie on a different winding to measure the change in the field. The circles are used because they can clearly show increased positive field by expanding and decreased negative field by decreasing. Since the field rotates in the motor, the positive and negative parts will also rotate, and this will show different sizes of field at different time points. The goal was to show it in a way that was easiest to understand with the existing time. The work resulted in a well-functioning application with small delays. The application can improve its performance but also with more features. Most time in the work was spent on getting the values from the Arduino to the application and separating the string that the values are in when they are sent over.

Förord

Examensarbetet ingår som ett moment på elektroingenjörsprogrammmet 180 HP på chalmers tekniska högskola och omfattar 15 HP. Arbetet utfördes I skolan.

Vi vill tacka vår examinator Stefan Lundberg och handledare Sakib Sistek för stödet och inspirationen till detta examensarbete. Det har varit en hjälpsam och framåthävande miljö som utvecklade oss.

Malek Amer och Abshir Mohamed, Göteborg, Juni 2023

Innehåll

1 Inledning **1**
- 1.1 Bakgrund . . . 1
- 1.2 Syfte . . . 1
- 1.3 Mål . . . 1
- 1.4 Avgränsningar . . . 1

2 Teknisk Bakgrund **3**
- 2.1 Arduino . . . 3
- 2.2 Blåtandsmodul . . . 4
- 2.3 Halleffektsensor . . . 5
- 2.4 Matlab . . . 5
- 2.5 Android Studio . . . 6
- 2.6 Asynkronmotor . . . 6

3 Metod **10**
- 3.1 Hårdvara för att visualisera flöde . . . 10
- 3.2 Förstudie . . . 11
- 3.3 Mätning av flöde . . . 11
- 3.4 Utveckling av applikation . . . 11
- 3.5 Tester för slutresultat . . . 12

4 Systemkonstruktion **14**
- 4.1 Asynkronmotor . . . 15
- 4.2 Sändarkrets . . . 16
- 4.3 Mottagare . . . 17

5 Resultat och Diskussion **22**

6 Slutsats **26**

Bibliography **27**

A Appendix **I**
- A.1 Sändarkrets kod . . . I
- A.2 Android Studio kod . . . II

1
Inledning

1.1 Bakgrund

Jordens magnetfält är ett fenomen som stöts på dagligen och utan det skulle liv på jorden vara obefintligt. Utan magnetfältet skulle människor utsättas av skadliga partiklar som härstammar från solen, så kallade solvindar [1]. Ett exempel är planeten Mars som på grund av förlusten av sitt magnetfält även förlorade sin atmosfär och detta har lett till en uttorkad planet som drabbas av globala dammstormar [1]. Trots att magnetfältet är nödvändigt för alla livsformer på jorden, är det något som inte kan observeras med blotta ögat.

Magnetfält används i flertal apparater och maskiner som till exempel asynkronmotorn. Magnetfältet är kärnpunkten för asynkronmotorns funktion då detta får rotorn i asynkronmotor att rotera. Även fast kunskap om hur flödet roterar i motorn finns så kan det vara svårt att visualisera hur det faktiskt ser ut i realtid. Att skapa ett program som kan hjälpa till med att visualisera hur ett magnetfält roterar i en asynkronmotor skulle därmed kunna vara ett akademiskt verktyg för att underlätta förståelsen av konceptet för studerande.

1.2 Syfte

Syftet med detta examensarbete är att utveckla ett program som kan visualisera det osynliga magnetfältet i en asynkronmotor. Detta för att underlätta förståelsen av hur ett roterande magnetfält skapas i elmotorer.

1.3 Mål

Målet är att utveckla en applikation som kan visualisera hur magnetfältet roterar i en asynkronmotor. Magnetfältet ska läsas av med hjälp av sensorer. Sensorerna ska placeras in i motorn och sedan skicka över värdena till applikationen. I applikationen ska magnetfältet visualiseras med hjälp av värdena på ett lättförståeligt sätt.

1.4 Avgränsningar

I examensarbete kommer en Arduino Nano att användas för att skicka värdena från sensorerna till applikationen och den programmeras med c++. Anledningen till att

just Arduino nano valdes var för dess storlek. För att kunna läsa av magnetfältvärdena inne i asynkronmotorn behövdes en liten mikrokontroller som får plats i det trånga utrymme som finns inne i maskinen och detta var möjligt med hjälp av Arduino nano. Denna mikrokontroller använts också eftersom gruppen är bekanta med den och har använt den i tidigare projekt. För att ta emot värdena används en applikation som skapas med hjälp av Android studio som använder programmeringsspråket Java.

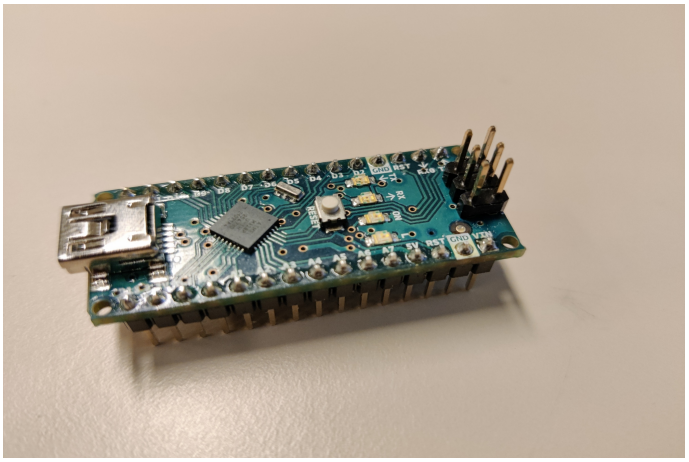
Systemet kommer bara att använda sig av en sensor för varje lindning för att underlätta konstruktionen. Med fler sensorer på varje lindning kan man få ett mer exakt svar på värdet men för enkelhetens skull används det inte här.

2
Teknisk Bakgrund

I detta kapitel beskrivs verktygen och komponenterna som använts i projektet.

2.1 Arduino

Arduino är en av de mest populära mikrokontrollerna bland studenter världen över. Då Arduino skapades 2005 var mikrokontrollers mer riktade för ingenjörer [2]. Företagets mål var att göra programmering enkelt för studenter så att alla kan skapa sina egna projekt. Arduinon består av en mikrokontroller krets, vilket är i huvudsak en dator på ett chip [2]. I Figur 2.1 ser man en bild på en Arduino Nano som innehåller en processorkärna, minne och in- och utgångs kontroller [2].



Figur 2.1: Bild på Arduino nano kortet som används i examensarbetet.

Arduino har tre huvudfunktioner:

- Samla indata
- Processa data
- Skicka utdata

Indata är data som skickas in till Arduinon och det kan vara allt från on/off signaler, varierande spänningssignaler och kommunikation från en annan mikrokontroller, så länge spänningsintervallet är inom Arduinos gränser [2]. För att använda indata programmeras kortet. Programmering av en Arduino kan vara att tända en lampa eller någonting mer komplext som att ladda upp väderinformation till en hemsida [2]. Utdata är den signal som skickas från Arduino till en komponent. Utdata är

väldigt likt indata där det kan vara on/off signaler, varierande spänning signaler och kommunikation till en annan mikrokontroller [2].

2.2 Blåtandsmodul

Blåtandsmodulen, även känd som HC05, är en enhet som utbyter information via en trådlös kommunikation över en kort distans med en annan blåtandsmodul [3]. För att båda blåtandsmodulerna ska kommunicera med varandra använder de en serieport [3]. Serieporten kommer i två konfigurationer och dessa är master och slav [3]. Master modulen söker och kopplas till andra enheter, medans slav modulen inte kan kopplas till andra enheter på egen hand [3]. Blåtandmodulen kan endast koppla till enheter inom cirka 10 meter [3].

Figur 2.2: Bild på blåtandsmodulen HC05.

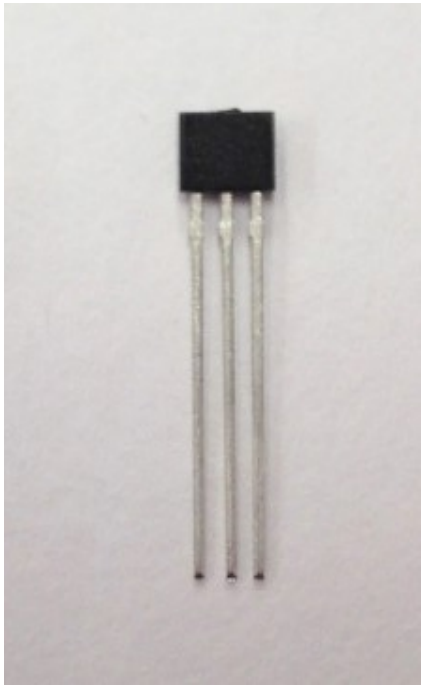
I Figur 2.2 visas blåtandsmodulen som används i arbetet och anslutningspinnarna från vänster till höger är:

- EN: Används för att komma in i blåtandsmodulens kommandoläge och det används för att ställa in specifika inställningar.
- VCC: Används för att försörja blåtandsmodulen med spänning.
- GND: Jordar blåtandsmodulen.
- TXD: Skickar data genom seriell kommunikation.
- RXD: Tar emot data genom seriell kommunikation.
- STATE: Kollar ifall modulen är kopplad till en annan modul.

Blåtandsmodulen kan använda ett bibliotek kallad AltSoftSerial. Detta ger blåtandsmodulen en extra serieport [4]. Med hjälp av AltSoftSerial är det möjligt att både ta emot och skicka ut data [4]. Detta använder pinnarna 8 och 9 på Arduino för att sända och ta emot data [4].

2.3 Halleffektsensor

Halleffektsensor är en magnetisk sensor som kan användas för att bestämma styrkan och riktningen av ett magnetfält [5]. Halleffektsensor består av en tunn rektangulär p-typ halvledare som en kontinuerlig ström tillsätts igenom [5]. I ett magnetfält utsätts halvledaren av ett tryck som reflekterar laddningsbärare, elektroner, till båda sidorna av halvledaren [5]. Denna förflyttning skapar en potentialskillnad mellan båda sidorna av halvledaren [5]. Rörelsen av elektroner i halvledaren är beroende av magnetfältets styrka och riktning [5]. Denna effekt av att ett magnetfält skapar en mätbar potentialskillnad kallas för halleffekt [5].



Figur 2.3: Bild på halleffektsensor SS490 som används i examensarbetet.

Figur 2.3 visar halleffektsensorn som används i examensarbetet och som kan ses i figuren har sensorn 3 ben. Det vänstra benet kopplas till matningsspänning, mittenbenet kopplas till jord och på det högra benet ges en utspänning som är proportionellt mot magnetfältstyrkan. Anledningen till att denna sensor har valts är för att utspänning ligger mellan 0 V till 5 V och denna spänning överstiger inte Arduinons gräns på 5 V. Dessutom passar det magnetiska mätintervallet av -600 Gauss till 600 Gauss projektet [6].

2.4 Matlab

Matlab är ett kalkyleringsprogram som är bra på att bearbeta, analysera och visualisera värden. Matlab är en programmerings plattform som drivs av ett matrisbaserat

språk som kan användas från enkla kommandon till avancerade applikationer [7]. Med hjälp av Matlab kan man manipulera matriser, plotta funktioner och data. Matlab har flertal inbyggda kommandon och matematiska funktioner som gör det enkelt för användaren att utföra matematiska ekvationer, skapa plottar och genomföra numeriska metoder [7].

2.5 Android Studio

Android Studio är det officiella IDE:t (intergated development evironment) för utvecklandet av andoid applikationer [8][9]. Det är baserat på IntelliJ IDEA, som är ett java IDE för mjukvara, och använder flertal funktioner för utveckling från IDE:t. Det är skapat för det specifika syftet att utveckla, testa och producera applikationer för Android system. Det är ett av flera verktyg för skapande av Android applikationer på marknaden. Några exempel på hur de skiljer sig från konkurrenterna är deras UI (User Interface) utvecklingsverktyg, emulator, flertal inbyggda verktyg för ”building” och strukturen av projekt i programmet. Programmeringsspråket är Java och användbara java funktioner förekommer även i programmet [8][9].

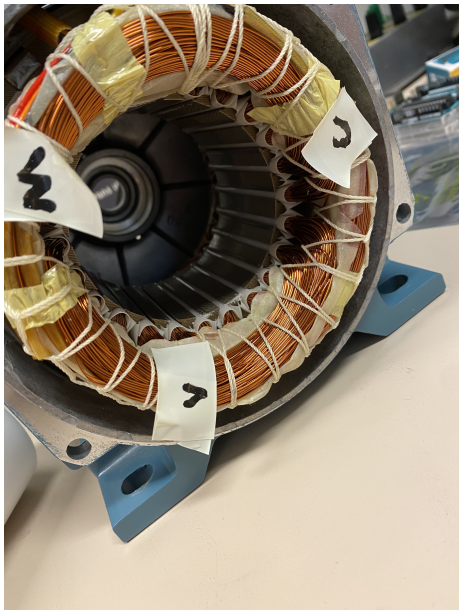
2.6 Asynkronmotor

Elmotorer kommer i flera olika former och versioner. Det är en maskin som omvandlar elektrisk energi till rörelseenergi. Majoriteten av elmotorer är av den roterande typen. Det betyder att de består av en roterande rotor och en stillastående stator men uppbyggnaden av dessa ser olika ut beroende på maskin. I detta examensarbete används en asynkronmotor för det är den vanligaste förekommande elmotorn [10] .

Asynkronmotorn tillhör gruppen AC-motorer som är baserade på ett roterande magnetfält i luftgapet mellan statorn och rotorn. Detta är en av de mest använda maskinerna bland elmotorer. Det som kännetecknar en asynkronmotor är den enkla och robusta designen, driftsäkerhet vid höga nivåer av arbete och att den är kostnadseffektiv. Motorn består av två huvuddelar som är statorn och rotorn [11]. Figur 2.3 visar hur asynkronmaskinens rotor kan se ut och Figur 2.4 visar hur statorn kan se ut.

Figur 2.4: Bilden visar en rotor på en rotoraxel

Som visas i Figur 2.4 består rotorn av en axel med lager, en laminerad kärna och lindningar. Lindningarna är oftast gjorda av aluminium men de kan också vara av koppar. Ett varv av lindningen motsvarar oftast en stång och stängerna är kortslutna i båda ändarna av kortslutningsringar. Detta gör att lindningen påminner om en bur och denna typ kallas också för burlindning. Rotorn läggs sedan in i statorn och hålls upp av axeln [11].



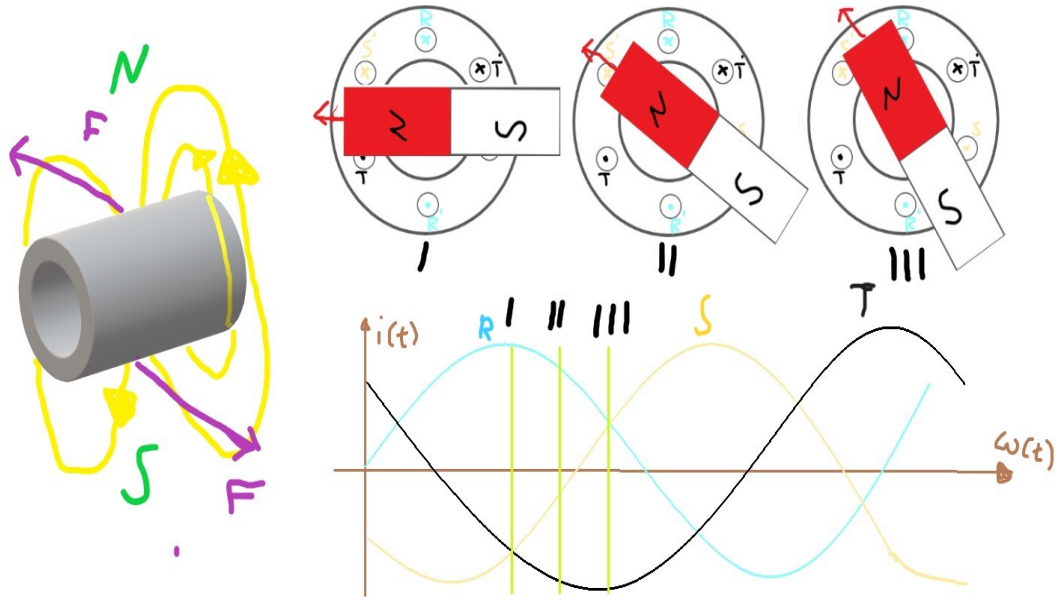
Figur 2.5: Bilden visar en bild på statorn och dess lindningar

Statorn utgörs av ett paket med laminerade plåtar och på insidan av paketet finns ett antal spår. I dessa spår läggs lindningarna, en spole per fas, och de består av isolerad koppartråd och beroende på utseendet av lindningen respektive kopplingen

ändras motorns poltal och därmed varvtal. Statorns huvudsakliga uppgift är att generera magnetfältet som roterar rotorn [10]. Statorn består även av motorhuset som är själva skalet till motorn och det tillverkas av pressgjuten aluminium för lågeffekt och grått gjutjärn och stål för högeffekt. Den innehåller även lagersköldar och fläktar [13].

Statorn är lindad med ett trefaslindningssystem eftersom det är anslutet till en trefasväxelspänning. Varje lindning är kopplad till var sin fasspänning. Den anslutna växelspänningen driver sinusformade strömmar med samma amplitud in i var och en av de tre lindningarna. Strömmarna är tidsförskjutna från varandra med 120 grader och detta medför att statorn bygger upp ett roterande magnetfält [13]. Anledningen till att detta sker bevisades av Galileo Ferraris efter 1885 [11]. Han skapade ett roterande flöde genom att koppla två spolar ortogonalt med omväxlande strömmar som är fasförskjutna med 120 grader.

I en trefasväxelspänningsmaskin placeras tre lindningar 120 grader förskjutna från varandra. Om en trefasspänning kopplas till lindningarna kommer en sinusformad ström flöda genom varje lindning. En strömbärande tråd genererar ett magnetfält runt omkring sig. Eftersom alla tre trådar skapar magnetfält runt omkring sig skapas ett gemensamt magnetfält av alla tre trådar tillsammans i ett specifikt tillfälle. Detta kallas scenario 1 som visas i Figur 2.6. Eftersom strömmen är omväxlande kommer det gemensamma magnetfältet ha olika riktningar beroende på vilket värde strömmarna har på sinuskurvan, och därmed ändras scenariot för magnetfältet. Dessa olika riktningar resulterar i ett roterande flöde och detta kan visas genom att visa alla tre strömmar samtidigt. Då visas alla olika variationer av värden på strömmar som ger olika positioner på magnetflödet. Under kontinuerlig tid kommer alla dessa positioner resultera i att magnetfältet roterar [11].



Figur 2.6: Bilden visar tre lindningar som är 120 grader förskjutna och det magnetiska flödet som bildas.

Det roterande magnetfältet inducerar en elektrisk spänning i rotorlindningen. Denna spänning driver en ström genom lindningen eftersom den är kortsluten. Tillsammans med det roterande fältet skapar strömmarna krafter som verkar på lindningarna och dessa genererar ett vridmoment. Vridmomentet accelererar rotorn i riktningen med det roterande fältet. Frekvensen på spänningen som induceras i rotorn minskar när hastigheten ökar eftersom skillnaden mellan det roterande magnetfältets hastighet och hastigheten på rotorn blir mindre. Om rotorn roterar med samma hastighet som det roterande fältet sägs den rotera med synkronhastighet och då induceras ingen spänning som medger att inget vridmoment skapas. Storleken på skillnaden mellan hastigheterna ökar och minskar beroende på belastningen som läggs på motorn, men är aldrig noll eftersom det alltid finns en friktion närvarande även utan belastning [13].

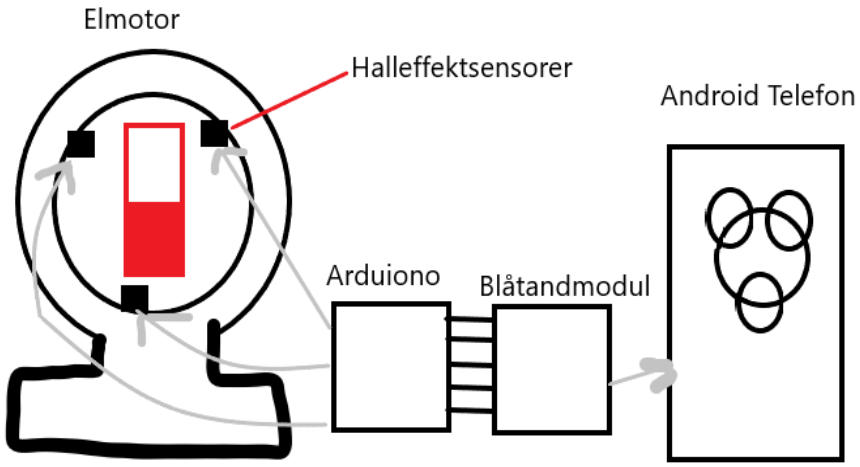
3

Metod

Detta kapitel kommer gå igenom metoder som används för att nå resultatet.

3.1 Hårdvara för att visualisera flöde

Material för detta projekt är: Arduino för att ta upp värdena från sensorerna, halleffektsensorer som mäter flödet, blåtandsmodul HC-05 för att överföra mätvärden från Arduino till applikationen, Breadboard för att koppla sensorerna till Arduinon, ett batteri eller spänningsaggregat för Arduinon samt Android studio för att programmera applikationen. En roterande magnet användes även för att generera det roterande magnetfältet i statorn manuellt. Se Figur 3.1 för att se hur delarna kopplas ihop.



Figur 3.1: Bilden visar materialen i kopplingen för systemet

3.2 Förstudie

Arbetet inleddes med en planeringsrapport där hur och när olika stadier i arbetet skulle utföras planerades. Denna planering resulterade i förstudier där de utvalda komponenterna och verktygen i projektet studerades för att öka kunskapen om hur de fungerar och hur de skulle samverka. Asynkronmotorn behövde repeteras från tidigare kurser. Datablad för halleffektsensorer studerades för att veta hur värden från sensorn skulle omvandlas till riktiga magnetfältsvärden. Då användande av Java för att producera appen var planerat studerades språket men på grund av tidigare erfarenheter av programmering behövdes inte för mycket studerande.

3.3 Mätning av flöde

Arbetsmetoden för att utföra detta arbete bestod av en hel del tester och utvärderande av resultat. Då halleffektsensorerna är de viktigaste i detta arbete, inleddes arbetet med det. Sensorerna kopplades till en Arduino som mätte värdena. Magnetfältet som sensorerna mätte varierades genom att en stavmagnet placerades på olika avstånd från sensorerna. Halleffektsensorn ger ut en analog spänning mellan 0 V till 5 V beroende på magnetfältets styrka och riktning. Denna spänning omvandlas i Arduinon till ett tal mellan 0 och 1024. Detta mappades om och studerades.

Blåtandsmodulen lades till då målet är att dessa värden ska skickas till en applikation. Valet låg mellan Blåtand och WiFi och eftersom erfarenhet om Blåtand fanns så valdes den. Först testades att skicka värden från en blåtandsmodul till en annan. Detta gjordes eftersom grafer skulle ritas i Matlab. Matlab användes som testinstrument då erfarenhet finns av programmet samt att det är lätt att plotta grafer i. Värdena som skickades till den andra blåtandsmodulen kopplades till en dator och skickades in i Matlab där grafer ritades. När detta uppnåddes utfördes tester där värden skickas via Blåtand till telefonen direkt. När detta fungerade påbörjades utvecklingen av applikationen.

3.4 Utveckling av applikation

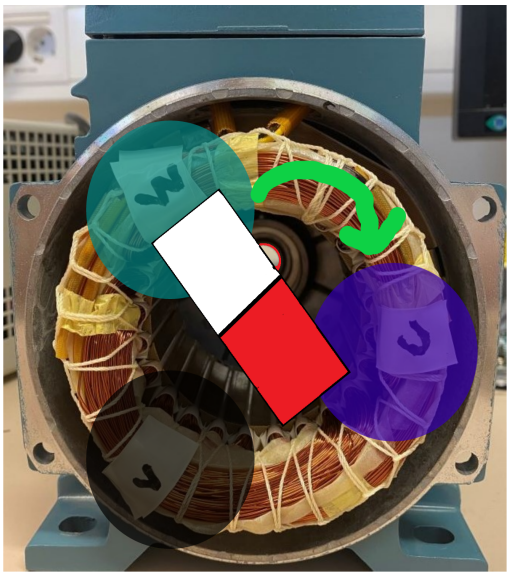
Efter att en stabil överföring av värden hade fåtts påbörjades utvecklingen av applikationen. Val av verktyg var Android studio och därmed språket java. Då detta var första gången båda medlemmarna i gruppen använde språket krävdes en del studerande av språket samt Androind Stuido verktyget. Detta gjordes med hjälp av Youtube men även forum på internet användes där uppladdade koder från olika personer studerades. Därefter studerades hur mottagandet av värden via Blåtand skulle utföras. När detta lyckades programmerades uppdelningen av dessa värden. Då värdena från Arduinon uppdateras konstant skedde detta i en typ av while-loop funktion för att få med alla uppdateringar. Till slut utvecklades UI, som sparades till sist då det var det enklaste.

3.5 Tester för slutresultat

För att kunna mäta flödet plockades en asynkronmaskin isär så att flödesmätaren får plats. Flödesmätaren består av hall-effekt sensorer som skickar värdena till en Arduino. Därefter ska en trefasväxelspänning implementeras som kan varieras i amplitud och frekvens. Flödet som skapas i motorn ska mätas och skickas via Blåtand från Arduinon till applikationen. På applikationen ska värdena visas i Gauss. På grund av tidsbrist kunde inte ett slut test i en aktiverad motor utföras då hål var tvungna att borras i rotorn för att stoppa sensorerna i hålet. I stället används en cylinder med en magnet i mitten. Sensorerna läggs på kanten där trefaslindningarna går. Därefter roteras cylindern som kommer visa hur magnetfältet snurrar i applikationen.

En visuell version av flödesvärdena ska visas i form av cirkulära figurer som ökar eller minskar i diameter baserat på värdet från varje lindning. Då motorn består av tre lindningar kommer 3 cirklar visas och när motorn snurrar kommer cirklarna röra sig baserat på vart magnetfältet är som högst i motorn.

Gruppens tankesätt vid val av cirkulära figurer för att demonstrera flödet var att man tänkte att en stavmagnet placeras i mitten av statorn. Detta ska representera det magnetiska flödet som bildas i en aktiv asynkronmotor. När motorn är igång kommer magnetfältet rotera och därmed roterar stavmagneten, se Figur 3.2. Där den positiva sidan (vita) av stavmagneten är kommer det vara ett ökat positivt flöde och där den negativa sidan (röda) är kommer det vara ett ökat negativt flöde. Cirklarna representerar mätpunkter inuti rotorn som är placerade vid lindningarna. Ett ökat flöde representeras av en expanderad cirkel och ett minskat flöde (ökat negativt) representeras av ett minskande cirkel. Detta tycker gruppen presenterar rörelsen av flödet väldigt tydligt och enkelt.



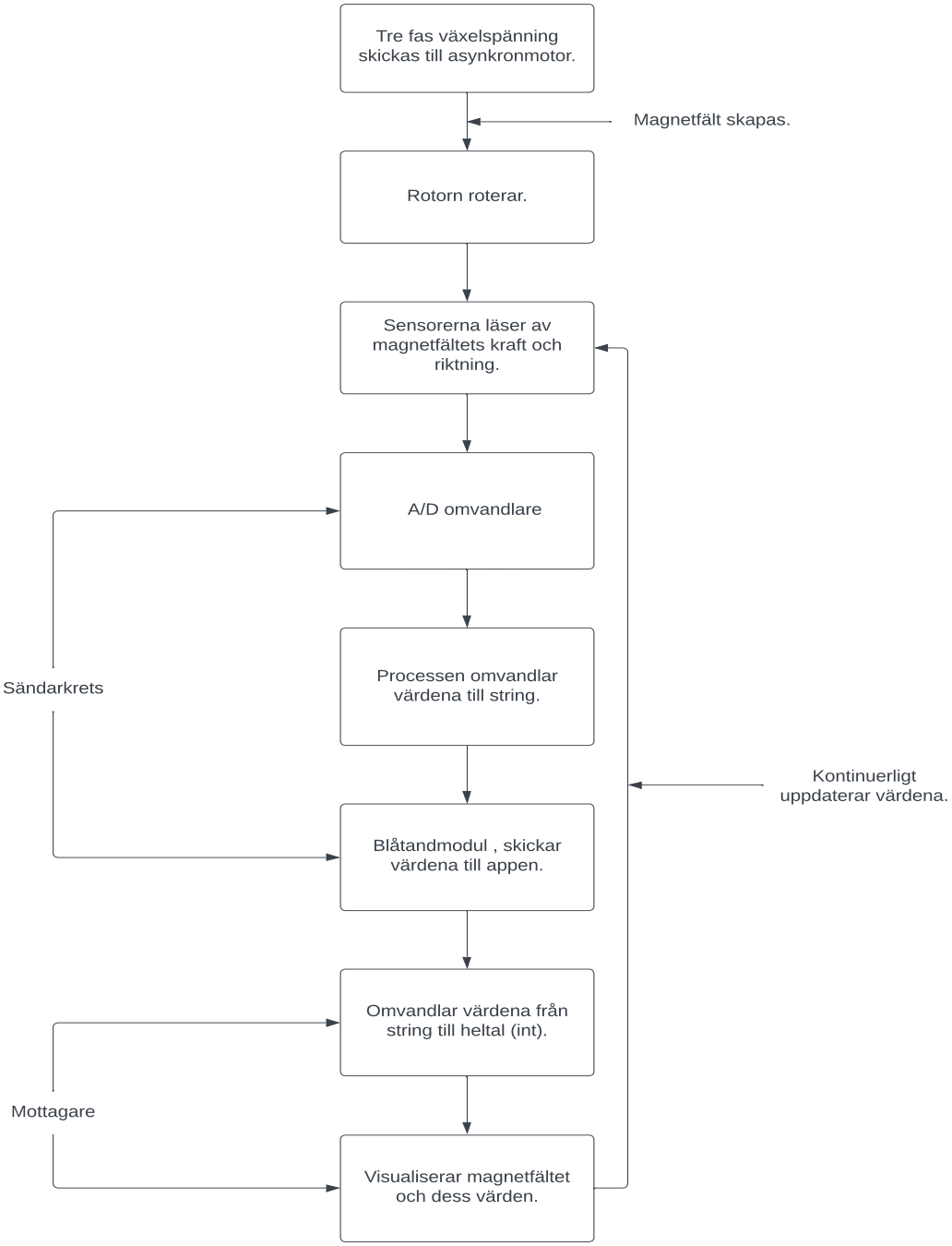
Figur 3.2: Bilden visualiserar gruppens tankesätt med en stavmagnet i en stator

Dessa visuella versioner ska programmeras i Android studio för att skapa en Android applikation. En knapp för att sätta igång mätningen måste tryckas för att starta mätningen som visar värdena.

4

Systemkonstruktion

För att kunna uppfylla målet av examensarbetet måste flera steg tas, där varje komponent utför sin uppgift och arbeta i enighet med andra komponenter. Figur 4.1 visar ett övergripande flödesschema över hur flödet i asynkronmaskinen mäts, processas och skickas till appen för visualisering. I följande kapitel kommer varje del beskrivas mer i detalj.



Figur 4.1: Flödesschema för hela systemet.

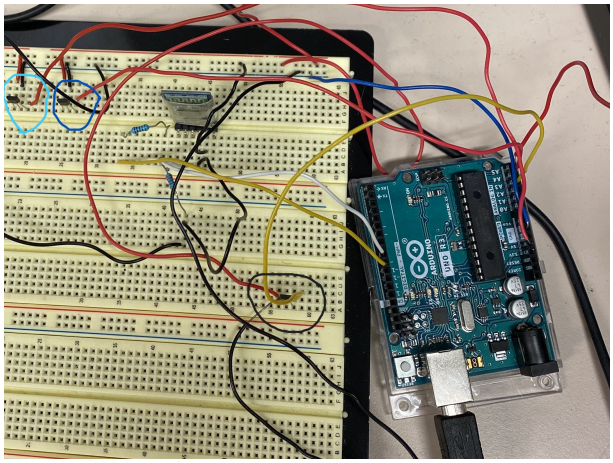
4.1 Asynkronmotor

För att kunna generera ett magnetfält kopplas en trefasväxelspänning till asynkronmotor vilket leder till att lindningarna i statorn skapar ett magnetiskt flöde. Detta indikerar att ett magnetfält har skapats och att systemet är redo att skicka över

värdena till appen. Men på grund av tidsbristen omvandlades asynkronmotor till en PMSM vilket står för permanentmagnetiserad synkronmotor. Flödet i luftgapet som skapas av magneterna i PMSM rotorn har samma utseende som flödet som skapas av trefaslindningarna. Därmed blir visualiseringen av det roterande flödet densamma om rotorn för PMSM roteras i asynkronmaskinens luftgap. För att kunna läsa av magnetfältets storlek sätts sensorerna vid varje faslindning på asynkronmaskinens stator.

4.2 Sändarkrets

Figur 4.2 visar en bild av sändarkretsens komponenter. De tre cirklarna visar halleffektsensorerna som skickar värdena till Arduino. Dessa ska sitta vid lindningarna på asynkronmotorn men för att enklare visualisera kopplingen av sändarkretsen är de kopplade i kopplingsdäcket. Längst upp i bilden är blåtandsmodulen som är kopplad till Arduino.



Figur 4.2: Visar sändarkretsen med de tre halleffektsensorerna inringade, blåtandsmodulen och arduino.

Figur 4.2 illustrerar hur sensorerna är kopplade till Arduino. Halleffektsensorerna matas med en 5 V spänning och är kopplade till jord. Dessutom skickar sensorn magnetfältsvärdena till Arduinons analoga ingångar. Därefter omvandlas värdena i Arduino till string och skickas ut till blåtandsmodulen med hjälp de digitala pinnarna 8 och 9 som motsvarar mottagare och sändare. Till sist skickar blåtandsmodulen ut värdena till mobiltelefonens Blåtand.

I Figur 4.3 visas hur magnetfältsvärdena omvandlas från en analog spänning till ett tal. Detta görs med hjälp av 'analogRead' som läser av spänningen av sensorerna. Som man kan se i Figur 4.1 omvandlas värdena från analog till digital. I Figur 4.3 används kommando 'map' som ändrar intervallet på mätvärdet.

```
// Tar emot data från Hall effekt sensor
Sensor1 = analogRead(A0);
Sensor1 = map(Sensor1, 0, 1024, -600, 600);
Sensor2 = analogRead(A1);
Sensor2 = map(Sensor2, 0, 1024, -600, 600);
Sensor3 = analogRead(A2);
Sensor3 = map(Sensor3, 0, 1024, -600, 600);
```

Figur 4.3: Magnetfält värdena läses av genom analog input och omvandlas till mätvärden i Gauss.

Anledningen till att intervallen av magnetfält värdena ändras är för att arduino ändrar från analog till digital. Spänningen från halleffektsensorena har intervallet 0 V till 5 V, vilket motsvarar -600 Gauss till 600 Gauss. Detta används i appen för att visualisera magnetfältstyrkan.

Figur 4.4 visar hur sensorvärdena omvandlas till string och samlas i variabeln data. Detta skickas sedan ut genom 'altserial.print(data)' som används för att skicka variabeln data med blåtandsmodulen. 'Serial.print(data)' används för att dubbelkolla så att blåtandsmodulen skickar rätt data till applikationen. Stringen innehåller '#' och ',' för att separera sensorvärdena och för att veta var stringen börjar.

```
//Skickar alla värden i en String via bluetooth till mobilens bluetooth
String data = "#," + String(Sensor1) + "," + String(Sensor2) + "," + String(Sensor3) + ",";
altSerial.print(data);
Serial.println(data);
```

Figur 4.4: String skickas från sändare till mottagare

4.3 Mottagare

Appen kommer att använda mobilens Blåtand för att ta emot magnetfältsvärdena från Arduinon. Appen programmerades i Android studio som använder programmeringsspråket java. För att kunna kommunicera med blåtandsmodulen i sändarkretsen behöver 'Socket' av blåtandsmodulen kopplas. I figur 4.5 kan man se för att säkerställa att mobilen kopplas till blåtandsmodulen så utförs ett do kommando som genomför btSocket.connect()"upp till 3 gånger eller ända till Blåtand har kopplats.

```
int counter = 0;
do {
    try {
        btSocket = hc05.createRfcommSocketToServiceRecord(mUUID);
        System.out.println(btSocket);
        btSocket.connect();
        System.out.println(btSocket.isConnected());
    } catch (IOException e) {
        e.printStackTrace();
    }
    counter++;
} while (!btSocket.isConnected() && counter < 3);

try {
    inputStream = btSocket.getInputStream();
    inputStream.skip(inputStream.available());
} catch (IOException e) {
    throw new RuntimeException(e);
}
```

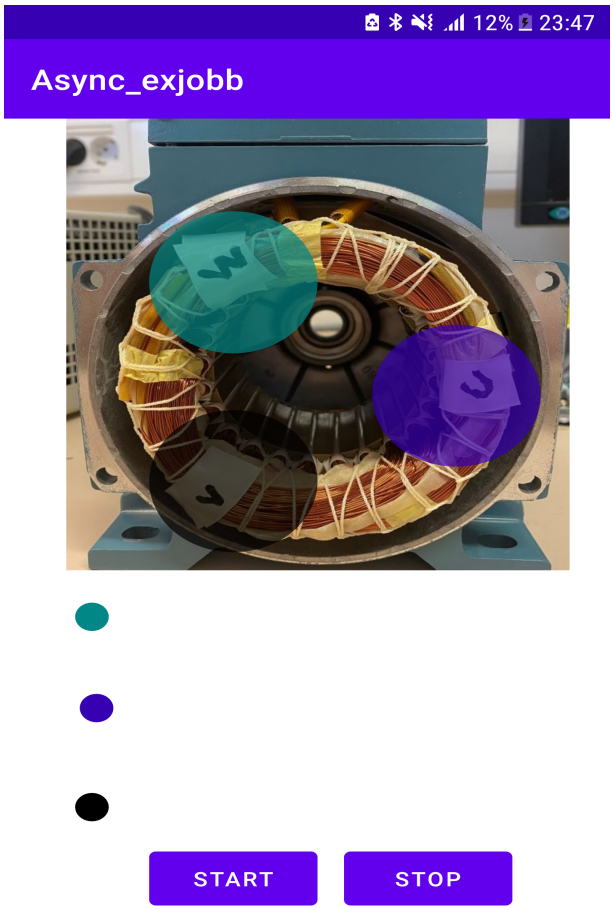
Figur 4.5: Visar koden för kopplingen mellan blåtandsmodulerna

Figur 4.6 visar hur stringen separeras och omvandlas till heltal (int). Detta görs genom att detektera när ',' och '#' dyker upp. Anledningen till att värden omvandlas tillbaka till heltal (int) är för att kunna ändra på figurerna och texterna som används för att visualisera magnetfältet.

```
            b = (char) inputStream.read();
            //System.out.println(b);
        } catch (IOException ex) {
            throw new RuntimeException(ex);
        }
        if (b == ',') {
            if (tempstring.length() > 0) {
                try {
                    value[i] = Integer.parseInt(tempstring);
                    temp[i] = (value[i] + 600)/2;

                    params1.width = temp[0];
                    params1.height = temp[0];

                    params2.width = temp[1];
                    params2.height = temp[1];

                    params3.width = temp[2];
                    params3.height = temp[2];

                    publishProgress(customCounter);
                } catch (NumberFormatException nfe) {
                    //System.out.println("not a value");
                }
                i++;
                //System.out.println(i);
                tempstring = "";
            }
        } else {
            tempstring += b;
        }
        if (b == '#') {
            i = 0;
            tempstring = "";
        }
```

Figur 4.6: Visar hur koden omvandlar data från string till int

För att data från blåtandsmodulen ska kunna användas av applikationen måste den omvandlas till heltal. Detta kommer sedan användas för att visualisera magnetfältet där cirklarna ökar och minskar i storlek beroende på hur stort magnetfältet är. För att ändra cirklarnas storlek ändras deras bredd (width) och höjd (height) baserat på value[]. För att figurerna inte ska bli för stora så ändras deras intervall från -600 till 600 till ett intervall från 0 till 200 i temp[]. Applikationen består av färgkodade cirklar som ändrar sin bredd och höjd baserat på temp[]. Dessa ligger på en bakgrundsbild av statorn i labbet som ska visa placeringen av lindningarna i motorn. Under bilden kommer värdet av flödet i varje lindning presenteras i Gauss bredvid miniatyrcirklarna. Detta värde tas direkt från value[] då det är värdet som mäts upp direkt från sensorerna, se Figur 4.7.



Figur 4.7: Bilden visar den slutliga versionen av applikationen

Figur 4.8 kan man se hur figurerna och textens parametrar modifieras. Detta görs med hjälp av 'setLayoutParams' som ändrar på figurerna och 'setText' som ändrar på texten. Detta uppdateras kontinuerligt inom funktionen 'onProgressUpdate'.

```
@SuppressLint("SetTextI18n")
@Override
protected void onProgressUpdate(Integer... values) {
    super.onProgressUpdate(values);
    C1.setLayoutParams(params1);
    C2.setLayoutParams(params2);
    C3.setLayoutParams(params3);
    T1.setText("Gauss:"+value[0]);
    T2.setText("Gauss:"+value[1]);
    T3.setText("Gauss:"+value[2]);
}
```

Figur 4.8: Visar hur figurerna och texterna uppdateras

För att kontinuerligt se hur magnetfältet ändras över tiden behöver magnetfälts-

värdena läsas av regelbundet. Detta görs med hjälp av asynctask. Där koden som omvandlar string till heltal genomförs i 'doinbackground', se appendix. Detta görs vid sidan av huvudtråden och gör det möjligt för skärmen att uppdateras. För att uppdatera skärmen används 'onProgressUpdate' som uppdaterar skärmen kontinuerlig. Detta uppdaterar kontinuerligt de expanderande cirklarna och Gauss värdena som skrivs ut under dem.

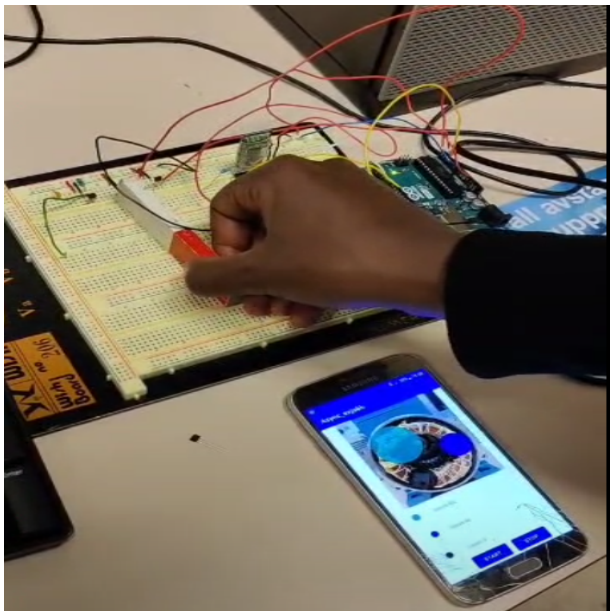
5

Resultat och Diskussion

En fullständig applikation som tar emot mätvärden från tre halleffektsensorer och visualiserar mätvärdena i realtid skapades. I Figur 4.7 visas en skärmbild på applikationen där den över delen visar motorn och i den nedre delen visas mätvärdena från de tre sensorerna i Gauss. Appen visar cirkulära figurer vid varje lindning i motorn. Dessa expanderar och krymper beroende på värdet de får från sensorn. Detta illustrerar ökandet och minskandet av flödet i varje lindning. Då flödet roterar i maskinen så kommer varje lindning mäta upp olika storlekar av flöde beroende på vart flödet är. I applikationen kan man se en bild på statorn och cirklarna runt den. Detta ska ge en mer illustrativ bild av vad som faktiskt händer.

Som man kan se i Figur 4.7 visas cirklarna vid varje lindning samt bakgrundsbilden på en motor. Bakgrundsbilden är en bild på motorn vi har i labbet, i bilden kan man se statorn, lindningarna och motorhuset. Cirklarna är färgkodade. Detta är för att lättare kunna hålla koll på de tre olika lindningarna. Under finns en miniatyrversion av varje cirkel samt ett värde i Gauss bredvid. Detta visar värdet på flödet som lindningen har i Gauss. Längst ner i applikationen finns det en startknapp och en stoppknapp. Dessa knappar sätter igång eller stänger av realtidsmätningen.

Cirklarna kommer ändra storlek beroende på det magnetiska flödet som kommer över sensorerna. Varje sensor illustreras av en cirkel. Detta testades med en tvåpolig magnet som läggs mot sensorerna. Detta visas i figur 5.1.

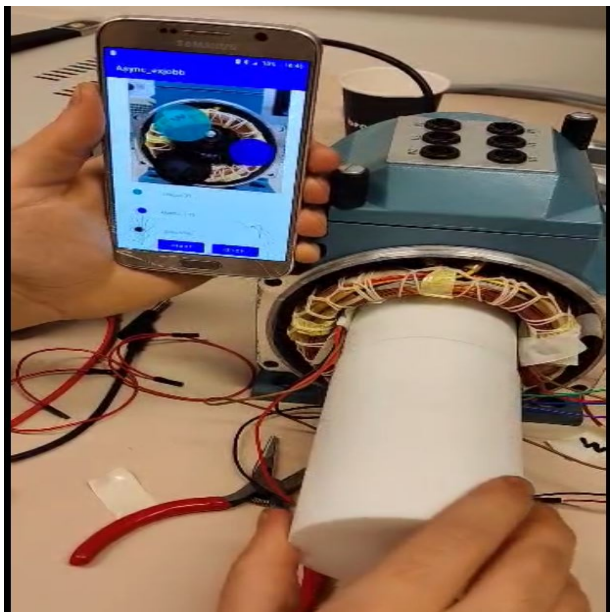


Figur 5.1: Bilden visar när cirkeln expanderar när en magnet hålls mot sensorn

Figur 5.1 visar hur den ljusblå cirkeln som är kopplad till den sensor som magneten läggs framför expanderar när det magnetiska flödet ökar med hjälp av magneten. Detta visar att applikationen fungerar som planerat. Detta fenomen kommer förekomma inuti en asynkronmotor då flödet kommer öka eller minska olika vid varje lindning under rotationen. När cirklarna expanderar eller minskar är inte animationen hackig eller seg. Den är relativt jämn och visar en naturlig expansion i realtid. Hursomhelst kommer ändringen av cirklarna i applikationen med en liten fördröjning på några hundra millisekunder. Detta beror på en del faktorer. Bluetooth överföringen är en av anledningarna. I överföringen skickas en sträng av text med alla värdena konstant. Dessa ska tas emot i applikationen och ska sedan delas upp och skickas till sin array. Då denna överföring och uppdelning sker konstant i iterationer kommer de ta extra tid att processa. Därför förekommer fördröjningar i programmet. Detta kan utvecklas med hjälp av bättre utrustning. Men ett lättare sätt att förebygga fördröjningen kan vara i koden. Vi hann inte testa alla sätt att uppdatera variabler i realtid baserat på inputdata från en bluetoothstream. Det bästa vi hittade var 'asyncTask' men möjligtvis finns det en bättre metod.

Sluttestet gjordes som sagt med en cylinderformad permanentmagnet som passade perfekt i statorn. I varje lindning sattes en halleffektsensor som kopplades till Arduinon och skickade värden till applikationen. Permanentmagneten roterades för hand och skapade ett roterande magnetfält i statorn som fångades upp av våra sensorer. Resultatet visades lyckat i applikationen. Cirklarna expanderade och krympte på rätt sätt, vid rätt tillfälle och i rätt ordning. Först expanderade den första cirkeln sedan började föregående cirkel krympa, därefter börjar nästa cirkel expandera och nuvarande krympa. Detta repeterades under rotation som visar hur det positiva och negativa flödet roterar. Se Figur 5.2. Det var inte för mycket hackighet och seghet i animationen förutom fördröjningen som nämndes innan. Applikationen var responsiv under testningen och tester med att rotera snabbare och byta riktning gjordes,

som också gav positiva resultat.



Figur 5.2: Bilden visar ett skärmklipp ifrån när sluttestet utfördes och hur det såg ut.

Syftet med detta projekt var att skapa en demostrator som ska visa rörelsen av flödet i en asynkronmotor i realtid. Detta uppnåddes på ett bra och illustrativt sätt. Hursomhelst var planen i början att plotta resultatet i en 3D-graf. Under arbetsgången var en 3D-plot huvudfokuset men båda gruppmedlemmarna hade svårt att visualisera hur detta skulle plottas ut med värdena från sensorn. När tiden kom att implementera detta resulterade de i en ändring av planen. Ett alternativt sätt att representera förändring av flödet i motorn diskuterades och resultatet blev en bild av en motor som visar lindningarna. Varje lindning markeras och förändring av flödet ska visas på varje lindning i form av en expanderande cirkel. Gruppen tycker att detta sätt visar flödesförändringen mycket tydlig och är lätt att förstå. Å andra sidan får man inte med de mer komplexa aspekterna av den sinusformade rörelsen av flödet i en lindning. Detta kan vara ett förbättringsområde för projektet.

En annan liten potentiell förbättring på arbetet är förminskning av fördröjningen på värdeförändringen. Denna fördröjning är inte märkvärdig och påverkar inte processen för mycket men det kan förbättras. Fördröjningen ligger på några hundra millisekunder. Detta kan förbättras med bättre material som till exempel snabbare och starkare Bluetooth moduler. Det kan dock även förbättras med en annan metod att ta emot en string genom Bluetooth och separera den iterativt. Tiden för att undersöka flera metoder än asyncTask fanns dock inte, men detta går att undersöka djupare.

Den svåraste och mest tidskrävande delen av projektet var att lyckas med att skicka över en string från Arduino till applikationen, separera den till de enskilda värdena och göra om de till heltal, allt i realtid. Detta tog mycket längre tid än planerat då

ett flertal tester av metoder misslyckades eller kom väldigt nära men aldrig i mål. Detta gjorde även att tid inte fanns för att lägga till extra funktioner i applikationen. Tid fanns inte heller till att testa appen i en riktig motor då förberedelserna inte hann göras. Detta berodde främst på att detta var gruppens första arbete med java och Android studio. Java är väldigt användbart men även ett väldigt stort språk med massa olika verktyg och metoder inom språket. Därmed finns det flera olika metoder att nå samma resultat och på grund av tidsbrist går det inte att utforska allt. Hursomhelst fick gruppen ut väldigt mycket av detta och lärde sig en hel del, speciellt om java och Android studio.

Andra förbättringsaspekter för detta projekt är en utvecklad applikation i flera aspekter. Ett exempel är att lägga till en sökfunktion för att koppla till flera Bluetooth moduler. Programmet är just nu hårdkodat till att koppla till en specifik Bluetooth modul och bara den. Detta är bra för om man har flera liknande demonstratorer och man vill kunna koppla sig mellan allihop. Alltså är detta en bra förbättring för en mer utvecklad produkt som har flera olika versioner av sig. Ett annat exempel på förbättring i applikationen är att lägga till en funktion som kan styra frekvensen av motorn som mätaren ligger på. För att uppnå detta måste spänningsmatningen till motorn kunna styras genom applikationen och därmed måste parametrarna kunna ställas in därifrån. Detta kommer fördjupa applikationen mycket mer och göra den mycket mer illustrativ av hur flödet fungerar i en motor. Eftersom hastigheten av rotationen påverkas av effekten kan flödets påverkan av effekten studeras med denna implementation. Alltså ger det en mer komplett applikation i ett undervisningsperspektiv.

6
Slutsats

Gruppen anser att de lyckats uppnå en applikation som kan mäta upp en förändring av ett roterande magnetiskt flöde i en asynkronmotor med hjälp utav sensorer och demonstrera förändringarna i realtid. Att visualisera flödet i Gauss är även uppnått. Applikationen uppnådde uppsatta mål och syftet väldigt bra.

Sammanfattningsvis har detta vart ett väldigt aktuellt och lärorikt projekt. Att demonstrera hur flödet rör sig i en asynkronmotor är inte ett väldigt populärt projekt och inte många har utvecklat något inom området. Detta gör att projekt har hjälpt till med utvecklingen inom området. Projektet kan utveckla den pedagogiska aspekten i detta ämne då applikationen på ett tydligt sätt visa det roterande flödet i maskinen. Gruppen är nöjda med arbetet och uppskattar allt som lärts under tiden.

Litteraturförteckning

[1] Gough, E. (2022) We might know why Mars lost its magnetic field, Universe Today. Available at: https://www.universetoday.com/154461/we-might-know-why-mars-lost-its-magnetic-field/ (Hämtad: 12 Juni 2023).

[2] M, Schultz. (2022) What is Arduino? – simply explained, All3DP. All3DP. Tillgänglig: https://all3dp.com/2/what-is-arduino-simply-explained/ (Hämtad: November 18, 2022).

[3] A, Javaid. (2022) How to Interface Bluetooth Module (HC-05) with Arduino Uno, How to interface bluetooth module (HC-05) with Arduino Uno. Linuxhint. Tillgänglig: https://linuxhint.com/interface-bluetooth-module-arduino-uno/ (Hämtad: November 20, 2022).

[4] PJRC (inget datum) Altsoftserial Library, AltSoftSerial Library. Tillgänglig: https://www.pjrc.com/teensy/td_libs_AltSoftSerial.html (Hämtad: 04 Juni 2023).

[5] Storr, W. (2022) Hall effect sensor and how magnets make it works, Basic Electronics Tutorials. Available at: https://www.electronics-tutorials.ws/electromagnetism/hall-effect.html (Hämtad: 04 Juni 2023).

[6] Honeywell, "Linear Hall-effect Sensor ICs: SS490 Series" 055843 Issue 2 datasheet, Jun. 2018 (Hämtad: 22 Juni 2022).

[7] TutorialsPoint (ingen datum) Matlab - Overview, Online Courses and eBooks Library. Tillgänglig: https://www.tutorialspoint.com/matlab/matlab_overview.htm (Hämtad: 04 Juni 2023).

[8] N. Mathur, "What is Android Studio and how it differs from other IDEs | Packt Hub," Packt Hub, May 30, 2018. https://hub.packtpub.com/android-studio-how-does-it-differ-from-other-ides/ (Hämtad Jun. 04, 2023).

[9] TechTarget Contributor, "Android Studio," Mobile Computing, 2023. https://www.techtarget.com/searchmobilecomputing/definition/Android-Studio (Hämtad Jun. 04, 2023).

[10] "Jernkontorets energihandbok," Jernkontorets energihandbok, 2014. https://www.energihandbok.se/elmotorer-teknik-och-funktion (Hämtad Jun. 04, 2023).

[11] J. Company, "What is an INDUCTION MOTOR (THREE-PHASE ASYNCHRONOUS MOTOR) RMF Rotating Magnetic Field," YouTube. Jun. 10, 2019. Hämtad: Jun. 26, 2023. [YouTube Video]. Available: https://www.youtube.com/watch?v=EcbHEOIVGcgab_channel=JAESCompany

[12] “Asynkronmaskinen.pdf: EEK565 Elkraftsteknik,” Instructure.com, 2022. https://chalmers.instructure.com/courses/18318/files/2080704?module_item_id=264191 (accessed Jul. 09, 2023).
[13] SEW-EURODRIVE, “Asynkrona växelströmsmotorer, AC-motorer | SEW-EURODRIVE,” Sew-eurodrive.se, 2014. https://www.sew-eurodrive.se/produkter/motorer/vaexelstroemsmotorer/vaexelstroemsmotorer.html (Hämtad Jun. 04, 2023).

A

Appendix

A.1 Sändarkrets kod

```
#include <AltSoftSerial.h>
#include <string.h>
int sensor;
int sensor1;
int sensor2;

int val_sensor;
int val_sensor1;
int val_sensor2;
int Value;

AltSoftSerial altSerial;


void setup() {
 altSerial.begin(9600);
 Serial.begin(9600); // open the serial port at 9600 bps:



}

void loop() {
 String data;

  sensor = analogRead(A0);  // read the input pin
  sensor1 = analogRead(A1);
  sensor2 = analogRead(A2);

  val_sensor = map(sensor, 1023, 0, 640, -640);
  val_sensor1 = map(sensor1, 1023, 0, 640, -640);
  val_sensor2 = map(sensor2, 1023, 0, 640, -640);

  //data = "#," + String(val_sensor)+ "," + String(val_sensor1)+ ","  + String(val_
```

```
   data = "#" + String(val_sensor) + "," + String(val_sensor1)+ ","  + String(val_s


  altSerial.println(data);  // prints a label



  Serial.println(data);         // prints a tab

  delay(80);

}
```

A.2 Android Studio kod

```
package com.example.async_exjobb;

import android.annotation.SuppressLint;
import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.ServiceConnection;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;

import androidx.appcompat.app.AppCompatActivity;

import java.io.IOException;
import java.io.InputStream;
import java.util.UUID;

public class MainActivity extends AppCompatActivity implements View.OnClickListener

    private static final String TAG = MainActivity.class.getSimpleName();

    private Button buttonStart;
    private Button buttonStop;
```

```
ImageView C1;
ImageView C2;
ImageView C3;

TextView T1;
TextView T2;
TextView T3;

BluetoothAdapter btAdapter = BluetoothAdapter.getDefaultAdapter();
BluetoothDevice hc05 = btAdapter.getRemoteDevice("00:19:10:09:49:0B");
BluetoothSocket btSocket = null;
InputStream inputStream = null;
static final UUID mUUID = UUID.fromString("00001101-0000-1000-8000-00805F9B34FB
private MyAsyncTask myAsyncTask;

private ServiceConnection  serviceConnection;

Handler handler;

private boolean mStopLoop;



@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    buttonStart = (Button) findViewById(R.id.buttonstart);
    buttonStop = (Button) findViewById(R.id.buttonstop);


    C1 = findViewById(R.id.C1);
    C2 = findViewById(R.id.C2);
    C3 = findViewById(R.id.C3);

    T1 = findViewById(R.id.T1);
    T2 = findViewById(R.id.T2);
    T3 = findViewById(R.id.T3);



    //textViewthreadCount = (TextView) findViewById(R.id.textViewthreadCount);

    buttonStart.setOnClickListener(this);
    buttonStop.setOnClickListener(this);
```

```
        System.out.println(btAdapter.getBondedDevices());
        System.out.println(hc05.getName());

        int counter = 0;
        do {
            try {
                btSocket = hc05.createRfcommSocketToServiceRecord(mUUID);
                System.out.println(btSocket);
                btSocket.connect();
                System.out.println(btSocket.isConnected());
            } catch (IOException e) {
                e.printStackTrace();
            }
            counter++;
        } while (!btSocket.isConnected() && counter < 3);

        try {
            inputStream = btSocket.getInputStream();
            inputStream.skip(inputStream.available());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }

        handler=new Handler();

    }

    @Override
    public void onClick(View view) {
        switch (view.getId()) {
            case R.id.buttonstart:
                mStopLoop = true;
                myAsyncTask=new MyAsyncTask();
                myAsyncTask.execute();
                break;
            case R.id.buttonstop:
                mStopLoop = false;
                //myAsyncTask.cancel(true);
                break;


        }
```

```
}

private class MyAsyncTask extends AsyncTask<Integer, Integer, Integer> {

    private int customCounter;

    ViewGroup.LayoutParams params1 = C1.getLayoutParams();
    ViewGroup.LayoutParams params2 = C2.getLayoutParams();
    ViewGroup.LayoutParams params3 = C3.getLayoutParams();

    int i = 0;

    int[] value= {0,0,0};
    int[] temp= {0,0,0};

    String tempstring = "";

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
        customCounter = 0;


    }

    @Override
    protected Integer doInBackground(Integer... integers) {
        while (mStopLoop) {
            try {
                Thread.sleep(1);

                    char b = 0;
                    try {
                        b = (char) inputStream.read();
                        //System.out.println(b);
                    } catch (IOException ex) {
                        throw new RuntimeException(ex);
                    }

                    //ViewGroup vg = findViewById(R.id.MainLayout);
                    //vg.invalidate();


                    if (b == ',') {
                        if (tempstring.length() > 0) {
```

```
                        try {
                            value[i] = Integer.parseInt(tempstring);

                            temp[i] = (value[i] + 600)/2;

                            params1.width = temp[0];
                            params1.height = temp[0];

                            params2.width = temp[1];
                            params2.height = temp[1];

                            params3.width = temp[2];
                            params3.height = temp[2];

                            publishProgress(customCounter);
                            //setContentView(R.layout.activity_main);


                        } catch (NumberFormatException nfe) {
                            //System.out.println("not a value");
                        }

                        i++;
                        //System.out.println(i);
                        tempstring = "";
                    }

                } else {
                    tempstring += b;
                }
                //System.out.println(tempstring);

                if (b == '#') {
                    i = 0;
                    tempstring = "";
                }

                // "#,123,132,141,"


                System.out.println("Value 1:" + value[0]);
                //System.out.println("Value 2:" + value[1]);
                //System.out.println("Value 3:" + value[2]);

            } catch (InterruptedException e) {
                throw new RuntimeException(e);
```

```
                }
                /*if(isCancelled()){
                    break;
                }*/
            }
            return customCounter;
        }

        @SuppressLint("SetTextI18n")
        @Override
        protected void onProgressUpdate(Integer... values) {
            super.onProgressUpdate(values);
            C1.setLayoutParams(params1);
            C2.setLayoutParams(params2);
            C3.setLayoutParams(params3);
            T1.setText("Gauss:"+value[0]);
            T2.setText("Gauss:"+value[1]);
            T3.setText("Gauss:"+value[2]);
        }

        @SuppressLint("SetTextI18n")
        @Override
        protected void onPostExecute(Integer integer) {
            super.onPostExecute(integer);
            C1.setLayoutParams(params1);
            C2.setLayoutParams(params2);
            C3.setLayoutParams(params3);
            T1.setText("Gauss:"+value[0]);
            T2.setText("Gauss:"+value[1]);
            T3.setText("Gauss:"+value[2]);
            //count=integer;
        }
    }


}
```

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

CHALMERS