CHALMERS

Idrifttagning, utveckling och utvärdering av hybridanläggning på Hönö

— *Kandidatarbete under institutionen för Energi och Miljö*

29 maj 2015

Charlotta Aguirre Nilsson	chanil@student.chalmers.se
Mattias Alnervik	alnervik@student.chalmers.se
Maria Nisshagen	marnis@student.chalmers.se
Maria Peshkova	peshkova@student.chalmers.se
Charlotte Rådahl	radahlc@student.chalmers.se
Jonathan Winther	jonwin@student.chalmers.se

ENMX02-15-08 – Kandidatarbete
Institutionen för Energi och Miljö
CHALMERS TEKNISKA HÖGSKOLA
Göteborg, Sverige 2015

Förord

Vi skulle vilja tacka alla som har hjälpt oss. I första hand våra handledare, forskningsingenjör Magnus Ellsén och doktorand Joachim Härsjö som alltid funnits där med goda råd och som stöttat oss hela vägen. Vi vill även tacka projektkoordinator och examinator Jimmy Ehnberg för all hjälpande feedback med rapporten. Stort tack till forskningsingenjör Robert Karlsson vars ovärderliga kunskap har hjälpt oss genom projektet.

Övriga personer som hjälpt oss, som vi känner tacksamhet till är Niclas Johansson från Imtech som hjälpte till med inkoppling av Windstar 3000 samt bidrog med expertis. Vi vill även tacka Annie Grundevik för all hjälp med busskort för resor till och från Hönö.

Göteborg, maj 2015

Charlotta, Mattias, Maria, Maria, Charlotte och Jonathan.

Abstract

Chalmers University of Technology owns a hybrid power facility located on the island Hönö, in the archipelago of Gothenburg, Sweden. The aim of this project is to analyze and evaluate if the hybrid power facility is able to produce more energy than it consumes on an annual basis. The hybrid power facility was put in use two years ago when the solar panels, Naps Pallas, and the wind turbine, Windstar 3000, were purchased and installed. The solar panels consists of 15 solar cells and Windstar 3000 is a five-bladed wind turbine with a rated power of 3 kW. The solar panels have been generating electricity as expected, but Windstar 3000 was forced to be taken down a few days after installation due to not being properly mounted. Besides not being properly installed it lacked a functional breaking mechanism that protects the turbine from breaking at higher wind speeds. Nor did the inverter to the wind turbine have an optimal power curve, adapted to give optimal energy production. This caused the wind turbine to produce lower power than it was supposed to. To assure that the facility has a higher energy production than consumption, a system had to be designed and installed that collects the energy data from the facility.

Due to lack of time only a part of the breaking circuit was made, the frequency measuring circuit. Arduino Nano was used for this purpose. The wind turbine had stopped working when this circuit, in combination with a part of another breaking circuit, was installed. Even though the wind turbine was taken down and analyzed, no errors were found and neither the breaking system or the final power curve could be tested.

With help from older weather statistics and the data collection system, made with both Raspberry Pi 2 model B and Arduino Mega 2560, a conclusion could be made regarding the energy production of the non-functional Windstar 3000. This conclusion would prove that it would be possible for the wind turbine and the solar cells to produce enough energy to overcome the consumption of the hybrid power facility.

Sammanfattning

Syftet med projektet är att undersöka om Chalmers hybridanläggning på Hönö i Öckerö kommun kan producera ett nettoöverskott av energi sett till årsbasis. Hybridanläggningen består av 15 solcellspaneler av märket Naps Pallas samt vindkraftverket Windstar 3000. Anläggningen sattes i bruk för två år sedan. Solcellerna har under den tiden fungerat bra och producerat förväntad mängd energi. Det fembladiga vindkraftverket Windstar 3000 har märkeffekt på 3 kW och var vid projektets start nedmonterat. Då vindkraftverket saknade ett fungerade bromssystem behövde detta åtgärdas för att förhindra att verket havererar. Omriktaren till Windstar 3000 hade ingen anpassad belastningskurva för optimal produktion, vilket gjorde att verket inte kunde nå upp till märkeffekt. För att kunna undersöka och verifiera att Windstar 3000 tillsammans med solcellspanelerna producerar ett nettoöverskott på el behövde ett system för datainsamling konstrueras och installeras.

Tidsbrist gjorde att bara den del av bromskretsen som behandlade frekvensmätning hann byggas, till detta användes Arduino Nano. När denna i kombination med en lånad krets för bromsning skulle installeras tillsammans med den uträknade belastningskurvan hade det nyuppsatta vindkraftverket gått sönder. Trots att turbinen plockades ner hittades inte felet vilket gjorde att vare sig bromssystemet eller belastningskurvan kunde testas på plats.

Med hjälp av datainsamlingsprogrammet, som programmerades för Raspberry Pi 2 modell B och Arduino Mega 2560, samt äldre väderstatistik från bland annat SMHI kunde slutsatsen dras att om Windstar 3000 hade fungerat och kunnat producera effekt som planerat skulle det inte innebära några större svårigheter att, tillsammans med solcellerna, uppnå tillräcklig producerad energi.

Innehåll

1 Inledning

Global uppvärmning har fått världen att se över sina respektive utsläpp av växthusgaser. I Sverige är energiproduktionen baserad på fossila bränslen [1] och står för 18 procent [2] av det totala utsläppet av växthusgaser år 2013, enligt Naturvårdsverkets statistik. En mer hållbar energiproduktion kan bli en bidragande faktor till minskad växthuseffekt.

EU-kommissionen har arbetat fram riktlinjer och mål för hur medlemsländernas totala utsläpp av växthusgaser borde se ut år 2050. Dessa riktlinjer presenteras i projektet Energy Roadmap 2050 [3], framtaget av EU och G8, där målet är att minska utsläppen med 80 - 95 procent 2050 jämfört med år 1990. Kommissionen trycker på det framtida behov av effektiva och ekonomiskt fördelaktiga energiproducenter, där förnyelsebara källor kommer vara ledande [3]. I och med detta har Sveriges regering tagit fram en vision med målsättning att år 2020 ska svensk energiproduktion till 50 procent vara förnyelsebar - där vind- och vattenkraft är de största aktörerna [4].

Chalmers tekniska högskola satsar på hållbar utveckling [5]. Forskningsprogrammet Chalmers Energy Initiative fördjupar sig i fyra stora områden, där ett av de hänvisar till att utveckla vindkraft till en mer effektiv energikälla [6]. Målen baseras på regeringens målsättningar och visioner. Institutionen för Energi och miljö på Chalmers bedriver forskning inom elkraft med fokus på tre huvudområden; elkraftsystem, kraftelektronik och elmaskiner [7]. Många projekt på institutionen har fokus på förnyelsebar energi och ett av dessa projekt är hybridanläggningen på Hönö, bestående av solceller och vindkraftverk [8].

1.1 Bakgrund

Sedan tidigare finns en vindkraftsanläggning på Hönö där Chalmers bedriver forskning. År 2013 investerade institutionen i en utbyggnad av anläggningen där 15 solcellspaneler och vindkraftverket Windstar 3000 installerades. I samband med detta blev anläggningen hybrid. Syftet med anläggningen är att möjliggöra experiment, utveckling och forskning på sol- och vindkraft i verklig miljö.

På anläggningsområdet finns en väderstation belägen på en mast som bland annat mäter vindhastighet och vindriktning. Det finns även två bodar: den ena är en personalbod och den andra är en arbetsbod innehållande diverse utrustning som energimätare för samtliga energiproducenter och omriktare. Solcellspanelerna är installerade på arbetsbodens tak och är av märket Naps Pallas. I direkt anslutning till bodarna är vindkraftverken Windstar 3000 och AirDolphin belägna, dessa har kapacitet på 3 respektive 1 kW.

Statistik rörande tidigare års vind- och solstatistik har tagits fram. Den vindstatistik som har använts är hämtad från en tidigare rapport och är data från en väderstation på anläggningsområdet [9]. Statistiken över antalet soltimmar har samlats in från SMHI och refererar till soltimmar från väderstationen i Göteborg [10]. Statistiken över

anläggningens förbrukning under åren 2013 - 2014 är tidigare framtagen genom manuell avläsning av energimätarna i arbetsboden [11].

Innan projektets start har mätningar på Windstar 3000 genomförts i laborationsmiljö av Magnus Ellsén. Mätningar gjordes med hjälp av en elektronisk last för att reglera ström och spänning ut från generatorn. Enligt datablad för Windstar 3000 är vindkraftverkets märkeffekt 3000 W [12]. Generatorn producerar en växelspänning som i en diodlikriktare likriktas till en likspänning, till vilken de flesta mätningar refererar till. Mätningarna visade att märkeffekten fås vid ett varvtal på 700 rpm, där strömmen ut ifrån generatorn är 24 A och likspänningen efter diodlikriktaren är 125 V DC. Vid 750 rpm visar mätningarna att märkeffekten överstigs med upp till 496 W för strömmar större än 18 A. Även mätningar då generatorn varit kortsluten och vid tomgång har utförts. Dessa visade att den maximala kortslutningsströmmen efter diodlikriktaren är 38,5 A DC, vilket fås vid varvtalet 750 rpm, och tomgångsspänningen uppgår till 190 V AC från generatorn vid samma varvtal. Effektivvärdet på spänningen blir enligt

$$U_{eff} = \frac{190}{\sqrt{2}} \tag{1}$$

109,7 V AC. Alla mätningar visar på att det maximala momentet sker vid en ström på omkring 28 - 30 A. Resultatet av mätningarna är sammanställda i Appendix A.

1.2 Syfte

Projektets syfte är att undersöka huruvida det är möjligt för Hönö-anläggningen att bli självförsörjande. Detta genom att studera om vindkraftverken tillsammans med solcellspanelerna kan producera mer energi än vad anläggningen konsumerar på årsbasis.

1.3 Projektets problemformuleringar

De problem som ligger till grund för projektet är: avsaknad av ett pålitligt bromssystem till Windstar 3000, avsaknad av ett system som kontinuerligt samlar in data om energiproduktion och konsumtion. Utöver detta är omriktaren inte optimerad för att leverera maximal effekt vid olika vindhastigheter.

1.3.1 Avsaknad av ett pålitligt bromssystem till Windstar 3000

Vid projektets start existerade inget pålitligt, välfungerande bromssystem till vindkraftverket Windstar 3000. Detta är en nödvändighet för att verket ska kunna vara igång utan konstant tillsyn. Utan ett väl anpassat och implementerat bromssystem riskerar vindkraftverket att haverera vid höga vindhastigheter. Det befintliga bromssystemet reagerar efter 500 V DC från diodlikriktaren, vilket är en spänning långt över spänningen vid märkvarvtal, vilken är ungefär 125 V. Därav behöver ett bromssystem anpassat för Windstar 3000 och dess parametrar vid märkeffekt designas och implementeras.

1.3.2 Avsaknad av system för kontinuerlig datainsamling från anläggningen

För att inte behöva befinna sig på anläggningen för manuell insamling av mätdata från energimätarna installerade i arbetsboden, behöver ett program konstrueras som kontinuerligt samlar in relevant data och gör den tillgänglig för projektgruppen. Mätdatan är väsentlig för den statistiska analys och utvärdering av anläggningens energiproduktion och konsumtion som projektet syftar till.

1.3.3 Omriktarens befintliga vind- och effektkurva ger ej maximal uteffekt

Ett av delproblemen i projektet är att omriktaren till Windstar 3000 inte är optimerad för att leverera maximal uteffekt med avseende på den aktuella vindhastigheten. För att åtgärda detta behöver omriktaren programmeras med en lämplig belastningskurva som anpassar vindkraftverket efter vindhastigheten.

1.4 Avgränsningar

Projektet behandlar det befintliga vindkraftverket Windstar 3000. Framtagning och utveckling av bromssystemet till vindkraftverket utförs i laboratorium.

Ingen hänsyn behöver tas gällande energilagring. All producerad energi, som inte förbrukas på anläggningen, matas direkt ut på det allmänna elnätet.

Projektet berör modifikation av Windstar 3000. Datainsamling sker från anläggningens relevanta del, där Windstar 3000 och solcellspanelerna inkluderas.

Tidsramen på cirka fyra månader är begränsande och leder till att slutsatsen om anläggningens energimässiga självständighet delvis utgår från vind- och solstatistik från tidigare år.

2 Teori

Avsnittet behandlar hur energi utvinns ur vind beskrivet med olika ekvationer och samband. Även funktionen av anläggningens energiproducenter samt teorin bakom säkerhetssystemet beskrivs. Därefter beskrivs datainsamling och optimering med teori bakom belastningskurvor.

2.1 Teori bakom vindkraftverks energiutvinning ur vind

Genom att ta till vara på rörelseenergin i vinden omvandlar vindkraftverket energin till elektrisk energi med hjälp av en generator. Vindens rörelseenergi beräknas enligt ekvation 2

$$E = \frac{mv^2}{2} \tag{2}$$

där E är vindens totala energi, m är vindens massa och v är vindhastigheten. Massan kan i sin tur beskrivas som

$$m = At\rho v \tag{3}$$

där A är arean som vindkraftverkets rotorblad sveper över, t är tiden och ρ är luftens densitet. Ekvation 4

$$E = \frac{At\rho v^3}{2} \tag{4}$$

är en sammansättning av ekvation 2 och 3. Då det är effekten i vinden som eftersöks deriveras båda led med avseende på tiden [13]. Detta resulterar i ekvation 5

$$P_{vind} = \frac{A\rho v^3}{2} \tag{5}$$

som visar att då vinden ökar, ökar den totala effekten kubiskt. Detta gör vindhastigheten till den mest avgörande parametern i effektberäkningen. Vindturbiner installeras av den anledningen på höga master där vindhastigheten är högre och mindre turbulens förekommer, jämfört med på marknivå [14][15][16].

En viktig parameter som begränsar möjligheten att utvinna energi ur vinden är vindkraftverkets effektfaktor, C_p. Parametern ingår i följande samband

$$P_{mek} = \frac{C_p(\beta, \lambda)A\rho v^3}{2} \tag{6}$$

där P_{mek} är den mekaniska effekten som överförs mot turbinens axel [15][17][18]. C_p beror på vindens infallsvinkel mot bladet, β, samt λ, även benämnd löptal [19].

C_p beräknas enligt

$$C_p(\beta, \lambda) = \frac{P_{mek}}{P_{vind}} \tag{7}$$

vilket kallas Betz lag. Betz lag beskriver den maximala energin som går att utvinna ur vinden och är teoretiskt sett $\frac{16}{27}$ eller 59,3 procent [17] för ideala vindkraftverk. Detta kommer ifrån en jämförelse av vindens hastighet före och efter turbinbladen [20][21]. I verkligheten finns det dock inga ideala vindkraftverk, utan förluster i verkets olika delar gör att ett bra verk med ostörd vind har ett C_p på ungefär 35 - 50 procent [18][22][23].

Löptalet, λ, beräknas som

$$\lambda = \frac{V_{tip}}{v} \tag{8}$$

och beskriver förhållandet mellan vingspetsarnas hastighet, V_{tip}, och vindhastigheten, v. Vingspetsarnas hastighet beskrivs i sin tur enligt

$$V_{tip} = \omega_m r \tag{9}$$

där ω_m är turbinaxelns mekaniska hastighet och r är turbinbladens längd.

I ideala fembladiga vindkraftverk är $\lambda_{opt} = 5$ [13]. Insättning i ekvation 8, där turbinen snurrar på märkvarvtal, 700 rpm, visar att det skulle krävas vindhastigheter på 21,66 m/s för att komma upp i detta. Märkvarvtalet för Windstar 3000 utgår alltså inte från ideal turbin.

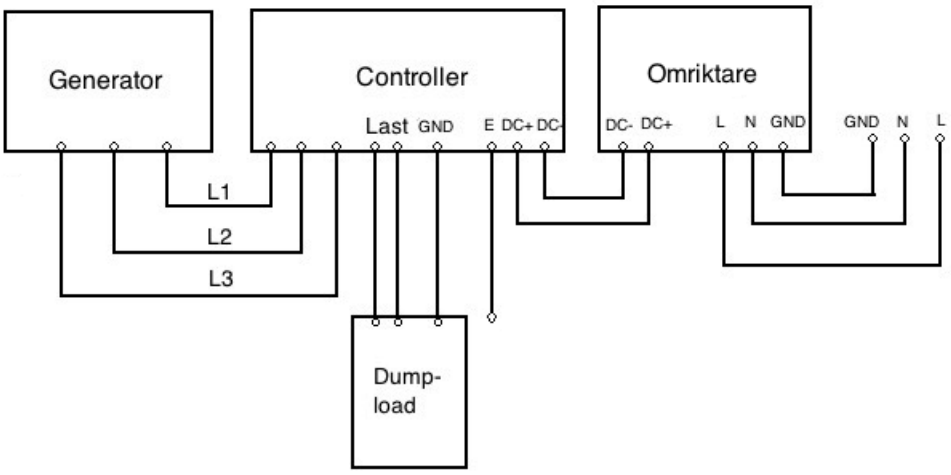
2.2 Vindkraftverket Windstar 3000

Vindkraftverket som projektet omfattar är ett fembladigt verk som enligt tillverkaren ska utvinna upp till en tredjedel mer effekt vid lägre vindhastigheter jämfört med motsvarande trebladiga vindkraftverk [24].

Vindkraftverket har fem förstärkta nylon-glasfiberblad med ett vingspann på 3,05 m, vilket ger en sveparea på 7,3 m^2. Enligt datablad är vindkraftverkets märkvarvtal 700 rpm vid en märkvind på 12 m/s och medelvärde för $C_p \geq 0{,}36$ [12].

2.2.1 Utrustning tillhörande Windstar 3000

I takt med att vindens hastighet och riktning varierar genereras ett skiftande vridmoment och frekvens i samma proportion. Eftersom växelspänning med varierande frekvens inte får skickas ut på nätet, likriktas den, för att sedan återigen växelriktas och skickas ut på nätet med nätfrekvens. En schematisk bild över utrustningen till Windstar 3000 visas i figur 1.



Figur 1: *Schematisk bild över utrustning tillhörande Windstar 3000.*

Windstar 3000 är sammankopplad med en controller, vars uppgift är att skydda verket vid höga vindhastigheter genom att likrikta och mäta spänningen. Den uppmätta spänningen bör ligga i intervallet 40 - 500 V DC. Om spänningen överstiger 500 V DC kopplas en last in med hjälp av en transistor. Genom att lasten kopplas in förbrukas överskottseffekten i denna och omriktaren skyddas mot höga spänningar.

Vindkraftverket har levererats och installerats tillsammans med omriktaren Ginlong GCI-5kW [25] som är sammankopplad med både nätet och controllern. Omriktaren har i uppgift att växelrikta likspänningen och skicka ut den på nätet med nätfrekvens. Det är även i omriktaren som belastningskurvan programmeras för att få så hög uteffekt som möjligt vid samtliga vindhastigheter. Kommunikationen med omriktaren sker via Bluetooth [26].

Två energimätare A43 Gold från ABB är installerade i arbetsboden. Då Windstar 3000, solcellerna och AirDolphin är inkopplade till varsin fas kan data från de enskilda producenterna tas fram. Energimätarna mäter och lagrar anläggningens totala energiproduktion och effekt, samt fasernas individuella värden och kommunikation med mätare sker via Modbus. Förutom den manuella funktionen fanns det inget system för mätning och lagring av dessa parametrar i början av projektet.

2.2.2 Vindkraftverkets läge och inkoppling

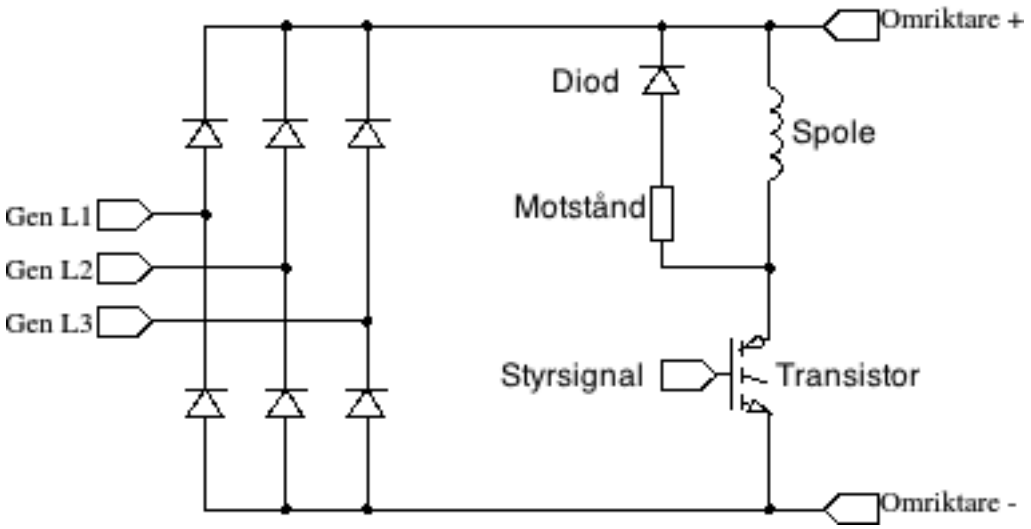
Windstar 3000 är installerad på en 10 m hög mast och är ansluten till arbetsboden via en 55 m lång kopparkabel. Då höjden är relativt låg och med tanke på mängden växtlighet runt anläggningen samt det faktum att bebyggelse omger verket från tre av fyra riktningar, finns risk för turbulent vind från väst, syd och öst.

2.3 Bromssystem till vindkraftverket Windstar 3000

Hård vind kan medföra förödande konsekvenser för vindkraftverk som saknar ett fungerande bromssystem. För att undvika haveri, behöver ett bromssystem designas och implementeras.

2.3.1 Vindkraftverkets funktion för bromsning

Bromssystemet delas upp i två delkretsar; huvud- och frekvenskrets. Huvudkretsen, som schematiskt visas i figur 2, består av en spole, transistor, diod och effektmotstånd. Kretsen hanterar höga strömmar som annars riskerar att förstöra omriktaren. Genom att läsa in växelspänningens frekvens via en fas från verkets generator kan turbinens varvtal bestämmas. Frekvenskretsen innehåller komponenter för omvandling av växelspänningens sinusvåg till en fyrkantsvåg och analys av signalen reglerar transistorn i huvudkretsen. Signalen från frekvenskretsens Arduino kan vara antingen hög eller låg, där den låga signalen tyder på att varvtalet är för högt och verket behöver bromsas ner. För att bromsa vindkraftverket krävs en motriktad ström in i generatorns utgång för induktion av ett bromsande magnetfält. Detta görs genom att transistorn växlar mellan att leda och inte leda. Då transistorn leder går strömmen in i generatorn och i det andra fallet förbränns den lagrade energin från spolen i effektmotståndet. Först när vindkraftverket har bromsats ner kortsluts generatorns tre faser med hjälp av ett relä för att få det att stanna helt.

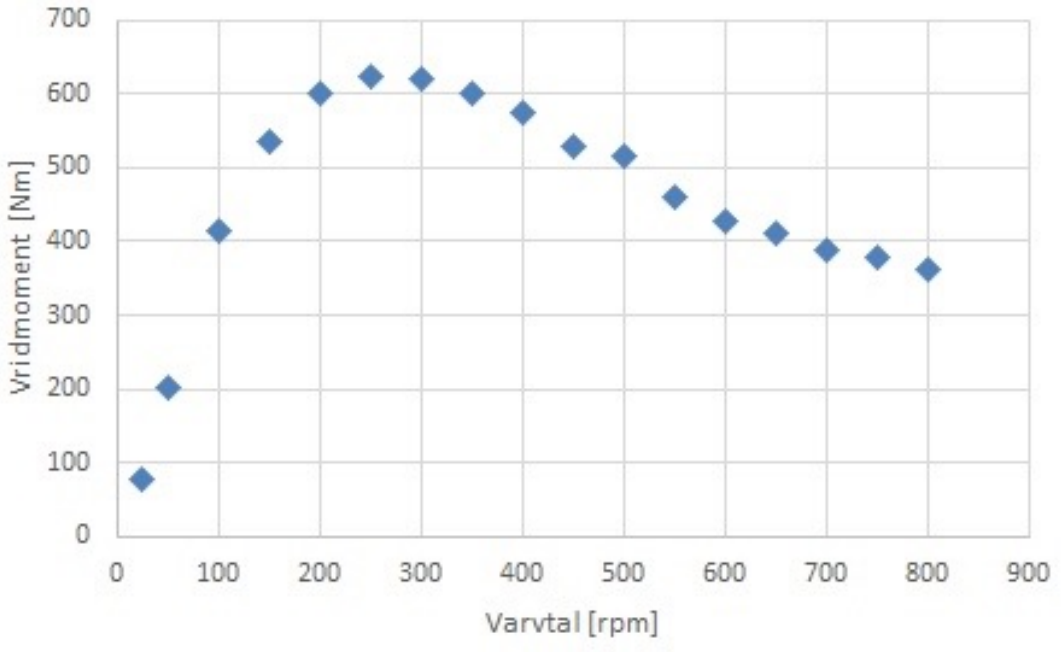


Figur 2: *Generellt schema över huvudkretsen och dess komponenter kopplade till generatorns tre faser.*

2.3.2 Bromssystemets transistorstyrning

För att bromsen ska fungera, behöver transistorn öppnas och slutas med ett visst tidsintervall. En sluten transistor skulle snart resultera i en kortslutning från positiv till negativ sida. Spänningen mellan plus och minus blir då noll och därmed kommer

den uttagna effekten ur generatorn, enligt effektformeln också att vara noll. Effekten konsumeras då istället i statorns lindningar. Förutom det faktum att en kortslutning kan överhetta och skada statorlindningarna är det inte säkert att rotorns varvtal sjunker. Generatorns momentkurva är inte linjär, vilket medför att motverkande vridmoment - rotorns bromsande effekt mot vinden - inte nödvändigtvis ökar med ett ökat strömuttag, vilket kan ses i figur 3.

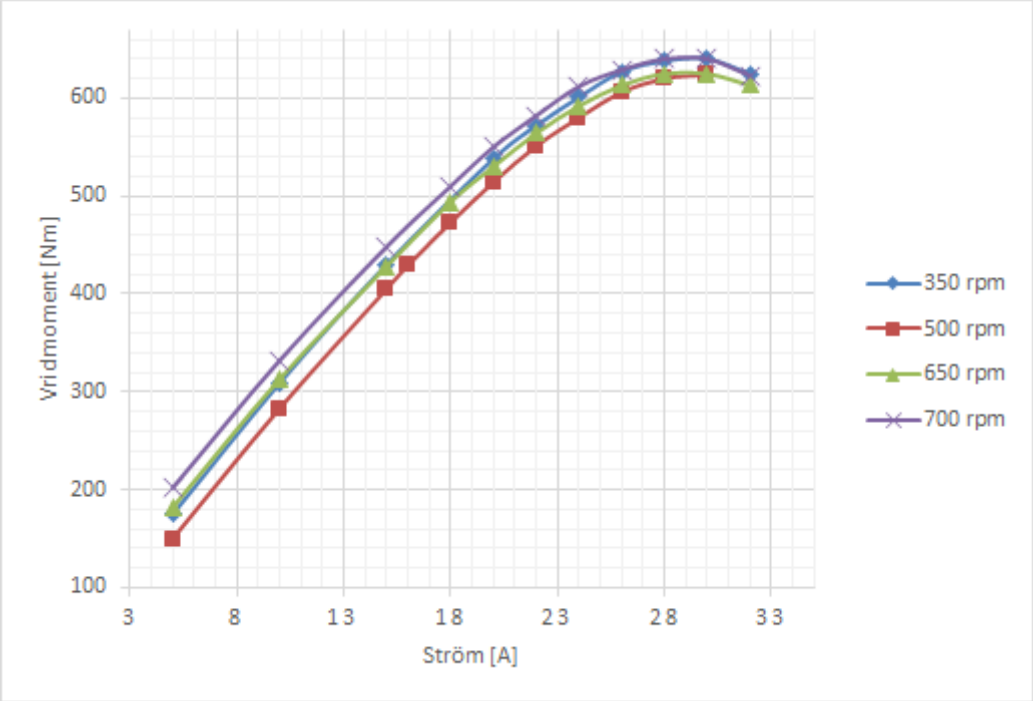


Figur 3: *Momentkurva vid kortslutning av generatorn. Figuren grundas på mätningar i Appendix A utförda av Magnus Ellsén.*

Om det ursprungliga varvtalet är högt kan det innebära att det motverkande momentet istället sjunker vid en kortslutning, vilket leder till att varvtalet ökar.

Transistorn behöver således styras så att strömuttaget från generatorn hålls på en optimal nivå för maximalt bromsande moment. Detta kan göras med hjälp av en så kallad hysterescontroller. Controllern läser av total uttagen ström från generatorn och låter transistorn sluta kretsen när strömmen ligger under ett fördefinierat börvärde och öppnar kretsen när strömmen nått en maxgräns. Den uttagna strömmen pendlar därför inom ett intervall när bromssystemet är aktiverat.

En del mätningar i laboratoriemiljö har utförts före projektets start och ger infomation om vridmoment, strömmar, spänningar och effekt. En graf över vridmoment mot uttagen ström vid olika varvtal visas i figur 4. Ur grafen utläses att turbinens maximala vridmoment återfinns vid ett strömuttag runt 28 - 30 A. Således är det inom detta intervall som strömuttaget bör vara för att uppnå maximalt bromsande moment. När varvtalet sjunker kommer det till slut bli omöjligt att upprätthålla korrekt strömuttag. Eftersom hysterescontrollern matas med ett konstant börvärde så kommer den fortsätta att arbeta för att uppnå börvärdet, vilket i praktiken innebär att transistorn leder hela tiden och därmed kortsluter kretsen. Varvtalet fortsätter då att sjunka mot noll. Vid låga varvtal är det lämpligt att kortsluta kretsen med ett relä för att stoppa vindkraftverket helt.



Figur 4: *Vridmoment vid olika varvtal och ström. Figuren grundas på mätningar i Appendix A utförda av Magnus Ellsén.*

2.3.3 Varvtalsmätning

Enligt mätningar genomförda i laboratorium på vindkraftverkets spänningsutveckling med avseende på strömuttag och varvtal, utvecklas den maximala effekten vid 750 varv per minut vilket kan ses i Appendix A.

Vindturbinens varvtal behöver registreras kontinuerligt för att kunna bromsa turbinen om 800 rpm skulle överskridas. Eftersom fasfrekvensen på synkrongeneratorn direkt beror av rotorns varvtal kan frekvensen mätas för att beräkna turbinens varvtal. Varvtalet ges av ekvation 10

$$n = \frac{f * 60}{p} \tag{10}$$

där n är rotorns varvtal, f är fasfrekvensen och p är generatorns antal polpar per fas. Polpartalet för Windstar 3000 anges i dess datablad till fem [27]. Vid märkvarvtal blir frekvensen alltså 58,33 Hz och vid 66 Hz ska bromsfunktionen aktiveras.

2.4 Solcellspanelerna Naps Pallas specifikationer

Anläggningen har 15 stycken Naps Pallas 230 SM3 PBB-paneler installerade. Enligt datablad har varje solpanel en märkeffekt på 230 W och den totala effekten kan uppgå till 3450 W [28]. Panelerna har ett ytskikt av albarinoglas som bidrar till ökad transmission av energi till solcellerna [29]. Solcellerna är seriekopplade för att anpassa spänningen

in till växelriktaren med en undre gräns på 350 V. Detta på grund av att nominell inspänning till den enfasiga växelriktaren StecaGrid, som matar ut spänning på nätet, är 350 V DC. Panelerna är kopplade till växelriktaren via en fem meter lång förtennad kopparkabel med 4 mm^2 tvärsnittsarea.

StecaGrid 3600 är även en energimätare anpassad för solpanelerna som matar effekten ut på nätet med nätfrekvensen. Apparaten lagrar mätdata och kommunicerar via RS-485 samt har en inbyggd reglerfunktion Maximum Power Point Tracking, MPPT. MPPT söker kontinuerligt efter en kombination av ström och spänning som resulterar i maximal uteffekt [30].

2.5 Insamling av data

För utvärdering av anläggningen behöver mätningar på både produktion och konsumtion av energi göras. För detta behöver ett system konstrueras som kan läsa in och spara informationen.

2.5.1 Hönö-anläggningens energimätare

Två energimätare A43 Gold från ABB är sedan tidigare installerade i arbetsboden. En av mätarna loggar energiproduktion från solpanelerna samt produktion från vindkraftverken Windstar 3000 och AirDolphin. Den andra mätaren loggar anläggningens totala energikonsumtion. Båda mätarna kan kommunicera genom protokollet Modbus RTU över RS-485.

2.5.2 Modbus och RS-485

Modbus är ett kommunikationsprotokoll publicerat år 1979 av företaget Modicon. Idag är protokollet fritt och har blivit en industriell de facto standard för kommunikation mellan elektroniska enheter [31].

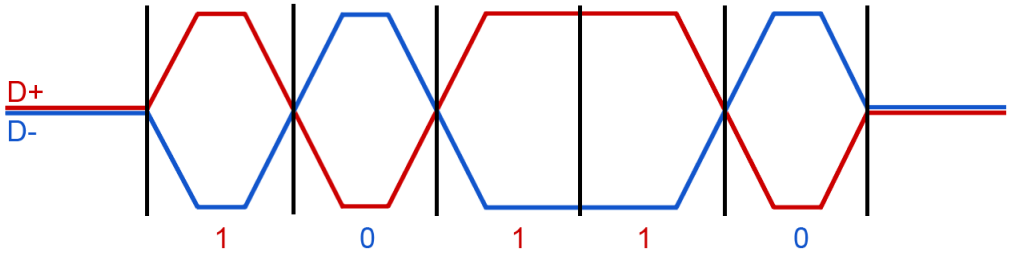
Modbus är uppbyggt med master/slave-representation. I ett nätverk finns en adresslös master och upp till 247 slavenheter kan existera, där alla slavenheter har en unik adress. Masterenheten är ensam om rättigheten att initiera kommunikation på bussen och slavarna får endast svara på begäran. Slavenheter tillgängliggör information i så kallade register, som antingen kan läsas från eller skrivas till på begäran av masterenheten. Varje register är 16 bitar stort, men de grupperas ofta i grupper om två eller fyra för att möjliggöra stora datavärden [31].

Ett typiskt modbuspaket som en masterenhet skickar för att begära data från en slavenhet visas i figur 5. Adressen bestämmer vilken slav som ska svara. Funktionskoden talar om för slavenheten vilken operation som ska utföras, exempelvis att läsa ett register, medan registeradressen specificerar vilken data som efterfrågas. Genom “antal register” kan information fås från flera efterföljande register med enbart en begäran.

Figur 5: *Innehåll i ett typiskt modbuspaket.*

Rätt slavenhet svarar på begäran med ett liknande meddelande, vilket inkluderar den efterfrågande informationen [31].

RS-485 är en hårdvarustandard för seriell kommunikation som skapar förutsättning för flera enheter att kommunicera, en i taget, över två tvinnade trådar. Kabellängder upp till 1200 meter och hastigheter upp till 10 Mbit/s tillåts [32]. Standarden använder en differensspänning för att definiera en etta eller en nolla. Spänningen i trådarna avviker lika mycket från referenspunkten, fast den ena positivt och den andra negativt. En skillnad på 200 mV mellan referens och spänningen i trådarna räcker för att detektera ett värde [32]. En illustration över en kort datasekvens kan ses i figur 6.



Figur 6: *Datasekvens över RS-485. Då ingång/utgång D+ har högre spänning än D- detekteras detta som en etta. Då D+ har lägre spänning D- blir detta istället en nolla.*

Notera att standarden enbart beskriver kommunikation på hårdvarunivå och inte tar hänsyn till hur informationen överförs rent logiskt. Energimätarna använder därför RS-485 för att fysiskt koppla samman mätarna med en masterenhet, medan Modbusprotokollet beskriver hur kommunikationen går till [31][33].

2.5.3 Enchipsdatorer

För att kommunicera med energimätarna behövs en enhet som kan agera Modbus-master. Valet föll på enchipsdatorerna Arduino och Raspberry Pi för detta ändamål. Mer om detta val följer i metod-kapitlet.

En enchipsdator eller mikrocontroller är en komplett dator, bestående av CPU, minne och I/O-portar. En enhet är oftast liten, billig och är lämplig för en rad olika ändamål. Exempelvis kan den fungera bra som en central styrenhet i både små och stora elektroniska kretsar som programmeras för att utföra en viss syssla [34].

Raspberry Pi[1] är en liten ARM-baserad enchipsdator som kan köra en fullfjädrad

[1]Raspberry Pi is a trademark of the Raspberry Pi Foundation

linuxdistribution. Förutom nätverksport, HDMI- och USB-anslutningar har den även rena digitala in- och utgångar som kan programmeras efter önskemål. Raspberry Pi utvecklades som ett billigt läromedel för barn som vill lära sig förstå datorer och programmering. Den har dock fått stor spridning utanför den tilltänkta målgruppen och används ofta som centralenheter i prototyp- och hobbysammanhang där kostnad, storlek, funktion och effektförbrukning är viktiga faktorer [35].

Arduino är ett projekt inom kategorin öppen mjukvara/hårdvara som utvecklar enchipsdatorer. De finns i flera olika versioner men de flesta baseras på Atmels AVR ATmega-processorer [36]. Något operativsystem behövs inte, utan den kör maskinkod direkt på processorn. Ett arduinoprogram skrivs i en C-dialekt [36], kompileras och skickas sedan över via RS-232 (COM-port) eller USB. Ett arduinokort har ett antal analoga och digitala in- och utgångar [37], vilka kan användas till vad som önskas. En Arduino har goda möjligheter till att styra funktioner på hårdvarunivå men det är mer arbete att bygga mer abstrakt mjukvara så som webbfunktioner. Därför kan det vara lämpligt att kombinera Arduino och Raspberry Pi för att kunna arbeta med ett brett spektrum från hårdvara till mjukvara.

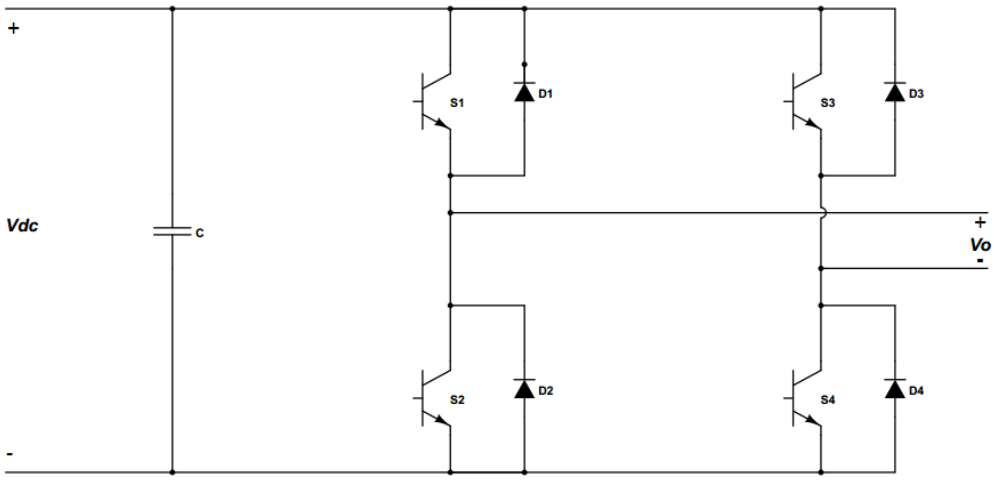
2.6 Optimering av vindkraftverket

För optimeringen av energiproduktionen hos Windstar 3000 kommer anläggningens redan installerade omriktare Ginlong (GCI-5KW) användas. Omriktaren har en maxström på 25 A vilket ger en linjär begränsning i uteffekt [25]. Den anpassar sig efter enfas och anpassar energin från kretsen efter elnätets 230 V och 50 Hz. Kopplingsschema för en enfas-omriktare ses i figur 7. Då Ginlong Technologies inte vill lämna specifikationer är detta ett generellt kopplingsschema för en enfas-omriktare bestående av fyra dioder i par med varsin transistor. Kondensatorn är till för att skydda omriktaren från strömspikar och skapa jämnare flöde in i omrikteran. V_{dc} är spänningen från DC/DC-omriktaren och V_0 är spänningen ut på nätet.

En belastningskurva programmeras in i omriktaren, där effekten är en funktion av spänningen med strömmen som variabel. Belastningen som läggs på är i form av strömuttag. Ökar uttaget av ström ökar också effekten.

För enklare programmering av omriktaren används en mjukvara vid namn PowerMonitor, utgivet av Ginlong Technologies som fungerar för PC. Detta program ansluts sedan via Bluetooth till omriktaren. I programmet visas ström och spänning in till omriktaren samt den på nätet uttagna ström, spänning och effekt. Omriktaren tillåter ändring på var tionde volt för att sedan medelvärdesbilda en följsam kurva efter vilken vindkraftverket rättar sig.

När den optimala belastningskurvan fastställts för Windstar 3000 kommer uteffekten vara högsta möjliga i förhållande till vindhastigheten. Med hjälp av insamlad data kommer en vind-effektkurva kunna sammanställas och jämföras med den vind-effektkurva tillverkaren menar att Windstar 3000 följer.



Figur 7: *Kopplingsschema för en enfas-omriktare [27].*

Med hjälp av olika tester på anläggningen och beräkningar på fembladiga vindkraftverk designas en belastningskurva till Windstar 3000. På plats hjälper LabVIEW till att plotta olika kurvor med aktuella värden för att kunna bedömma hur olika de faktiska kurvorna ser ut i förhållande med de teoretiskt beräknade.

3 Genomförande

Då projektet innefattar fyra delprojekt delades arbetsuppgifterna in i olika undergrupper och utformningen av dessa grupper varierade under projektets gång. Uppdelning av projektet har lett till ett konstant parallellt arbete med de olika delarna.

Majoriteten av arbetet rörande vindkraftverkets bromssystem har skett i laboratorium på Chalmers. Framtagning av detta gjordes inledningsvis med hjälp av simuleringsprogram vilket sedan testades genom praktiska försök och experiment. En kravspecifikation togs fram som systemet anpassades efter.

Utan en automatisk datainsamling fanns det ingen möjlighet att samla in och analysera data utan att behöva vara ute på anläggningen. Hårdvarukomponenter behövde därför väljas och tillsammans med mjukvara möjliggöra regelbunden insamling av data för extern avläsning.

Vid optimering av vindkraftverket behövdes en belastningskurva till omriktaren tas fram. Framtagningen av vindkraftverkets kurva genomfördes dels genom tester på plats på Hönö och dels via beräkningar, vilka illustrerades i grafer. Dessa två tillvägagångssätt kommer att förklaras närmare i detta avsnitt.

För undersökning om anläggningens nettoproduktion under ett år gjordes en statistisk undersökning gällande vindhastighet och antalet soltimmar för samtliga månader under en tre- respektive fyra- och tolvårsperiod. Dessa parametrar ligger till grund för att räkna ut den teoretiska årsproduktionen för både Windstar 3000 och Naps Pallas.

3.1 Installation av Windstar 3000

Tidigare projektarbete som behandlade hybridanläggningen på Hönö var också de som beställde vindkraftverket Windstar 3000 och solcellspanelerna. De konstaterade Windstar 3000s bristande bromssystem och vindkraftverket monterades ned efter 20 dagar [26].

Efter ungefär halva projekttiden åkte projektgruppen tillsammans med handledaren ut till Hönö och monterade ihop samt reste vindkraftverket. Bromssystemet var inte redo för inkoppling vid den tidpunkten, därav kunde verket endast vara igång under bevakning. När bromssystemet var färdigt för test ute på anläggningen inträffade ett svårförklarat fel, vindkraftverket varvade inte upp trots relativt goda vindförhållanden. Verket monterades ner för analys. Ingen märkbar avvikelse noterades och vindkraftverket restes igen.

3.2 Design och implementering av vindkraftverkets bromssystem

Det framtagna bromssystemet, baserat på en befintlig krets som använts i ett liknande system, testades och verifierades i simuleringsprogrammet Circuit Simulator. En nedskalad version av kretsen kopplades därefter upp på en testplatta för att undersöka om systemet fungerar som beräknat. Mätningar på ström och spänning med både oscilloskop och multimeter genomfördes för att undersöka strömmars storlek och hur de flyter i kretsen. Uppmätta värden i Appendix A visade på att den maximala strömmen genom kretsen är 38 A DC. Detta kan dock variera beroende på temperatur, därför har ett påslag på tio procent gjorts. Alla komponenter med strömintag från generatorn bör därför klara en likström på 42 A DC.

3.2.1 Kravspecifikation för bromssystemet

Systemet ska kunna genomföra en säker inbromsning och parkering av vindkraftverket vid olika vindförhållanden. Vindkraftverket ska kunna startas och stoppas med fysiska knappar lokalt på Hönö, men även fjärrstyrning över Internet ska vara möjlig. Vid fel på nätet eller strömavbrott ska bromssystemet kunna drivas av en alternativ spänningskälla. Skulle backupbatterierna ta slut ska inte vindkraftverket kunna starta av sig själv utan kräva en manuell uppstart. Kraven hänvisar även till att bromssystemet ska vara autonomt, alltså själv detektera om varvtalet blir för högt och då starta inbromsningen. För att veta när inbromsning pågår bör en signal vara synlig utifrån.

3.2.2 Indelning av bromssystemet

Systemet delas upp i två delkretsar. Den första benämns som huvudkrets och består av en spole, diod, ett effektmotstånd och en transistor med tillhörande styrkomponenter. Figur 2 visar en generell bild över hur huvudkretsen är uppbyggd. Transistorns styrkomponenter har till uppgift att, beroende på frekvens från generatorn eller strömavbrott på nätet, öppna eller stänga transistorn. Frekvensmätningen, som är den andra delkretsen, läser in spänningskurvan från en av faserna från generatorn och omvandlar generatorns sinusvåg till en fyrkantsvåg samt transformerar ned spänningen, vilket möjliggör mätning av frekvensen.

3.2.3 Komponenter till bromssystemets huvudkrets

Vid val av effektmotstånd behövdes hänsyn tas till vilken maximal effekt som bör kunna omvandlas till värme. Simulering av kretsen resulterade i en toppeffekt på 10 kW under ett pulsat intervall med frekvensen 10 kHz, vilket motsvarar en periodtid på 0,1 ms. På grund av detta valdes två parallellkopplade 22 Ω motstånd, som enskilt kan ta hand om en märkeffekt på 2 kW. Enligt datablad kan motstånden under en tidsperiod på fem sekunder ta hand om upp till tre gånger märkeffekten [38], vilket motsvarar en total effekt på 12 kW. Dioden, S70M [39], som sitter direkt efter motståndet klarar av en ström på 70 A samt en backspänning på 1 kV, vilket med god marginal är inom de gränser som dioden maximalt utsätts för.

Spolens induktans beräknas till L= 5,7 mH utifrån sambandet

$$V = L * \frac{di}{dt} \tag{11}$$

där strömintervallet är 2 A och tidsintervallet är periodtiden 0,1 ms. Utifrån detta, samt att järnpulverkärnor har ett högre magnetiskt mättnadsvärde än ferritkärnor, väljs en toroid järnpulverkärna, T400-26 av Micrometals. Kärnans induktansfaktor A_L är 131 nH/N^2, vilket också innebär att antal varv som kärnan ska lindas med är $N = 208$. Detta utifrån ekvation 12

$$N = \sqrt{\frac{L}{A_L}} \tag{12}$$

Tråden som kärnan lindas med är en 3 mm koppartråd. Den totala trådlängden blir 18 m och resistansen i tråden blir 40 mΩ.

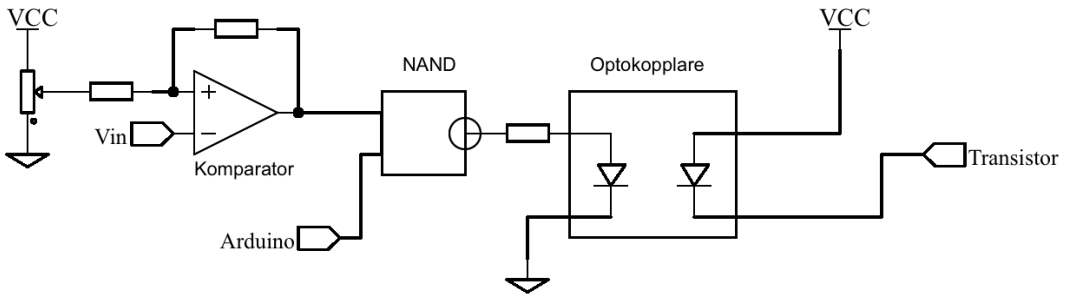
Transistorn i kretsen är en IGBT-transistor [40]. Enligt datablad klarar komponenten av en spänning på 1200 V och en nominell DC-ström på 100 A. Båda värden ligger utanför de intervall som maximalt belastar kretsen.

Komponenterna som utgör kretsens transistorstyrning består av en komparator, en NAND-grind och en optokopplare. En bild över transistorstyrningens uppbyggnad illustreras i figur 8. Då strömmen genom spolen och transistorn bör hålla sig inom ett hysteresband behöver komparatorn ställas in utifrån detta. En strömtransducer [41] placerad vid diodlikriktarens utgång, mäter och omvandlar strömmen från diodlikriktaren till en spänning som går in i komparatorns minussida, markerat V_{in} i figur 8. Enligt datablad motsvarar en ström på 28 - 30 A uppmätt av transducern en transformerad spänning på 8,4 - 9,0 V, vilket blir hysteresbandets gränser. Detta strömintervall väljs då, enligt tidigare mätningar, generatorns största moment fås vid en likström på 28 - 30 A produceras, vilket i sin tur resulterar i en effektiv inbromsning. Transducern valdes då den klarar av kretsens maximala ström samt att den matas med 12 V, som finns tillgängligt från batterierna. Komparatorn, kopplad i en inverterande krets konstruerar ett hysteresband genom att en referensspänning från potentiometern är kopplad till komparatorns två motstånd [42]. Därefter jämför komparatorn spänningen V_{in} med referensspänningen och därigenom bildas en fyrkantsvåg med höga toppar då referensspänningen är högre än V_{in}. Framtagning av en lämplig referensspänning beräknades enligt ekvation 13 och 14

$$\frac{R_2}{R_1} = \frac{V_{High} - V_{Low}}{V_{hyst}} \tag{13}$$

$$V_{TH_+} = \frac{R_1}{R_2} * (V_{REF} - V_{Low}) + V_{REF} \tag{14}$$

där R_1 och R_2 motsvarar komparatorns motstånd, V_{High} och V_{Low} är 12 respektive 0 V, vilket är den spänningsnivå som toppar och dalar i fyrkantsvågen har. V_{hyst} refererar till skillnaden mellan hysteresbandets högre och lägre nivå, alltså 0,6 V. R_1 och R_2 sattes



Figur 8: *En förenklad illustration över det framtagna systemet för styrning av transistorn i huvudkretsen.*

till 50 kΩ respektive 1 MΩ för att motsvara kvoten i ekvation 13. Insättning i ekvation 14 gav en referensspänning på 8,57 V, som ställdes in med hjälp av potentiometern.

NAND-grinden är en inverterande grind som kopplas till både komparatorns utgång och bromssystemets Arduino. Transistorn leder då det ligger två nollor på grindens ingång. Den andra nollan erhålles från frekvenskretsens Arduino om vindkraftverkets varvtal är högre än 800 rpm. Grinden matas med ± 12 V genom batterierna [43]. Med hjälp av ett motstånd på 750 Ω direkt efter grinden anpassas strömmen in i optokopplaren, som enligt datablad ska vara mellan 10 - 16 mA. Optokopplarens uppgift är att frånskilja transistorn galvaniskt från resterande del av styrkretsen [44]. Detta är för att undvika att eventuella jordströmmar och störningar transfereras till transistorn. Utgången på optokopplaren spänningssätts med 15 V vilket görs med en DC/DC-omvandlare [45] från de 24 V som batterierna kan lämna.

Kretsen utgör ett samarbete mellan mätning av vindkraftverkets frekvens och den ström som generatorn levererar. Transistorn öppnas och stängs i den takt som strömmen genom transducern når de övre och undre gränserna i det satta hysteresbandet.

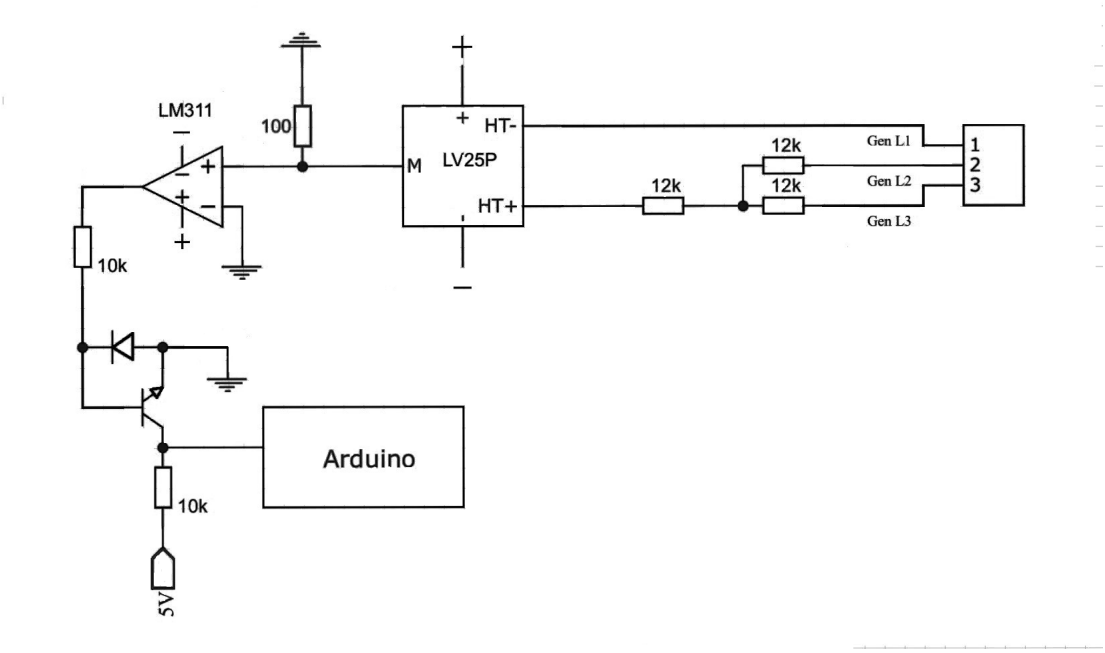
3.2.4 Komponenter till systemet för frekvensmätning

Vid design av kretsen för frekvensmätning ritades kretsen upp och komponenternas värden beräknades teoretiskt. Genom simulering i bland annat programmet Circuit Simulator kunde toppvärden för potentialer och strömmar antas. Simuleringen gav även en bra bild över systemet och visade på att det fungerade som beräknat. Då systemet var väl teoretiskt utvecklat monterades det på en kopplingsplatta för verifiering och för att ge en uppskattning av hur värden mellan det teoretiska och det verkliga fallet skiljer sig. Kretsen på testplattan modifierades för att uppnå de sökta egenskaperna och därefter ritades kretsen upp i programmet EasyEda för konvertering till ett kretskort. För både schemat och kretskortet till frekvenskretsen se Appendix L.

Spänningsomvandlaren LV25-P isolerar kretsen galvaniskt från generatorn och mäter spänningen från en fas [46]. Tidigare mätningar med avseende på varvtal vid tomgångsspänning visade att vid 750 rpm fås en toppspänning på 190 V, vilket

motsvarar en nominell spänning på 109,7 V per fas. Datablad för spänningsomvandlaren förordar att strömmen in ej får överstiga 10 mA. Detta fås genom Y-kopplingen som visas i figur 9, där två av faserna samt utgången är kopplade till varsitt 12 kΩ motstånd. Sinusvågen från den första fasen dämpas i spänningsomvandlaren. Därefter, med hjälp av ett jordkopplat motstånd på utgången mäts och skickas signalen till komparatorn LM311. Komparatorns uppgift är att omvandla sinusvågen till en fyrkantsvåg. Referensspänningen till komparatorn sätts till strax över 0 V via en potentiometer. På så sätt matar komparatorn inte ut någon fyrkantsvåg om inspänningen är för låg för att kunna särskiljas från brus. Utsignalen från komparatorn har en spänning på ± 15 V. Då mikroprocessorn, av modell Arduino Nano, endast kan ta emot en signal som är mellan 0 och 5 V måste signalen modifieras. Med hjälp av en pålagd transistorkrets med diod och motstånd fås en fyrkantsvåg som går mellan de önskade värdena. Arduinon läser därefter av frekvensen genom att mäta periodtiden mellan vågens toppar. Den fullständiga programkoden för Arduinon finns i Appendix D.

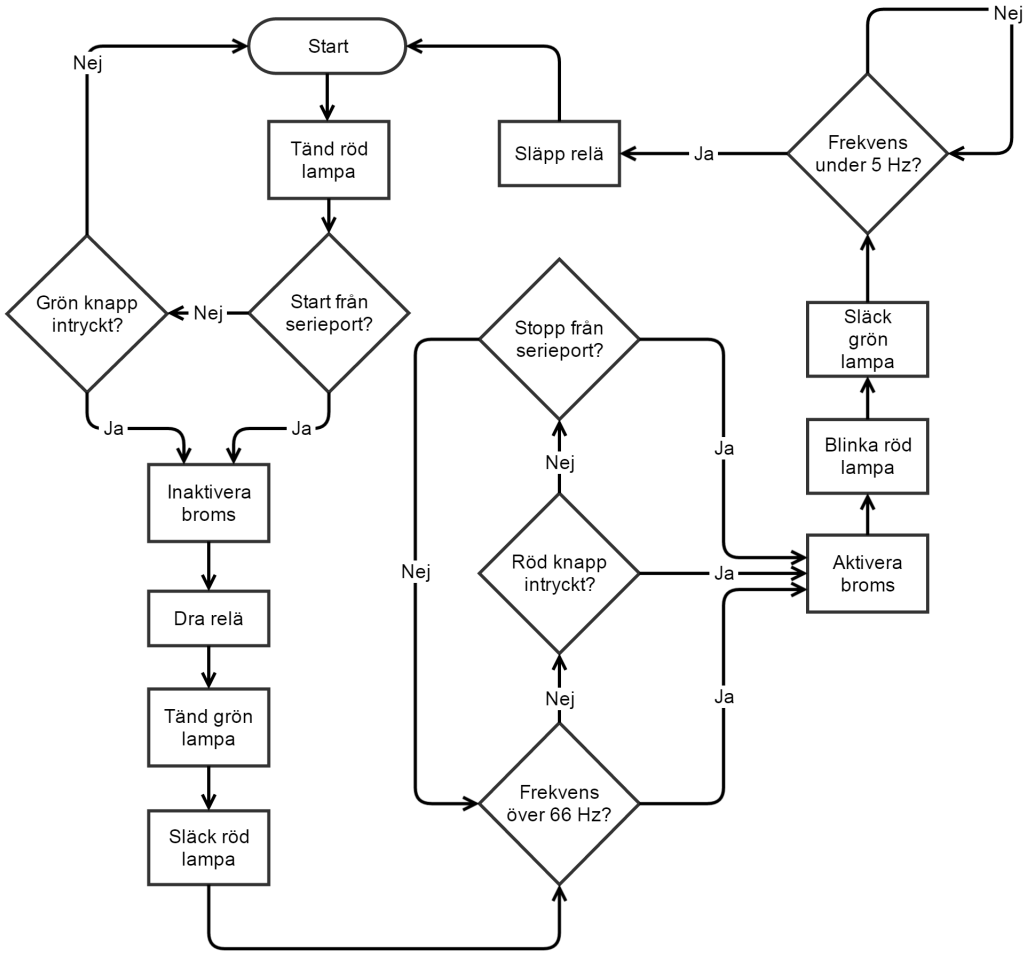
Mätningar genomförda på generatorn innan projektets start visade att 3 kW fås först vid 750 rpm. För att undvika att gå miste om hög effektutvinning, sätts bromsfunktionen igång först vid 800 rpm. Arduinon beräknar frekvensen och om den stiger över 66 Hz, motsvarande 800 rpm, skickas en nolla ut och bromssystemet startas. Vid strömavbrott stängs omriktaren av och vindkraftverket blir olastat. Detta medför att varvtalet stiger och bromssystemet aktiveras genom Arduinon.



Figur 9: *En förenklad illustration över det framtagna systemet för frekvensmätning.*

Då bromssekvensen är aktiv och Arduinon mäter en frekvens underskridande 5 Hz kortsluts generatorn genom två parallellkopplade reläer kopplade till alla tre faser. Dessa gör att vindkraftverket stannar helt.

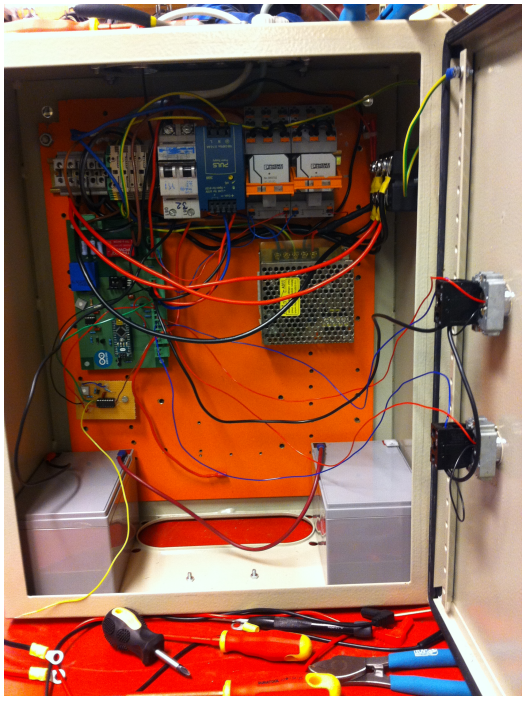
Det framtagna kretskortet monterades tillsammans med batterierna och reläerna i ett plåtskåp. I figur 11 kan den färdigmonterade kretsen ses. Systemet innefattar även en manuell på- och avstängning av bromsfunktionen. Detta genom att Arduinon är kopplad till två knappar, en röd och en grön. Vindkraftverket startas manuellt genom att trycka på den gröna knappen, som då börjar lysa. Då bromssystemet är aktivt släcks den gröna knappen och den röda börjar blinka. Övergången kan skötas manuellt genom en knapptryckning. Då varvtalet är såpass lågt så att reläerna slår till, lyser den röda knappen med konstant ljus. På så vis är det möjligt att observera om vindkraftverket är igång eller inte. Bromssystemet kan även aktiveras och avaktiveras från en dator via serieporten. På så sätt går det att fjärrstyra vindkraftverket via Internet. Ett flödesschema över hela styrprogrammet visas i figur 10.



Figur 10: *Programflödet hos bromssystemets Arduino.*

3.2.5 Den slutliga kretsen för bromssystemet

På grund av tidsbrist har endast frekvenskretsen hunnit byggas, testas och tillämpas. Huvudkretsen har simulerats och komponenter till denna har valts. En huvudkrets, liknande den teoretiskt framtagna har använts, men med de beräknade effektmotstånden. Denna krets kommer från projektet ”Vindkraftverk och fartyg” [47]



Figur 11: *Den monterade frekvenskretsen i ett plåtskåp tillsammans med reläer, batterier och dörrmonterade knappar.*

och har samma funktion som den beräknade kretsen, trots att vissa komponenter är ordnade på ett annat sätt.

Transistorn som finns i den använda kretsen är en IGBT transistor [48], med en inbyggd diod som substituerar dioden i den tänkta kretsen. Transistorn har liknande specifikationer och klarar av belastningarna från generatorn. Spänningsfallet över transistorn får enligt datablad vara maximalt 1200 V och nominell ström får uppgå till 150 A. Vissa komponenter i den lånade huvudkretsen är dessutom samma som i den tänkta. Spolkärnan och koppartråden är densamma som i den framräknade kretsen, men med färre antal varv.

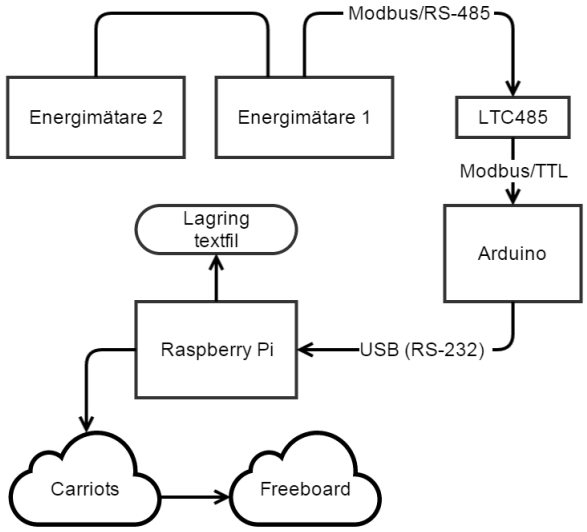
Delkretsen för transistorstyrning läser även av strömmen från diodlikriktaren, vilket även var implementerat i det beräknade systemet. Efter en del operationer med de uppmätta strömvärden, som innebär kontroll över strömmens befintlighet inom korrekt intervall, skickas en signal ut. Signalen i kombination med en signal från frekvenskretsens Arduino styr slutligen transistorn.

Den slutliga kretsen har testats i laboratorium med ett likspänningsaggregat, som simulerar den ström och spänning som matas ut från diodlikriktaren. Aggregatet kunde dock maximalt generera 20 A men upp till 300 V. Systemet har testats med 100 V, 20 A och fungerat felfritt så långt. Vid detta test ändrades frekvenskretsen till att reagera på strömmar under 20 A, för säkerställning av att metoden fungerade och test av systemets egna reglerfunktioner. Vid installationen av systemet på anläggningen har bromsfunktionens felfrihet inte kunnat säkerställas på grund av att turbinen inte kunnat komma upp i högre varvtal än 96 rpm. Vindkraftverket tycks inte fungerat som det skulle

men inget uppenbart fel kunde noteras vid mätningar. Vindkraftverket fälldes ner och undersöktes, men inte heller det gav något tydligt svar på varför rotorn inte går upp i varv. Tidsbrist hindrar vidare felsökning.

3.3 Insamling av information från energimätarna

Inledningsvis valdes ett kommunikationssätt som energimätarna stödjer och en studie gällande detta gjordes. Det valda kommunikationssättet blev protokollet Modbus då det är en industristandard och ger goda möjligheter att nå all information som mätarna skapar. Utifrån detta skrevs programkod för att styra två enchipsdatorer, en Raspberry Pi och en Arduino. I systemet läser Arduinon av energimätarna och de efterfrågade värden sparas på Raspberry Pi. En del av insamlad data skickas till en molntjänst för visualisering av energiproduktion från solpanelerna och Windstar 3000 samt anläggningens energikonsumtion. En översiktsbild över dataflödet visas i figur 12. Komponenterna kommer nedan att gås igenom mer utförligt.

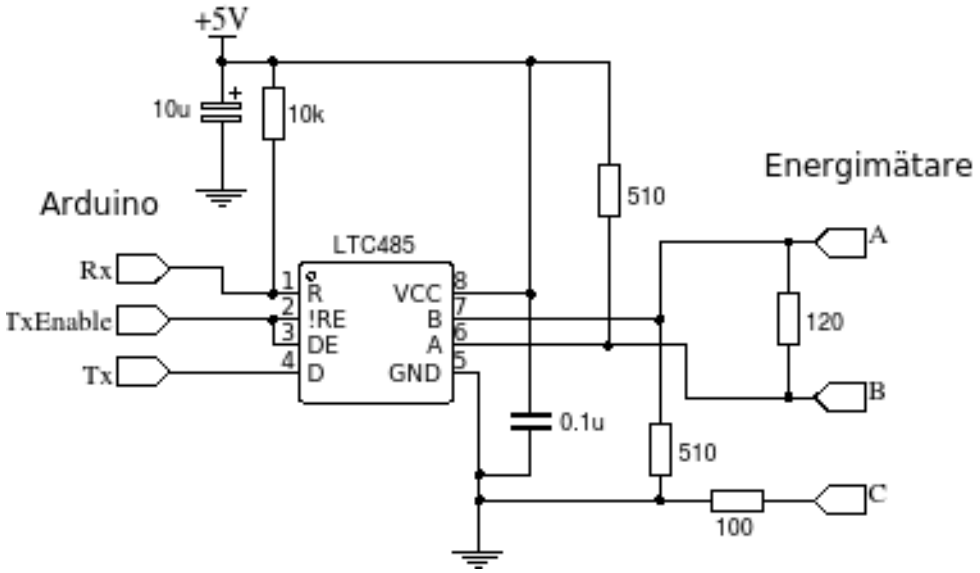


Figur 12: *Översikt över hela mätsystemet.*

3.3.1 Hårdvarukomponenter för kommunikation

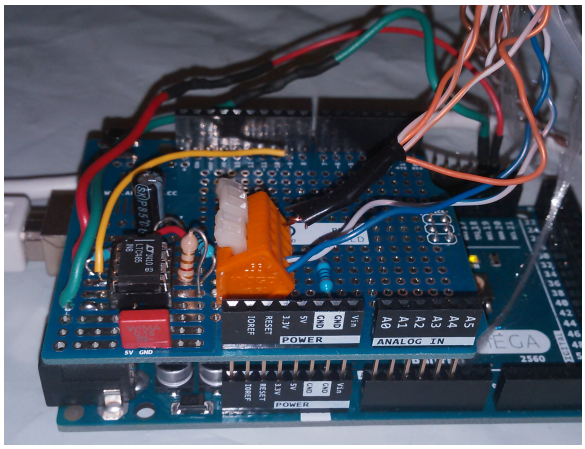
Istället för att börja arbeta med obekant hård- och mjukvara valdes Arduino och Raspberry Pi. Kunskap och erfarenhet om dessa plattformar fanns redan vid projektets start inom gruppen. Plattformarna har dessutom stor användarbas och är väldokumenterade - det är lätt att finna hjälp på Internet om problem uppstår. Arduino och Raspberry Pi finns tillgängliga i ett antal olika modeller. Raspberry Pi 2 modell B samt Arduino Mega 2560 valdes. Vad Raspberry Pi beträffar valdes aktuell modell då den är den senaste och snabbaste versionen. Någon prisskillnad mellan den senaste och den äldre modellen fanns inte. För att kunna kommunicera med både energimätarna och Raspberry Pi behöver Arduinon ha minst två serieportar. De flesta Arduinoversioner har bara en, men Mega 2560 har fyra därför valdes denna.

Arduinoenheten ansluts till Raspberry Pi med RS-232 över USB för att möjliggöra överföring av data. Energimätarna ansluts i en bussnätverkskonfiguration över RS-485. Arduino kan dock inte prata med RS-485 direkt eftersom standarden har elektriska krav som Arduino inte kan uppfylla. Istället behöver en IC-krets, hanterbar av mikroprocessorn, som agerar brygga mellan RS-485 och spänningsnivåer användas. En LTC485 [49] är lämplig för ändamålet. Ett kopplingsschema för denna adapter visas i figur 13. Komponenterna monterades på ett experimentkort anpassat för Arduino,



Figur 13: *Kopplingsschema för adapter. Anslutningar för Arduino finns på vänstra sidan, medan energimätarna ansluts på högersidan. Kretsen strömförsörjs av Arduinon.*

en så kallad shield, som sedan anslöts till Arduinon. En bild på den färdiga kretsen, provisoriskt uppkopplad, kan ses i figur 14.



Figur 14: *Provisorisk anslutning av energimätarna till Arduino. LTC485 tillsammans med kringkomponenter på shield.*

3.3.2 Utvecklad programvara för datorerna

Program för Arduino skrivs i en variant av språket C. Koden som har utvecklats finns tillgänglig i Appendix B. I koden benämns energimätaren som läser av elproduktionen som energimätare 1, medan mätaren som läser av anläggningens konsumtion benämns energimätare 2. Programmet har skrivits och kompilerats med hjälp av Arduino IDE version 1.6.0 från arduino.cc.

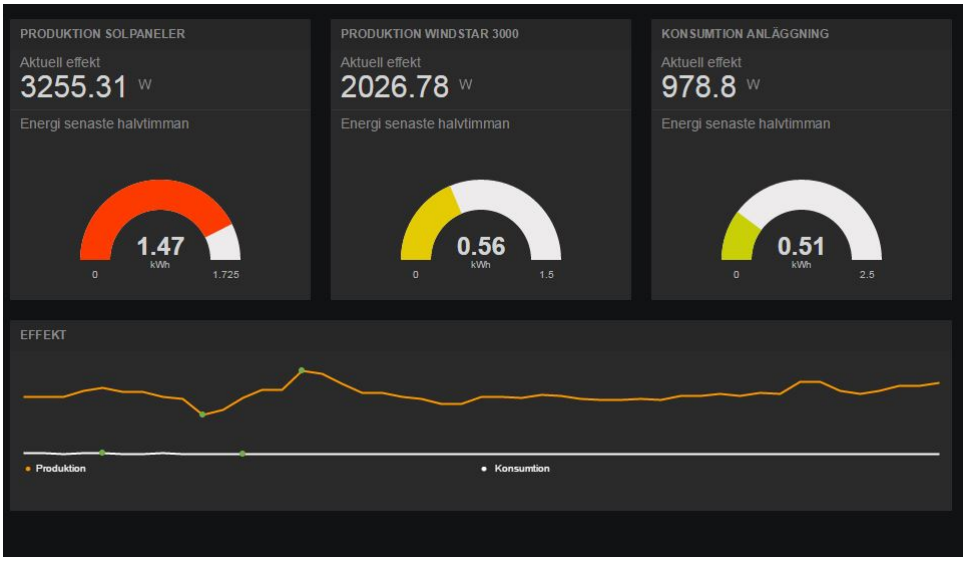
För att kommunicera med Modbus används en programmodul vid namn Simple Modbus Master. Modulen implementerar protokollet i form av en tillståndsmaskin, FSM. Den uppdateras kontinuerligt under drift och frågar hela tiden energimätarna efter ny information. Tillståndsmaskinen används genom att definiera paket innehållande all information som krävs för att kommunicera med energimätarna. Fyra paket är implementerade, anpassade för att hämta information om ackumulerad, exporterad och importerad energi från energimätarna samt momentana värden (spänningar, strömmar och effektvärden). Om någon energimätare slutar svara, exempelvis på grund av strömavbrott eller störningar, kommer FSM:en att försöka fem gånger till innan paketet inaktiveras. Om ett eller flera paket har inaktiverats kan de återaktiveras igen med kommando på serieporten.

Programmet som körs på Arduinon är anpassat för att seriellt kommunicera med ett Python-skript på en annan dator, i det här fallet Raspberry Pi. Skriptet tar hand om data från Arduinon och behandlar informationen. Skriptet är skrivet i Python 2.7 och är uppdelat i fyra olika filer: measureMain.py, momentary.py, accumulated.py samt webConnection.py. All programkod finns i Appendix C. Huvudprogrammet finns i measureMain och hjälpfunktioner finns i de övriga filerna. Skriptet skriver medelvärdesbildad information om spänningar, strömmar och aktiv effekt till textfiler en gång per minut. Likaså lagrar det värden för importerad respektive exporterad energi var tionde minut.

Var femte sekund laddas aktuella värden upp till en webbtjänst vid namn Carriots. Carriots är en molnplattform utformad för att ta emot och lagra stora mängder data från små, internetanslutna enheter. Datan kan sedan distribueras vidare till andra enheter och webbtjänster, eller sparas lokalt för vidare analys. I detta fall distribueras data vidare till en internetbaserad kontrollpanel, Freeboard. Här visualiseras realtidsinformation om anläggningens effekt- och energiproduktion samt motsvarande konsumtion. En skärmbild över Freeboard-panelen visas i figur 15 [50].

3.4 Design och implementering av optimeringskurvan

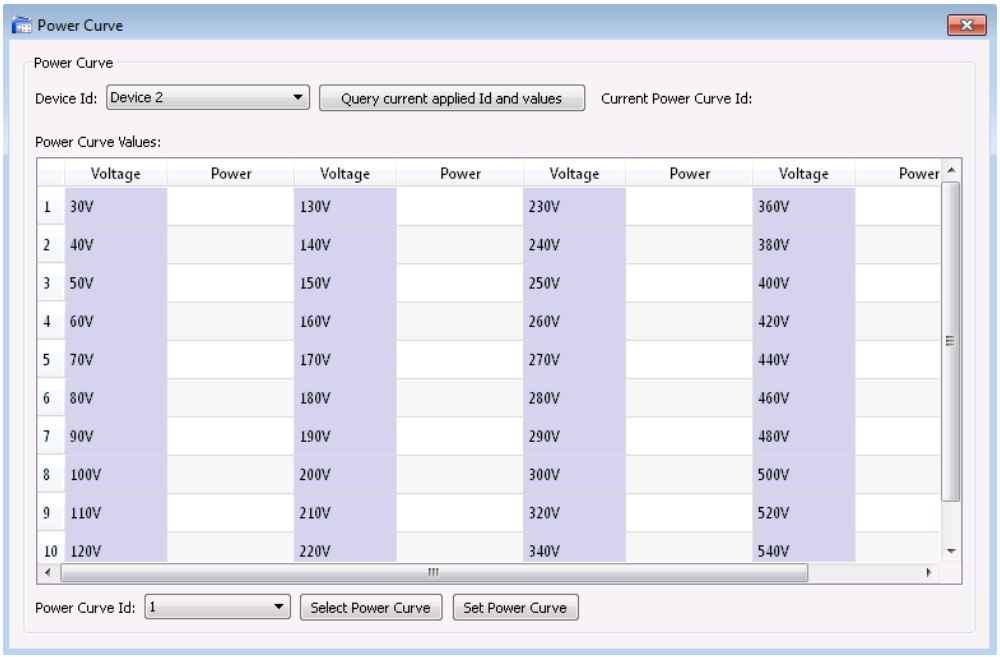
I databladet för Windstar 3000 finns en vind-effektkurva. Enligt återförsäljare är testerna för att uppnå kurvan gjorda i en vindtunnel [24]. Eftersom vind-effektkurvan är framtagen i laboratoriemiljö, försvinner troligen några procentenheter av verkningsgraden. Optimeringens mål var därför att försöka uppnå denna kurva och med den, även utlovade effekt. För att uppnå en optimal vind-effektkurva angreps problemet på två sätt; först genom tester på Hönö och därefter genom beräkningar på turbin, generator och effekt.



Figur 15: *Visualisering av data på Freeboard [50].*

3.4.1 Test av belastningskurvor på anläggningen

För att förenkla programmeringen av omriktaren användes programmet PowerMonitor. Det är möjligt att ändra inställningarna för belastningskurvan direkt i omriktaren, men PowerMonitor förenklar avsevärt inprogrammering av dessa. En dator kopplades upp via Bluetooth till omriktaren och därefter kunde belastningskurvan ändras till en ny, vilken visas i figur 16.

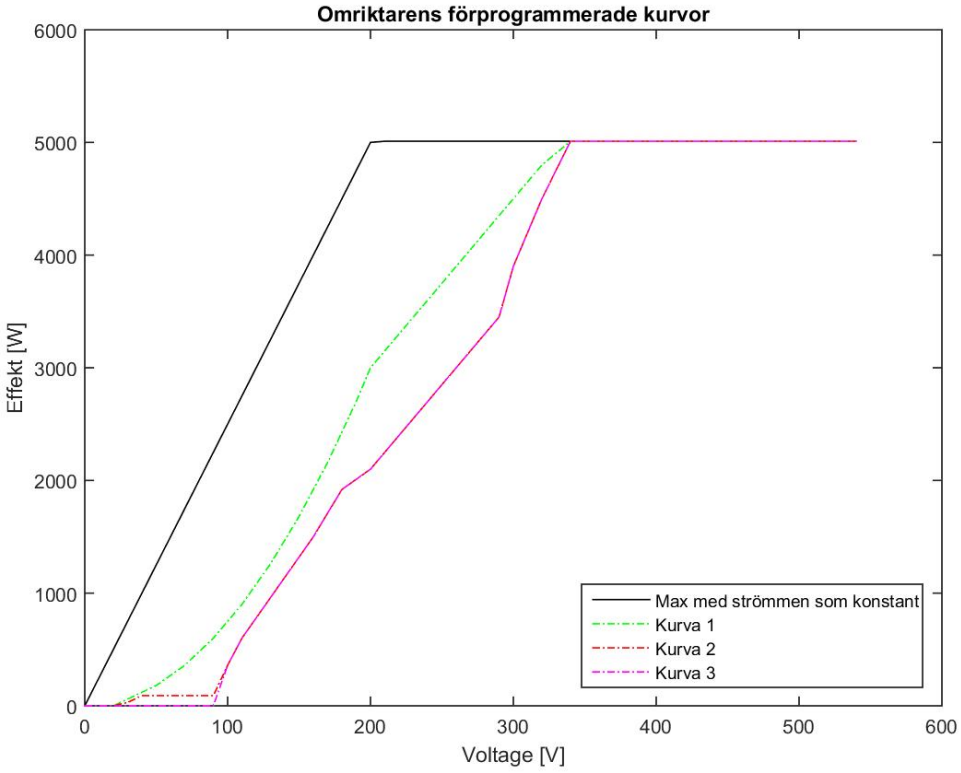


	Voltage	Power	Voltage	Power	Voltage	Power	Voltage	Power
1	30V		130V		230V		360V	
2	40V		140V		240V		380V	
3	50V		150V		250V		400V	
4	60V		160V		260V		420V	
5	70V		170V		270V		440V	
6	80V		180V		280V		460V	
7	90V		190V		290V		480V	
8	100V		200V		300V		500V	
9	110V		210V		320V		520V	
10	120V		220V		340V		540V	

Figur 16: *PowerMonitors interface.*

Vid projektets start låg tre kurvor inprogrammerade i omriktaren. Dessa kurvor antogs som utgångspunkt vid utformning av den nya. Som ses i figur 17 var i synnerhet kurva

två och tre lika varandra, medan kurva ett föreföll mer genomarbetad i optimeringssyfte. Då omriktaren var anpassad för 5000 W uppfattades det som att den branta ökningen kom för sent. Detta överensstämde även med tidigare mätningar gjorda på generatorn som visade att 3000 W kunde uppnås vid 125 V, vilket kan ses i Appendix A, figur 23.



Figur 17: *Omriktarens förprogrammerade belastningskurvor.*

Det valdes att starta med ett mellanting mellan kurva ett och två, vars lutning är svag i början men blir sedan brantare. Tanken var att nå 3000 W runt 130 - 140 V då detta kräver något mindre uttag av ström än mätningarna visat. Efter 3000 W fortsätter ökningen linjärt genom att följa maxtaket med avseende på omriktarens maximala ström 25 A. Den svagare initiala lutningen kommer sig av önskan att komma upp i märkvarvtal genom att lägga låg belastning vid svagare vindar.

Under den del av projektet som handlade om att testa fram en belastaningskurva besöktes Hönö ett antal tillfällen. Från dessa tillfällen kunde spänningsnivå vid vindar mellan 2 - 12 m/s mätas och därefter antogs en kurva. Denna reviderades med ganska små förändringar vid flertalet olika försök. Den som visade på mest uteffekt i förhållande till vindhastigheten går att beskåda i figur 20, i denna bild ses även de uträknade värdena, kod finns bifogat i Appendix J.

3.4.2 Numerisk analys av belastningskurva

Det har även genomförts en numerisk analys av belastningskurvan, genom att beräkna hur spänning och effekt beror av varandra. Ekvationerna 2 - 9 som behandlades i teoriavsnittet har använts för att illustrera grafer i MATLAB.

På anläggningen användes ett befintligt program i LabVIEW för att samla in aktuell mätdata av spänning, vindhastighet och varvtal. Relevant mätdata sammanställdes i MATLAB, se Appendix E för kod och Appendix F för grafer. De förenklade graferna, figur 28 och 31 i Appendix F, skapades med hjälp av en MATLAB-funktion som avrundar värden på y-axeln, kod finns i Appendix G [51].

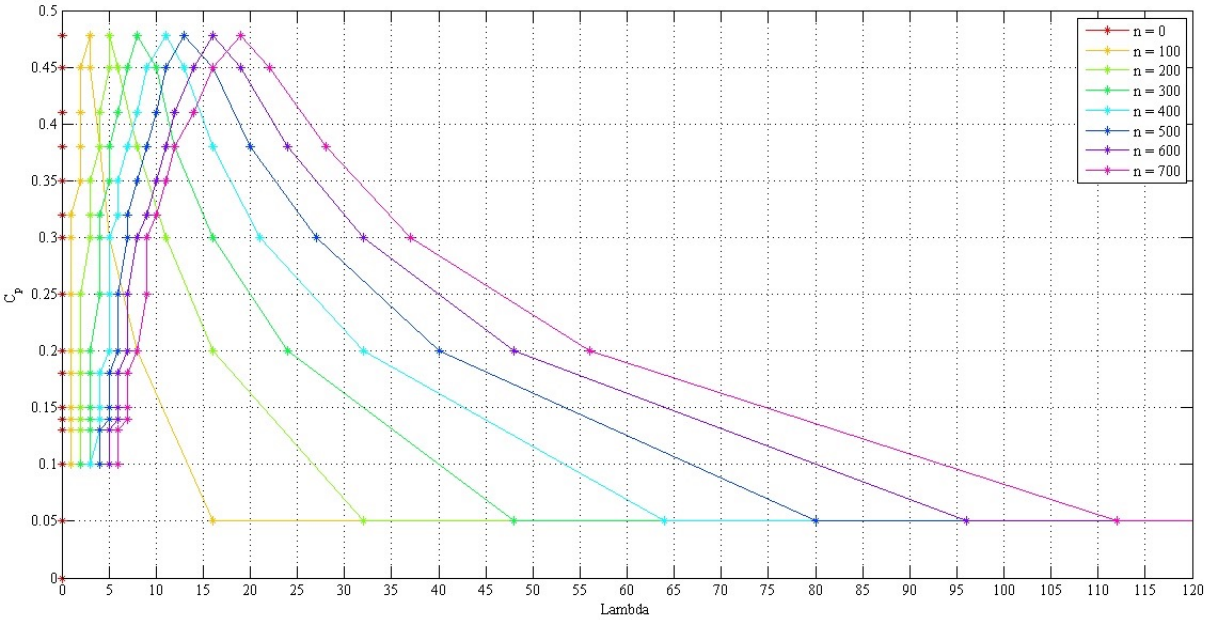
Enligt tillverkarens specifikationer varierar C_p-värdet för Windstar 3000 med olika vindhastigheter vilket går att se i tabell 1 [52].

Tabell 1: *Vindhastighet mot C_p på havsnivå*

Vindhastighet [m/s]	C_p
2.00	0.05
2.50	0.2
3.00	0.3
4.00	0.38
5.00	0.45
6.00	0.478
7.00	0.45
8.00	0.41
9.00	0.38
10.00	0.35
11.00	0.32
12.00	0.3
13.00	0.25
14.00	0.2
15.00	0.18
16.00	0.15
17.00	0.14
18.00	0.13
19.00	0.1
20.00	0.1

Löptalet, λ, räknades ut med hjälp av ekvation 8 där de givna vindhastigheterna användes. Specifikationen beskrev dock inte vilket varvtal som berörde korrelationen, därför var beräkningarna tvungna att baseras på flera olika varvtal. Varvtal från 0 till 700 rpm valdes, då märkvarvtal är 700 rpm. Beräkningarna genomfördes med 100 rpm i stegintervall och resultaten visas i figur 18, för kod se Appendix H.

Högsta möjliga C_p eftersträvas för att åstadkomma en så hög effekt som möjligt enligt ekvation 6. Det högsta C_p för det lägsta beräknade varvtalet valdes som utgångspunkt i analysen, vilket är $C_p = 0{,}478$ för varvtalet 100 rpm. Detta motsvarar ett löptal på $\lambda = 3$.



Figur 18: *Graf över C_p och λ vid olika varvtal.*

Ur ekvation 8 och 9 härleddes följande uttryck

$$v = \frac{V_{tip}}{\lambda} = \frac{\omega_m r}{\lambda} = \frac{2\pi n r}{60\lambda} \tag{15}$$

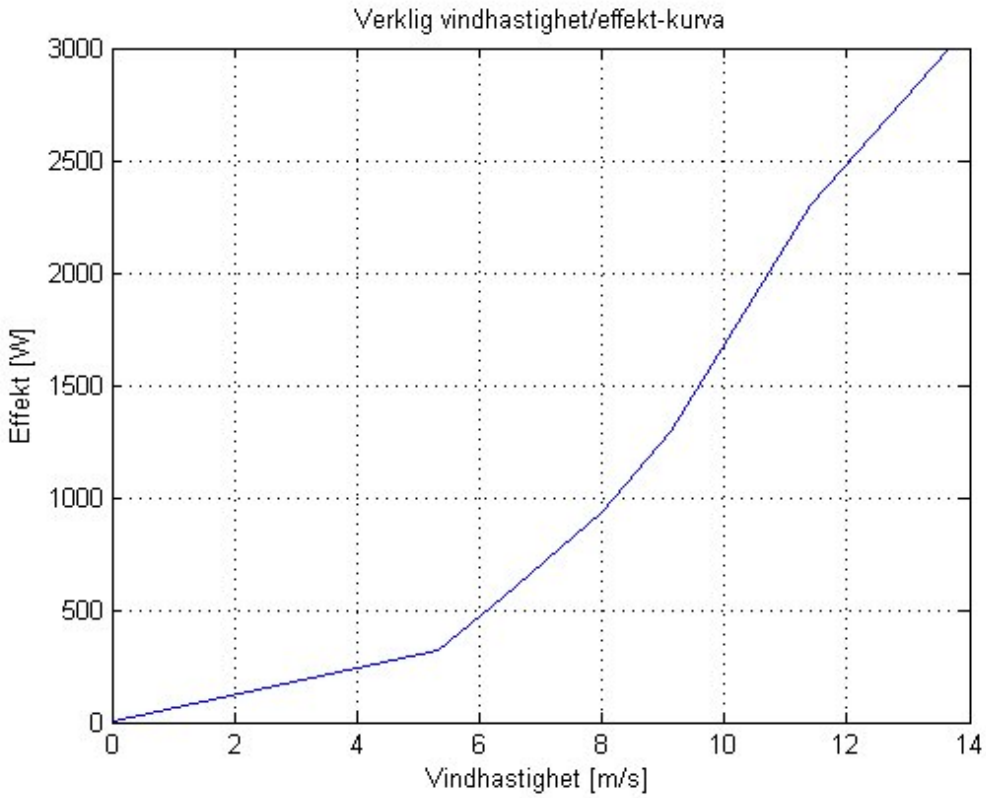
där löptalet, λ, sattes in i ekvation 15 och varvtalet, n, baserades på tidigare valt λ. En vindhastighet räknades fram genom ekvationen och jämfördes med motsvarande punkt för vindhastighet i figur 28, se Appendix F. Om varvtalet var lägre än tidigare givet varvtal med hänsyn till vindhastigheten kunde inte ovanstående givet C_p och λ användas utan ett lägre C_p och λ behövde antas och därefter upprepades proceduren med ekvation 15. Om varvtalet var högre än det givna med avseende på vindhastigheten var det möjligt att använda värdena då det alltid går att ta ut ett lägre varvtal.

Efter att vindhastigheten bestämts användes ekvation 6, med givet C_p, för att räkna ut den högsta teoretiska effekten Windstar 3000 kan producera. Figur 31 i Appendix F användes för att läsa av spänning i förhållande till vindhastighet och därmed kunde en spänning erhållas för den bestämda vindhastigheten. Utifrån detta erhölls både spänning och effekt för givet varvtal.

Proceduren repeterades för alla givna varvtal och därefter sammanställdes en spänning-effektkurva, tidigare nämnd belastningskurva. Då de tidigare mätningarna inte innehöll mätningar över 515 rpm och 128 V kunde inte värden över dessa läsas av, utan uppskattades utifrån grafernas lutning och utseende. Vid varvtalet 600 rpm och spänningen 130 V uppskattades vindhastigheten till 13,69 m/s. Kurvan behövde justeras då spänningarna inte var anpassade till omriktaren, det vill säga spänningarna inte hade ökningen på 10 V per effektvärde. Kurvan anpassades genom att avläsa ungefärliga värden för spänning i intervall om tio och sedan sammanställa samtliga effekter. Mätningar på spänningar över 130 V existerade inte, som tidigare nämnt.

Dessa anpassades med hjälp av effektformeln. Maxström 25 A antogs, då vindkraftverket troligen producerar denna ström vid högre spänningar. Strömmen multiplicerades sedan med var tionde spänningsvärde från 140 V till 200 V för att få fram effekten. Vid 200 V nås maxeffekten 5000 W och resterande spänningar över 200 V erhöll därför denna effekt. När kurvan var anpassad matades den sedan in i omriktaren via datorprogrammet PowerMonitor.

För att kunna jämföra tillverkarens vind-effektkurva behövde även en vind-effektkurva från de numeriska beräkningarna tas fram. Utifrån ekvation 15 kunde vindhastigheten räknas fram, där de tidigare uträknade och uppskattade varvtalen från 0 till 600 rpm användes med sina motsvarande λ-värden. Effektvärden erhölls av ekvation 6 nämnt i teoriavsnittet. Utifrån de framräknade parametrarna kunde en vind-effektkurva tas fram och visas i figur 19, för kod i MATLAB se Appendix I.



Figur 19: *Vind-effektkurva framtagen utifrån de numeriska beräkningarna.*

3.5 Energiutvärdering av anläggningen

Då vinden konstant ändrar riktning och hastighet behöver medelvärden räknas fram för att underlätta effektberäkning. Antalet soltimmar per månad varierar och statistiken från anläggningens solproduktion är begränsad. Ett allmänt samband baserat på energiproduktion per soltimme behöver därför tas fram.

3.5.1 Vinddata

Medelvärden för vindhastigheter sammanställdes månadsvis på Hönö mellan år 2009 och 2012 [9]. Vindhastigheterna mättes av flera olika typer av anemometrar[2] som finns installerade på anläggningen. Valet föll på en mekanisk anemometer då den hade flest antal mätvärden tillgängliga. Ett snitt på månadernas olika vindhastigheter beräknades för att endast behöva räkna på en vindhastighet per månad.

De olika medelvindhastigheternas effekter lästes av i vind-effektkurvan, se figur 19, och därefter plockades en medeleffekt ut för varje månad. Genom att dividera varje medeleffekt med 1000 och sedan multiplicera med antalet timmar per månad erhölls antalet kWh el producerad av vinden för samtliga månader under ett år.

3.5.2 Solcellernas energiproduktion på årsbasis

Enligt uppmätta värden från Steca Grid [11] producerade solcellerna 3488 kWh år 2014. Statistiken över antalet soltimmar hämtades från SMHI [10] och var i Göteborg 1835 stycken under år 2014. Detta motsvarar 1,9 kWh/soltimme. Medelvärdet av soltimmar för år 2002 - 2014 är 1759,3 vilket, baserat på tidigare uttryck, ger en ungefärlig energiproduktion från solcellerna på 3342,7 kWh per år.

För att få fram en graf över energiproduktion från solcellerna mot anläggningens energikonsumtion togs ett månadsvis medelvärde över antalet soltimmar för åren 2011 - 2014 från SMHIs hemsida [10]. Valet ansågs ha tillräcklig statistisk spridning för att kunna representera ett bra medeltal. Antalet soltimmar per månad multiplicerades sedan med 1,9 för att få fram produktionen. Resultatet presenteras i resultatdelen i figur 21.

3.5.3 Data för konsumtion och sammanställning

Konsumtionen avlästes från sammanställda dokument utifrån tidigare avläsningar av energimätarna [11]. Med anläggningens uträknade produktion utifrån sol- och vinddata samt anläggningens konsumtion skapades en gemensam graf, se figur 21. Kod till grafen finns bifogat i Appendix K. Det totala antalet kilowattimmar för produktion och konsumtion summerades individuellt. Därefter beräknades differensen för att undersöka huruvida anläggningen producerar mer än vad som årligen förbrukas.

[2] Anemometer är ett mätinstrument som bestämmer luftmassors rörelsehastigheter.

4 Resultat

Projektets resultat för de tre delmomenten; bromssystemet, insamling av data samt optimering av vindkraftverket redovisas i detta kapitel. Då alla moment var slutförda och redo för sluttest i verkligheten, visade det sig att vindkraftverket inte fungerade som det skulle, därav kunde ingen slutlig utvärdering av optimeringskurvan tillsammans med bromssystemet genomföras. Anledningen till varför Windstar 3000 slutade fungera är fortfarande okänd. Under projektets gång kunde en del mätvärden sammanställas, baserat på optimeringskurvor. Dessa mätvärden ligger till grund för projektets slutsatser.

4.1 Resultat av bromssystemets uppbyggnad och implementering

På grund av att Windstar 3000 inte fungerade som det skulle vid installation av bromssystemet kunde inga resultat eller mätvärden i verkliga förhållanden fås. Vindkraftverket kom inte upp i tillräckligt högt varvtal för att systemet skulle börja bromsa, trots att vinden var tillräckligt stark. Därav kunde ingen verifiering av bromssystemets pålitlighet göras på plats. Däremot fungerade systemet felfritt i laboratorium för de spänningnar och strömmar det testades med. Dessa spänningar och strömmar matades från ett likspänningsaggregat, och referensvärden reglerades för att få systemet att agera vid lägre strömmar än vad det är avsedd för. Vid detta tillfälle fungerade systemet helt i enlighet med vad som förväntades.

4.2 Insamling av mätdata

Ett program integrerat i Raspberry Pi och Arduino Mega skapades och samlar in mätdata för följande parametrar; exporterad och importerad effekt, spänningar, ström, producerad effekt och total konsumerad energi. Programmet fungerar som det ska och skriver ut till textfiler som används till datasammanställning i MATLAB. Koden för insamlingsprogrammen finns bifogat i Appendix B & C. Anläggningens aktuella produktion och konsumtion finns illustrerad och tillgänglig på hemsidan Freeboard [50].

4.3 Optimering

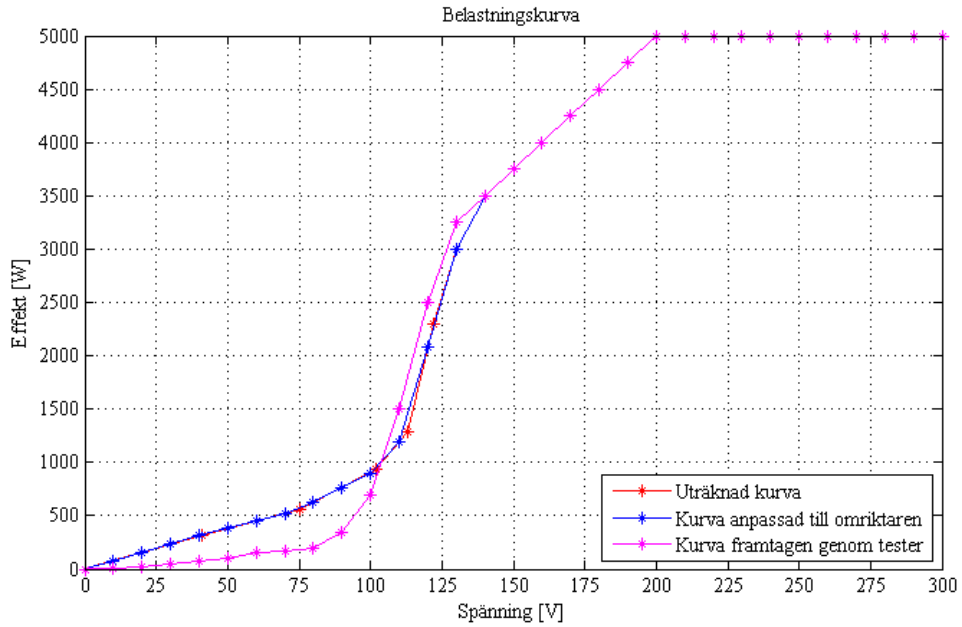
Manuella tester av belastningskurvor ute på Hönö gav mer förståelse gällande omriktaren och i samband med LabVIEW-mätningarna kunde även resultatet visualiseras. Genom ett flertal tester visade sig en kurva mest gynnsam och denna visas tillsammans med den framräknade kurvan i figur 20.

Som redovisades i metodavsnittet erhölls C_p-λ-kurvor från den numeriska beräkningen. Genom avläsning i kurvorna och beräkningar därifrån kunde följande värden sammanställas i tabell 2.

Tabell 2: *Mätdata sammanställd från den numeriska beräkningen.*

n [RPM]	C_p	λ	v [m/s]	U [V]	P [W]
100	0,478	3	5,32	41	321,8
200	0,478	5	6,39	75	557,1
300	0,41	6	7,98	102,1	931,58
400	0,38	7	9,13	113,05	1291,2
500	0,32	7	11,41	121,9	2300
600	0,25	7	13,9 (antaget)	130	3000

För att kunna programmera in kurvan i omriktaren anpassades värdena från tidigare uträkningar. De nya värdena valdes som den slutgiltiga kurvan och visas tillsammans med den ursprungliga kurvan och den framtagna kurvan visas i figur 20.

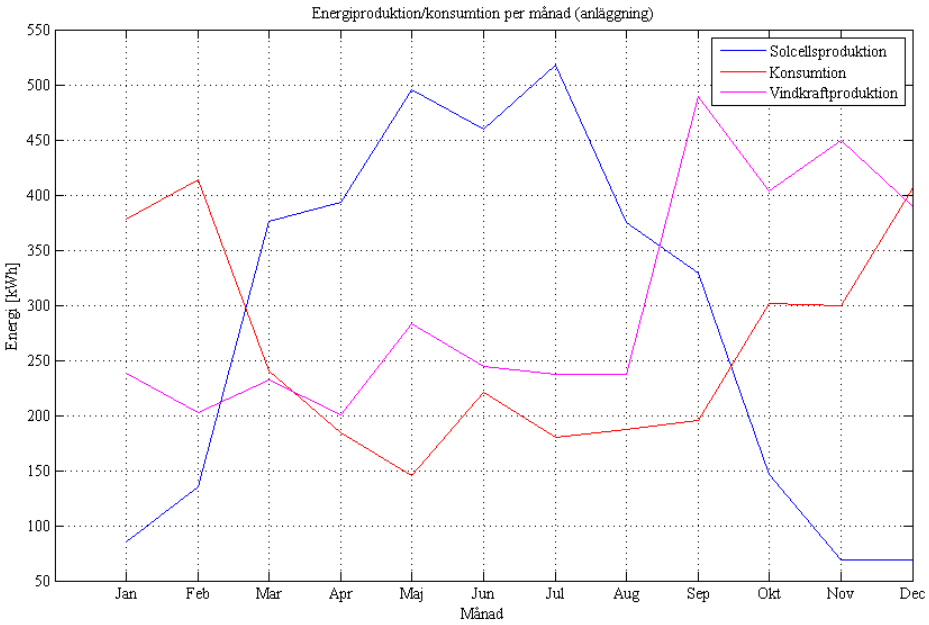


Figur 20: *Belastningskurvor framplockade på olika sätt.*

4.4 Resultat av sammanställning och utvärdering av anläggningens produktion och konsumtion

Utifrån uträkningar av energiproduktionen för både Windstar 3000 och solcellspanelerna samt konsumtionen på anläggningen skapades en graf som visas i figur 21. Som kan ses i figuren är att energiprodukitonen från solcellerna är mycket högre under sommarhalvåret och energiproduktionen från vindkraftverket är högre under den senare delen av året. Energikonsumtionen ökar med kallare temperaturer, alltså under vinterhalvåret.

En summering och uträkning av differensen för produktionen och konsumtionen visade på ett överskott av energi. Differensen blev 3900 kWh/år. Vindkraftsproduktionen



Figur 21: *Produktion och konsumtion av energi på anläggningen Hönö under ett år.*

är dock teoretiskt framräknad genom genomsnittliga vindhastigheter och den effekt vindkraftverket skulle kunna tänkas generera. Då vindkraftverket i dagsläget inte fungerar är vindkraftsproduktionen noll. Detta innebär att produktionen bara överskrider konsumtionen med ungefär 298 kWh/år.

5 Diskussion och analys

Windstar 3000 är dessvärre inte optimalt placerad då bebyggelsen i söder skapar mycket turbulenta vindar vilket gör att verket ofta roterar runt sin vertikalaxel. Detta medför försämrad energiproduktion och ger även med stor sannolikhet ett försämrat mätresultat vilket påverkar slutresultatet. En åtgärd till att förbättra resultatet skulle exempelvis vara att placera vindkraftverket på en högre altitud eller i en friare omgivning för att undvika de oregelbundna sydvindarna.

Som tidigare nämnts är $\lambda_{opt} = 5$, detta tolkades vid optimeringens start som en konstant men visade sig variera med vindhastighet. Om löptalet alltid hade varit satt till fem, hade det dels krävts höga vindar för att komma upp i märkvarvtal och dels hade både Windstar 3000 och alla andra vindkraftverk haft okontrollerat varvtal vid riktigt höga vindhastigheter. Vid den typen av vindar måste löptalet sänkas för att undvika okontrollerad uppvarvning av turbinen.

Vid projektets start tillhandahölls två olika datablad för Windstar 3000, en från återförsäljare Windforce och en från tillverkare HYE. Vind-effektkurvan från återförsäljaren Windforce är tvivelaktig och gav värden på 4000 W vid 11 m/s. Enligt ekvation 5 skulle detta innebära ett C_p på 67,2 procent. Som tidigare visats ligger en begränsning av Betz lag på 59,3 procent och detta gör att siffrorna från Windforce datablad inte är korrekta. Windforce hemsida besöktes för att försöka förstå hur detta är möjligt och där uppvisades en helt annan kurva, identisk med den som de kinesiska tillverkarna HYE uppvisade.

Det anades att Windforce hade fuskat i sin iver att sälja mer och använt sig av ett försäljningsknep, att överdriva vindkraftverkets högsta möjliga produktion. Det måste dock ses som ett stort misstag då begränsningen av Betz lag överskrids. Det är något som torde göra en påläst köpare skeptisk och kanske få denne att välja en annan modell och/eller återförsäljare.

En konversation med Windforce har förts och enligt dem beror vindkraftverkets effektivitet på belastningskurvan. Windforce menade att de har den belastningskurva som ska medföra optimal energiproduktion. Då kurvan granskades upplevdes den tveksam. Den kom inte upp i 3000 W förrän vid 300 V, en spänning Windstar 3000 under mätningar aldrig har lyckats generera. Av denna anledning valdes belastningskurvan snabbt bort.

Innan bromssystemet installerades fanns det tillfälle att köra vindkraftverket vid stark vind. För att det skulle vara möjligt att stanna verket utan att förstöra generatorns permanentmagneter användes en elektrisk last. Denna kunde verka på ungefär samma sätt som ett bromssystem. På lasten kunde maximal spänning och ström ställas in. Då Windstar 3000 började varva upp och spänningen ökade begränsades det till den spänning som ställts in på lasten. Detta gjorde att kontrollen över verkets rotationshastighet behölls. Om strömuttaget ökade samtidigt som spänningen sänktes bromsades verket ner.

Den beräknade produktionen från både Windstar 3000 och solcellspanelerna är ungefärlig och utgår från medelvärden. Därför kommer utfallet från år till år variera i enlighet med hur mycket väderförhållanden avviker från medelvärdet.

Även om solcellspanelerna ensamma kan tillgodose anläggningen med mer energi än vad som konsumeras sett till årsbasis, behöver man ha i åtanke att 298 kWh mer per år inte ger mycket marginal. Om det blir en väldigt kall vinter eller en regnig sommar behöver anläggningen naturligtvis konsumera mer energi. Därför bör båda energiproducenter vara i drift för att ha en större säkerhetsmarginal. Energikonsumtionen kan även reduceras, till exempel om bodarna isoleras bättre eftersom det läcker ut en del värme genom väggarna.

Ytterligare svårigheter som uppenbarade sig under arbetet med optimeringen involverade omriktaren. En sådan svårighet var att vid 50 V programmerades effekten till 380 W med hjälp av PowerMonitor. Omriktaren ändrade dock värdet till 360 W. Försök gjordes att manuellt programmera omriktaren, men inte heller då sparades värdet. Detta är dock såpass små förändringar att det i verkligheten troligtvis inte spelar någon roll.

Något som försvårar möjligheten att dra slutsatser kring huruvida det designade bromssystemet fungerar eller ej, är det faktum att verket är trasigt. Under laborationstester fungerade systemet perfekt och belastningskurvor har testas med vindkraftverket igång. Dock borde något ha hänt med vindkraftverket under tiden mellan det senaste testet av belastningskurvan och montering av bromssystemet. Vad som kan ha skett är oklart. Under felsökningen har strömmarnas karakteristik undersökts och inga avvikelser har hittats. Nedtagning av verket och kontroll av rotorbladens montering har också gjorts utan att några brister upptäcktes.

Vad som skulle kunna hänt är att vinden fått tag i rotorbladen trots att vindkraftverket varit parkerat. Om detta är fallet finns risk att generatorns permanentmagneter har blivit skadade och Windstar 3000 förmodligen inte kommer kunna generera någon el. Detta eftersom permanentmagneterna gör elproduktion möjlig. Om så är fallet krävs att ett nytt vindkraftverk installeras och belastningskurvan borde därefter på nytt anpassas. Skulle ett vindkraftverk i samma effektnivå som Windstar 3000 installeras borde samma bromssystem kunna vara kvar och kopplas in till det nya, med viss anpassning.

Installationen av bromssystemet och därigenom upptäckten av vindkraftverkets haveri noterades sent, då design och uppbyggnad av systemet tog längre tid än planerat. Detta på grund av en lång inläsningsperiod i ämnet och felaktiga arbetsmetoder i början av projektet. Huruvida en tidig installation av bromssystemet skulle förhindrat vindkraftverket från haveri är oklart då orsaken till detta inte kunnat fastställas.

Som småskalig energiproducent skulle dock den privata aktören rekommenderas att investera i solceller framför mindre vindkraftverk, då dessa är betydligt enklare att använda och saknar mekaniska delar som riskerar att gå sönder.

6 Slutsatser

Om Windstar 3000 tillsammans med solcellerna hade fungerat och levererat uteffekt utifrån teoretiska beräkningar hade det årliga överskottet statistiskt varit ungefär 3900 kWh/år. Detta kan anses vara en god marginal, men då Windstar 3000 inte fungerar blir nettoöverskottet endast 300 kWh. Marginalen skulle kunna överskridas under ett år med exempelvis ett lågt antal soltimmar eller om konsumtionen ökar.

Referenser

[1] Naturvårdsverket. (2014). Fossila bränslen. [Online], URL: http://www.naturvardsverket.se/Miljoarbete-i-samhallet/Miljoarbete-i-Sverige/Uppdelat-efter-omrade/Energi/Fossila-branslen/ (hämtad 2015-02-09).

[2] ——, (11 mars 2015). Nationella utsläpp och upptag av växthusgaser. [Online], URL: http://www.naturvardsverket.se/klimat2013 (hämtad 2015-04-20).

[3] Europa Kommissionen. (2012). Energy Roadmap 2050. [Online], URL: https://www.energy.eu/publications/Energy-Roadmap-2050.pdf (hämtad 2015-02-05).

[4] Regeringskansliet. (2 jan. 2015). Mål och visioner. [Online], URL: http://www.regeringen.se/sb/d/2448/a/252160 (hämtad 2015-02-06).

[5] Chalmers Tekniska Högskola. (3 nov. 2014). Chalmers för en hållbar framtid. [Online], URL: http://www.chalmers.se/sv/om-chalmers/miljo-och-hallbar-utveckling/Sidor/default.aspx (hämtad 2015-02-06).

[6] ——, (5 sept. 2013). Large-scale Renewable Electricity Generation and Grid Integration. [Online], URL: http://www.chalmers.se/en/areas-of-advance/energy/cei/Pages/Large-scale-renewable-electricity-generation-and-grid-integration.aspx (hämtad 2015-02-05).

[7] ——, (25 mars 2015). Avdelningen för elteknik. [Online], URL: http://www.chalmers.se/sv/institutioner/ee/organisation/elteknik/Sidor/default.aspx (hämtad 2015-04-05).

[8] ——, (14 jan. 2015). Forskningsprojekt. [Online], URL: http://www.chalmers.se/sv/institutioner/ee/forskning/Sidor/forskningsprojekt.aspx (hämtad 2015-04-05).

[9] M. Koushik, "Hönö Wind Resource Assessment", Chalmers University of Technology, Göteborg, Sverige, tekn. rapport, 2014.

[10] SMHI Sveriges meteorologiska och hydrologiska institut. Års- och månadsstatistik. [Online], URL: http://www.smhi.se/klimatdata/meteorologi/2.1240 (hämtad 2015-05-11).

[11] M. Ellsén, "Energimätare Hönö vindkraftstation", Göteborg, Sverige, 2014, Unpublished.

[12] Guangzhou HY Energy Technology Co., LTD, *"HY ENERGY"*, Guangzhou, China, 2013.

[13] T. Burton, N. Jenkins, D. Sharpe och E. Bossanyi, *Wind Energy Handbook*, 2nd ed. Cambridge, United Kingdom och New York, NY, USA: John Wiley och sons, 2011.

[14] D. Levitan. (24 sept. 2012). High-Altitude Wind Energy: Huge Potential - And Hurdles. [Online], URL: http://e360.yale.edu/feature/high_altitude_wind_energy_huge_potential_and_hurdles/2576/ (hämtad 2015-04-05).

[15] D. J. MacKay, *Sustainable Energy - Without the Hot Air*. Cambridge, England: UIT Cambridge Ltd, 2012.

[16] W. D. Lubitz, "Impact of ambient turbulence on performance of a small wind turbine", *Renewable Energy*, vol. 61, s. 69–73, 2011. URL: http://www.sciencedirect.com/science/article/pii/S0960148112004855.

[17] T. Wizelius, *Vindkraft i teori och praktik*, 3. utg. Lund, Sverige: Studentlitteratur, 2015.

[18] M. R. Patel, *Wind and Solar Power Systems - Design, Analysis, and Operation*, 2nd ed. CRC Press Taylor & Francis Group, 2006.

[19] M. H. Ali, *Wind Energy Systems: Solutions for Power Quality and Stabilization*. CRC Press Taylor & Francis Group, 2012.

[20] The University of British Columbia. (18 maj 2010). Wind Tubines - Betz Law Explained. [Online], URL: http://c21.phas.ubc.ca/article/wind-turbines-betz-law-explained (hämtad 2015-04-28).

[21] M. Ragheb och A. M. Ragheb, "Wind turbines theory - the betz equation and optimal rotor tip speed ratio", University of Illinois, Urbana-Champaign, Illinois, USA, tekn. rapport, 5 juli 2011.

[22] Svensk Vindenergi. (23 aug. 2012). Lathund – olika begrepp när du talar om vindkraft. [Online], URL: http://www.vindkraftsbranschen.se/start/vindkraft/lathund-2/ (hämtad 2015-05-07).

[23] Västra Götalandsregionen. (26 febr. 2015). Energi och teknik. [Online], URL: http://www.powervast.se/sv/Ovriga-sidor/Power-Vast/Power-Vast/Om-vindkraft/Energi-och-teknik/ (hämtad 2015-05-07).

[24] Windforce. Windstar 3000. [Online], URL: http://www.windforce.se/vindkraft-windstar3000.php (hämtad 2015-05-09).

[25] L. Ningbo Ginlong Technologies Co., *''Wind Grid Tie Inverter"*, China, 2012.

[26] M. Behrendt, C. Ekman, M. Lundgren, A. Rodionov och F. Rolff, "Projektering, installation och utvärdering av en liten hybridanläggning bestående av solceller och vindkraftverk", Chalmers Tekniska Högskola, Göteborg, Sverige, tekn. rapport, 2013.

[27] A. Einarsson, D. Jansson, H. Barsk, M. F. Eriksson och P. Wadström, "Evaluation and optimization of a small 3 kw wind turbine system", Göteborg, Sverige, 2013.

[28] Stecagrid Elelktronik, *''Installations- und Bedienungsanleitung"*, Memmingen, Tyskland.

[29] A. Nositschka, *''Sun-Light Harvesting with Surface Patterned Glass for Photovoltaics"*, [Online]. URL: http://www.lehigh.edu/imi/teched/SolarWS/T3f_Nositschka.pdf (hämtad 2015-05-18).

[30] Leonics. Basics of MPPT Solar Charge Controller. [Online], URL: http://www.leonics.com/support/article2_14j/articles2_14j_en.php (hämtad 2015-05-09).

[31] Simply Modbus. Frequently Asked Questions. [Online], URL: http://www.simplymodbus.ca/faq.htm (hämtad 2015-04-09).

[32] L. Biez. (april 2015). Introduction to RS485. [Online], URL: http://www.lammertbies.nl/comm/info/RS-485.html (hämtad 2015-05-17).

[33] B & B Electronics. Basics of the RS-485 Standard. [Online], URL: http://www.bb-elec.com/Learning-Center/All-White-Papers/Serial/Basics-of-the-RS-485-Standard.aspx (hämtad 2015-04-09).

[34] How Stuff Works? Mikrokontroller. [Online], URL: http://electronics.howstuffworks.com/mikrokontroller1.htm (hämtad 2015-04-09).

[35] Raspberry Pi. What is Raspberry Pi? [Online], URL: https://www.raspberrypi.org/help/what-is-a-raspberry-pi/ (hämtad 2015-04-09).

[36] Arduino. What is Arduino? [Online], URL: http://www.arduino.cc/en/Guide/Introduction (hämtad 2015-03-24).

[37] ——, Compare board specs. [Online], URL: http://www.arduino.cc/en/Products.Compare (hämtad 2015-05-18).

[38] TE Connectivity, *''High Power Wire Wound Resistor"*, [Online], aug. 2011. URL: http://docs-europe.electrocomponents.com/webdocs/1360/0900766b813600ff.pdf (hämtad 2015-05-17).

[39] Genesic Semiconductor, *''Silicon Standard Recovery Diod"*, [Online]. URL: http://www.farnell.com/datasheets/1911900.pdf (hämtad 2015-05-17).

[40] Powerex, *''Data Sheet CM100DY-24A"*, [Online]. URL: http://www.pwrx.com/pwrx/docs/cm100dy_24a.pdf (hämtad 2015-04-19).

[41] LEM, *''LEM LAH 25-NP CURRENT TRANSDUCER, 25A, PCB"*, [Online]. URL: http://se.farnell.com/lem/lah-25-np/current-transducer-25a-pcb/dp/2146849 (hämtad 2015-04-19).

[42] Fairchild Semiconductor, *''LM111-N/LM211-N/LM311-N Voltage Comparator"*, [Online], mars 2013. URL: http://www.farnell.com/datasheets/1716431.pdf (hämtad 2015-05-17).

[43] Texas Instruments, *''CD40107B Types"*, [Online], okt. 2013. URL: http://www.farnell.com/datasheets/1847419.pdf (hämtad 2015-04-19).

[44] Fairchild, *''FOD3184 3A Output Current, High Speed MOSFET/IGBT Gate Driver Optocoupler"*, [Online], febr. 2011. URL: http://www.farnell.com/datasheets/1817594.pdf (hämtad 2015-04-19).

[45] XP power, *''40 Watts, JCK Series"*, [Online], 21 jan. 2011. URL: http://www.farnell.com/datasheets/1903699.pdf (hämtad 2015-04-19).

[46] LEM, *''Voltage Transducer LV 25-P"*, [Online], 12 aug. 2014. URL: hhttp://www.farnell.com/datasheets/1866272.pdf (hämtad 2015-04-19).

[47] M. Ellsén et.al., "Vindkraftverk på fartyg", Unpublished.

[48] Semikron, *''SKM150GAL12T4"*, [Online], 3 sept. 2013. URL: http://www.semikron.com/dl/service-support/downloads/download/semikron-datasheet-skm150gal12t4-22892300 (hämtad 2015-04-19).

[49] Linear Technology, *''Low Power RS485 Interface Transciever"*, [Online]. URL: http://cds.linear.com/docs/en/datasheet/485fk.pdf (hämtad 2015-05-17).

[50] C. A. Nilsson, M. Alnervik, M. Nisshagen, M. Peshkova, C. Rådahl och J. Winther. Testanläggning Hönö. [Online], URL: https://freeboard.io/board/aMh89M (hämtad 2015-04-23).

[51] M. Ellsén. (2004). MATLAB bin.m. version R2015a.

[52] M. Eriksson. (8 okt. 2013). Re: Regarding HY 3000L - request for additional information. Personligt e-postmeddelande.

Appendix

A Dokumentation över tidigare mätningar utförda i laboratoriemiljö.

Mätningar på WindStar 3000 från WindForce.							
Uppkopplad mot DC-maskin i Maskinlabbet på Chalmers Elteknik.							
Mätningar av Magnus Ellsén 2013-10-25 t.o.m. 2013-10-30.							
Vilken ström ger max moment?							
	n	Idc	Udc	T			
	1/min	A	V	Nm			
	650	24,27	118,2	577			
	650	20,05	138	508			
		24,26	118	577			
		25		584			
		25,57		590			
		26,17		598			
		26,7		604			
		26,95		602			
		27,5		609			
		28,3		608			
		28,81		611			
		33,3		582			
		32,8		599			
		32,3		600			
		32		604			
		31,69		604			
10/28/13							
kl.	n	Idc	Udc	Udiod	T	R	Gtemp
	1/min	A	V	V	Nm	ohm	gr C
	650	32,7	63,2	65,9	590	1,93	58
	650	37,8	1,5	4,7	412	0	48
	600	37,6	1,5	4,7	428	0	33
	550	37,7	1,5	4,76	460	0	44
	500	36,5	1,5	4,7	515	0	
	450	35,5	1,5	4,7	530	0	
	400	34,4			575		
	350	32,8			602		59
	300	30,7			620		
	250	28,1			625		63
	200	24,6			601		
	150	20,2			535		
	100	14,4			416		63
	50	6,5			202		
	25	2,2			79		
	700	37,9			389		
	750	38,5			378		
	800	38,4			362		

Figur 22: *Resultat av mätningar på moment och strömmar vid olika varvtal då generatorn har varit kopplat till en elektronisk last.*

10/28/13							
		På elektronisk last					
kl.	n	Idc	Udc	T	Gtemp	Pdc	Pmek
	1/min	A	V	Nm	gr C	W	W
13:30	350	5	98,6	175		493	
	350	10	82,5	309		825	
	350	15	71,5	430		1072,5	
	350	20	58,2	538		1164	
	350	22	51,7	572		1137,4	
	350	24	44,5	601		1068	
	350	26	36,9	627		959,4	
	350	28	28	638	46	784	
	350	30	19,3	640		579	
	350	32	7,2	625	54	230,4	
	650	32	63,2	613		2022,4	
	650	30	77,5	625		2325	
	650	28	90,1	625	61	2522,8	
	650	26	102,4	613		2662,4	
	650	24	112,9	592	64	2709,6	
	650	22	122,7	564		2699,4	
	650	20	131,3	530		2626	
	650	18	139,4	494	66	2509,2	
	650	15	150	428		2250	
	650	10	169,4	313		1694	
	650	5	190,5	182	66	952,5	
	700	5	206,4	202	61	1032	
	700	10	184,7	332		1847	
	700	15	165,1	448		2476,5	
	700	18	153,9	510		2770,2	
	700	20	145,2	550	59	2904	
	700	22	135,8	582		2987,6	
	700	24	125	612		3000	
	700	26	112,6	629	62	2927,6	
	700	28	99	640		2772	
	700	30	83,6	640		2508	
	700	32	65,4	623	67	2092,8	
	700	22	146,5	546	30	3223	
	750	22	158,9	545	33	3495,8	
	750	20	167,1	510	36	3342	
	750	18	174,7	470		3144,6	
	750	16	181,7	423		2907,2	
	750	15	185,2	403	38	2778	
	750	10	205,6	283		2056	
	750	5	227,5	150		1137,5	
	500	5	147,9	150	40	739,5	
	500	10	130,8	283		1308	
	500	15	114,9	405		1723,5	
	500	16	111,9	429		1790,4	
	500	18	105,6	473	40	1900,8	
	500	20	99	514		1980	
	500	22	91,4	551		2010,8	
	500	24	83,1	580	44	1994,4	
	500	26	74,2	606		1929,2	
	500	28	63,5	620		1778	
	500	30	51,6	624	48	1548	

Figur 23: *Resultat på spänningar då turbinen har varierats i varvtal och strömvärden.*

10/30/13					
Mätning av tomgångsspänning					
Gen. Temp = 24 gr C					
Momentkalibrering: Nollmomentvisning: -0009					
10 kg * 0,5 m: -0488					
		Fluke 39			
	n	U1	U2	T	Gtemp
	1/min	V	V	Nm	gr C
	0	0,2	0,4	−7	24
	99	24,9	24,9	−4	
	200	50,6	50,4	−3	
	300	75,7	75,7	−2	
	400	101,2	101,2	−1	
	500	126,3	126,3	−1	
	600	152	151	1	
	650	164	164	2	
	700	177	177	3	
	750	190	183	4	
	0	0,2	0,4	−11	27

Figur 24: *Resultat av gerneratorns tomgångsspänning vid varierande varvtal.*

B Mätdatasystem: Arduinokod

modbusInterface.ino

```
        #include <SimpleModbusMaster.h>
//Finns att hämta via länken https://code.google.com/p/simple-modbus/
//här finns även utförlig dokumentation över hur modulen fungerar

//Deklarerar konstanter. #define gör samma jobb som const
#define baud 9600
#define timeout 1000
#define polling 200 //väntetid mellan varje fråga så slavarna hinner bli idle
#define retry_count 5
#define TxEnablePin 9 //pinne kopplad till LTC485
#define TOTAL_NO_OF_REGISTERS 58 //antal register för data från slavenhet,
//1 register = 16 bit

enum
{
  pEnergy1,
  pPower1,
  pEnergy2,
  pPower2,
  TOTAL_NO_OF_PACKETS //antal paket, uppdateras automatiskt
};

Packet packets[TOTAL_NO_OF_PACKETS];
//array av paket som ska skapas (ett paket innehåller en modbus-request)

unsigned int regs[TOTAL_NO_OF_REGISTERS]; //Skapar plats i minnet för data från
//slav (int är en 16-bitars datatyp)
char charReceived;

void setup() {
  //kod som körs en gång:
  Serial.begin(9600); //Initiera seriell kommunikation över USB
  memset(regs, 0, sizeof(regs)); //ser till att hela dataregistret är satt till
  //noll

  //Arrayn av paket, slav-ID, funktion, adress (aktiv export L1, 546C hex = 21612
  //dec), antal register som ska läsas, index för första datalagring i arrayen
  //regs
  modbus_construct(&packets[pEnergy1], 1, READ_HOLDING_REGISTERS, 0x5460, 24, 0);
  //En konstruktor per paket
  modbus_construct(&packets[pEnergy2], 2, READ_HOLDING_REGISTERS, 0x5000, 4, 24);
  //Läs total energi från energimätare 2
  modbus_construct(&packets[pPower1], 1, READ_HOLDING_REGISTERS, 0x5B00, 28, 28);
  //Läs momentana spänningar, strömmar, effekt
  modbus_construct(&packets[pPower2], 2, READ_HOLDING_REGISTERS, 0x5B14, 2, 56);
  //Momentan effekt energimätare 2

  //Port (fysiska pinnar är 18 o 19), hastighet, format för seriella kopplingen -
  //här 8 bitar utan paritet samt två stoppbitar, timeout, delay, antal försök,
  //enablePin, vår array av paket, antal paket, array där data sparas i Arduino
  modbus_configure(&Serial1, baud, SERIAL_8N2, timeout, polling, retry_count,
  TxEnablePin, packets, TOTAL_NO_OF_PACKETS, regs);
}

void loop() {
  // kod som körs kontinuerligt:

  modbus_update();
  //update ska köras minst en gång varje varv, får inte köras under if-sats e.
  //dyl. RPi ges alltid senaste info från regs, säger INTE till arduino när update
  //ska köras.

  if (Serial.available()){ //Om RPi frågar efter data
    charReceived = Serial.read(); //läs innehållet för att rensa cachen
    if (charReceived == '1'){ //Vi vill mata ut energistatus till RPi
      serialResponse(0, 28); //Läs av o returnera de 28 första registerna
```

```
67      }
68      else if (charReceived == '2'){
69        serialResponse(28, 58); //Momentana värden
70      }
71      else if (charReceived == '3'){
72        status();
73      }
74      else if (charReceived == '4')
75      {
76        setConnection();
77      }
78
79    }
80  }
81  void serialResponse(int intBegin, int intEnd){
82    String strTemp;
83    for (int i = intBegin; i < intEnd; i++)
84    { //Läs av de önskade registerna
85      strTemp = String(regs[i], HEX); //Konvertera till sträng
86      if (strTemp.length() < 4){ //Lägg på nollor - nollorna i MSB på hex
87      //försvinner
88      //i konverteringen. Nödvändigt eftersom 16-bitars registerna sätts ihop
89      //sekventiellt på RPi
90        for (int j = strTemp.length(); j < 4; j++)
91        {
92          strTemp = 0 + strTemp;
93        }
94      }
95      Serial.println(strTemp); //Skriv ut ett register i taget till RPi
96    }
97  }
98  void status(){
99    //Skriver ut hälsostatus för alla paket. Har någon energimätare slutat svara
100   //avbryts anslutningen genom timeout: status = 0
101   String strStatus;
102   strStatus = String(packets[pEnergy1].
103   connection) + String(packets[pEnergy2].connection)
104     + String(packets[pPower1].connection) + String(packets[pPower2].connection);
105   Serial.println(strStatus);
106 }
107 void setConnection(){
108   //Aktivera eller inaktivera paket manuellt via serieporten
109   delay(1000); //Alla efterföljande tecken från RPi ska hinna buffras
110   unsigned char tempStatus[4];
111   for (int i = 0; i < 4; i++){
112     tempStatus[i] = Serial.read();
113     if (tempStatus[i] == '1'){
114       tempStatus[i] = 1;
115     }
116     else{
117       tempStatus[i] = 0;
118     }
119   }
120   packets[pEnergy1].connection = tempStatus[0];
121   packets[pEnergy2].connection = tempStatus[1];
122   packets[pPower1].connection = tempStatus[2];
123   packets[pPower2].connection = tempStatus[3];
124 }
125
126
```

C Mätdatasystem: Pythonkod

accumulated.py

```
        # -*- coding: utf-8 -*-

import serial
import time
import os.path

def getData(ser):
#Hämta ackumulerade energivärden från arduino och
#returnera en lista med resultatet

    hexData = ''
    rawDataArray = [0]*28
    energyArray=[0]*7

    ser.write('1') #Frågar arduino efter energidata
    time.sleep(1)

    #Lägger in svaret från arduino i råformat i en lista. Tecken för
    #linefeed/carriage return sorteras bort
    for i in range (0, 28):
        response = ser.readline()
        rawDataArray[i] = response[0:4]

    #Lägger ihop varje rad i steg om fyra. Arduinon returnerar 16-bitars
    #hexadecimala block men värdena är 64 bitar
    for i in range(0,28,4):
        for j in range (i, i+4):
            hexData = hexData+rawDataArray[j]

        hexData='0x'+hexData #Symbolen 0x sätts före varje värde för att visa att
        #strängen är av hexadecimal typ
        floatData = float.fromhex(hexData) #Konverterar det hexadecimala värdet
        #till ett flyttal
        hexData = ''
        floatData = round(floatData*0.01,2) #Sätter rätt upplösning på värdet och
        #avrundar till två decimaler
        energyArray[i/4]=floatData #Placerar de slutgiltiga värdena i en ny lista
    return energyArray

def save(arrayToSave):
#Spara energivärden till fil

    activeImport = [0]*3
    activeExport = [0]*3

    #Filtrerar ut värden för importerad energi för solceller, Windstar och
    #AirDolphin
    for i in range (0,3):
        activeImport[i]=arrayToSave[i]

    #Sparar data till fil tillsammans med datum- och tidsstämpel
    file = open('Active_Import.txt', 'a')
    file.write(time.strftime("%Y-%m-%d") + ',' + time.strftime("%H:%M:%S") + '\n'
    + str(activeImport) + '\n')
    file.close()

    #Filtrerar ut värden för exporterad energi för solceller, Windstar och
    #AirDolphin
    for i in range (3,6):
        activeExport[i-3]=arrayToSave[i]

    file = open('Active_Export.txt', 'a')
    file.write(time.strftime("%Y-%m-%d") + ',' + time.strftime("%H:%M:%S") + '\n'
    + str(activeExport) +'\n')
    file.close()

```

```
    activeImportTot = arrayToSave[6] #Total importerad energi för anläggningens
    #drift
    file = open('Active_Import_Tot.txt', 'a')
    file.write(time.strftime("%Y-%m-%d") + ',' + time.strftime("%H:%M:%S") + '\n'
    + '[' + str(activeImportTot) + ']' +'\n')
    file.close()


```

momentary.py

```
        # -*- coding: utf-8 -*-
import serial
import time
import os.path

def getData(ser):
#Hämtar momentära data (spänning, ström, effekt) från arduino och returnerar i en
#lista

    hexData = ''
    rawDataArray=[0]*30
    momArray=[0]*15

    ser.write('2') #Frågar arduino efter aktuella värden
    time.sleep(1)

    #Lägger in svaret från arduino i råformat i en lista. Tecken för
    #linefeed/carriage return sorteras bort
    for i in range (0, 30):
        response = ser.readline()
        rawDataArray[i] = response[0:4]

    #Stegar igenom listan med rådata och formaterar värdena
    for i in range(0,30,2):
        for j in range (i, i+2):
            hexData = hexData+rawDataArray[j] #Hexadecimala 16-bitars block sätts
            #samman till 32 bitar.

        #Spänningar mellan faser och nolla samt mellan faserna
        if (i < 12):
            hexData='0x'+hexData #Symbolen 0x sätts före varje värde för att visa
            #att strängen är av hexadecimal typ
            floatData = float.fromhex(hexData) #Konverterar den hexadecimala
            #strängen till ett flyttal
            floatData = round(floatData*0.1,1) #Ger värdet rätt upplösning och
            #avrundas till en decimal
            momArray[i//2]=floatData

        #Strömmar i alla faser samt nolla
        if (i > 11 and i < 20):
            hexData='0x'+hexData
            floatData = float.fromhex(hexData)
            floatData = round(floatData*0.01,2)
            momArray[i//2]=floatData

        #Summerad aktiv effekt, effekt varje fas tillhörande de tre
        #kraftproducenterna, total effekt för anläggningen
        if (i > 19):
            hexData = '0x' + hexData
            iPower = int(hexData,16) #Konverterar hexadecimal sträng till integer

            #Lägger på tecken för att markera import/export av effekt
            if(iPower > 0x7FFFFFFF):
                iPower -= 0x100000000

            momArray[i//2] = round((float(iPower)) * 0.01,2) #Konverterar integer
            #till flyttal, fixar upplösning, avrundar

        hexData = ''
```

```
60      return momArray
61
62  def save(arrayToSave):
63  #Sparar momentära data till fil
64
65      voltage = [0]*6
66      current = [0]*4
67
68      #Filtrerar ut alla spänningsvärden
69      for i in range (0,6):
70          voltage[i]=arrayToSave[i]
71
72      #Sparar data till fil tillsammans med datum- och tidsstämpel
73      file = open('Voltage.txt', 'a')
74      file.write(str(voltage) + ',' + time.strftime("%Y-%m-%d") + ',' +
75      time.strftime("%H:%M:%S") + '\n')
76      file.close()
77
78      #Filtrerar ut alla strömvärden
79      for i in range (6,10):
80          current[i-6]=arrayToSave[i]
81
82      file = open('Current.txt', 'a')
83      file.write(str(current) + ',' + time.strftime("%Y-%m-%d") + ',' +
84      time.strftime("%H:%M:%S") + '\n')
85      file.close()
86
87      powerSite=arrayToSave[len(arrayToSave)-1] #Anläggningens totala
88      #effektkonsumtion
89      file = open('Total_Power_Site.txt', 'a')
90      file.write(str(powerSite) + ',' + time.strftime("%Y-%m-%d") + ',' +
91      time.strftime("%H:%M:%S") + '\n')
92      file.close()
93
94
95
```

webConnection.py

```
1  import json
2  import urllib2
3  import logging
4  import httplib
5  APIKEY="YOUR_API_KEY" #Nyckel och konto maskerade av säkerhetsskäl
6  DEVICE = "energyHono@YOUR_ACCOUNT.YOUR_ACCOUNT"
7
8  class Client (object):
9          api_url = "http://api.carriots.com/streams"
10
11         def __init__ (self, api_key = None, client_type = json):
12         #Initierar http-paket
13                 self.client_type = client_type
14                 self.api_key = api_key
15                 self.content_type = "application/json"
16                 self.headers = {'User-Agent': 'Raspberry-Carriots',
17                                 'Content-Type': self.content_type,
18                                 'Accept': self.content_type,
19                                 'Carriots.apikey': self.api_key}
20
21         def send (self, data):
22         #Skickar http-paketet
23                 self.data = json.dumps(data)
24                 request = urllib2.Request(Client.api_url, self.data,
25                 self.headers)
26
27                 #Tar hand om ev felmeddelanden om internetuppkopplingen
28                 #bryts e. dyl
29                 try:
30                     self.response = urllib2.urlopen(request)
31                 except urllib2.HTTPError, e:
```

```
                    logging.error('HTTPError = ' + str(e.code))
                    self.response = e.code
                except urllib2.URLError, e:
                    logging.error('URLError = ' + str(e.reason))
                    self.response = e.reason
                except httplib.HTTPException, e:
                    logging.error('HTTPException')
                    self.response = 'HTTP Exception'
                except Exception:
                    import traceback
                    logging.error('generic exception: ' + traceback.format_exc())
                    self.response =
                    'generic exception: ' + traceback.format_exc()
                return self.response

def sendData(prodSolar, prodWindstar, prodAirDolphin, consSite,pSolar,
pWindstar, pAirDolphin, pSite):
#Funktion för att skicka data till Carriots

    #Klumpar ihop anläggningens effektkonsumtion med solpanelernas effekt om
    #omriktare konsumerar mer effekt än vad som produceras
    if(pSolar > 0):
        pSite = pSite + pSolar
        pSolar = 0

    if(pWindstar > 0):
        pSite = pSite + pWindstar
        pWindstar = 0

    #Effektvärden är negativa vid export. För ökad tydlighet sätts alla värden
    #till positiva
    pSolar = -pSolar
    pWindstar = - pWindstar
    tot_power = pSolar + pWindstar #Total producerad effekt

    #Data som ska skickas görs redo i JSON-format
    data = {"protocol":"v2","device":DEVICE,"at":"now","data":{
        "solarCell_energy_production": prodSolar,
        "windstar_energy_production": prodWindstar,
        "airDolphin_energy_production": prodAirDolphin,
        "site_energy_consumtion": consSite,
        "solarCell_power": pSolar,
        "windstar_power": pWindstar,
        "airDolphin_power": pAirDolphin,
        "site_power": pSite,
        "total_power": tot_power
        }}

    client_carriots = Client(APIKEY)
    carriots_response = client_carriots.send(data)
    return carriots_response

```

measureMain.py

```
import serial
import time
import accumulated
import momentary
from collections import deque
import webConnection

ser = serial.Serial('/dev/ttyACM0', 9600) #Initierar serieporten för
#kommunikation med arduino
time.sleep(2)

#Deklarerar köer för att spara energidata en gång per minut i 30 minuter
qSolarCellProd = deque([])
qWindstarProd = deque([])
qAirDolphinProd = deque([])
```

```
qSolarCellCons = deque([])
qWindstarCons = deque([])
qAirDolphinCons = deque([])
qSiteCons = deque([])

m = 0
n = 0
k = 0

tempArray = [0] * 15 #Deklarerar en temporär lista, 15 enheter lång,
#innehållandes nollor
momArray = [0] * 15 #Här placeras alla aktuella momentanvärden
#(ström, spänning, effekt)
momToFileArray = [0] * 15 #Här lagras medelvärdesbildade momentanvärden som
#ska sparas till fil
energyArray = [0] * 4 #Lista för ackumulerad energi
powerArray = [0] * 4 #Temporär lista för aktuella effektvärden
previousTime1 = 0
previousTime5 = 0
previousTime60 = 0
previousTime600 = 0

while 1:

    if(1 <= (time.time() - previousTime1)): #Kod som körs varje sekund
        tempArray = momentary.getData(ser) #Hämtar aktuella momentanvärden
        momArray[:] = [sum(i) for i in zip(momArray,tempArray)] #Summerar varje
        #enskilt värde med korresponderande föregående värden
        n = n + 1 #Räknar antal gånger koden körts
        previousTime1 = time.time() #Uppdatera tidsstämpeln för senaste körning

    if(5 <= (time.time() - previousTime5)): #Kod som körs var femte sekund
        carriotsArray = [0] * 4 #Deklaration för lista för olika värden att
        #skicka till Carriots

        momArray[:] = [x / n for x in momArray] #Medelvärdesbildar de senaste fem
        #sekundernas mätvärden
        momToFileArray[:] = [sum(i) for i in zip(momToFileArray,momArray)]

        #Plockar effektvärdena ur momentanvärdeslistan och placerar dem först i
        #en ny
        for j in range(len(momArray) - 4,len(momArray)):
            powerArray[j - (len(momArray) - 4)] = momArray[j]

        #Kopierar energiinfo
        for j in range(0,4):
            carriotsArray[j] = energyArray[j]

        carriotsArray.extend(powerArray) #Utökar listan så den innehåller energi-
        #och effektvärden

        #Avrundar alla värden till två decimaler
        for j in range(0,8):
            carriotsArray[j] = round(carriotsArray[j],2)

        response = webConnection.sendData(*carriotsArray) #Skickar iväg
        #informationen till Carriots
        momArray = [0] * (len(momArray)) #Nolställer listan
        m = m + 1 #Räknar antal gånger koden körts
        n = 0 #Nollställer 1-sekundersräknaren
        previousTime5 = time.time()

    if(59 <= (time.time() - previousTime60)): #Kod som körs en gång varje minut

        #Medelvärdesbildar och avrundar
        momToFileArray[:] = [x / m for x in momToFileArray]
        for i in range(0,len(momToFileArray)):
            momToFileArray[i] = round(momToFileArray[i],2)

        #Om energimätarna slutat fungera förkastas datan, listan nollställs
```

```
86          if (checkStatus() == 0):
87              momToFileArray = [0] * 15
88
89          momentary.save(momToFileArray) #Data sparas till fil
90          momToFileArray = [0] * 15 #Rensa listan
91
92          tempArray = accumulated.getData(ser) #Hämtar ackumulerade energivärden
93
94          #Lägger till värdena i respektive kö
95          qSolarCellCons.append(tempArray[0])
96          qWindstarCons.append(tempArray[1])
97          qAirDolphinCons.append(tempArray[2])
98          qSolarCellProd.append(tempArray[3])
99          qWindstarProd.append(tempArray[4])
100         qAirDolphinProd.append(tempArray[5])
101         qSiteCons.append(tempArray[6])
102
103         k = k + 1
104
105         #Då koden körts 30 ggr, det vill säga i 30 minuter, jämför det
106         #äldsta värdet i kön med det senaste
107         if(k >= 30):
108             fSolarCellCons = tempArray[0] - qSolarCellCons.popleft()
109             fWindstarCons = tempArray[1] - qWindstarCons.popleft()
110             fAirDolphinCons = tempArray[2] - qAirDolphinCons.popleft()
111             fSolarCellProd = tempArray[3] - qSolarCellProd.popleft()
112             fWindstarProd = tempArray[4] - qWindstarProd.popleft()
113             fAirDolphinProd = tempArray[5] - qAirDolphinProd.popleft()
114             fSiteCons = tempArray[6] - qSiteCons.popleft()
115
116             fSiteCons = fSiteCons + fSolarCellCons + fWindstarCons #Klumpar ihop
117             #all konsumtion till ett värde
118             energyArray = [fSolarCellProd, fWindstarProd,fAirDolphinProd,
119             fSiteCons]
120         m = 0 #Nollställer 5-sekundersräknaren
121         previousTime60 = time.time()
122
123     if(600 <= (time.time() - previousTime600)): #Kod som körs var tionde minut
124         #Fungerar energimätarna hämtas och sparas aktuell ackumulerad energi,
125         #annars sparas, nollvärden
126         if(checkStatus() == 1):
127             tempArray = accumulated.getData(ser)
128         else:
129             tempArray = [0] * 15
130         accumulated.save(tempArray) #Spara till fil
131
132         previousTime600 = time.time()
133
134         #Testar att omaktivera arduinons modbuspaket energimätarna slutat
135         #fungera
136         if(checkStatus() == 0):
137             setStatus()
138
139     time.sleep(0.01)
140
141
142 def checkStatus():
143 #Funktion för att kontrollera att energimätarna fortfarande svarar som de ska
144
145     ser.write('3') #Frågar efter hälsostatus hos energimätarna
146     time.sleep(1) #Väntar en sekund för svar
147     status = 1
148     statusArray = []
149
150     #Hämtar seriell data från cachen och lägg till lista
151     for i in range(0,4):
152         statusArray.append(ser.read())
153         if(statusArray[i] == '0'): #Om något modbuspaket hos Arduino slutat
154         #fungera
155             status = 0
```

```
156     ser.flushInput() #Rensa cachen
157     return status
158 
159 def setStatus():
160 #Aktiverar modbuskommunikationen
161     ser.write("41111") #Aktiverar samtliga fyra paket hos arduinon
162     time.sleep(22) #Pausar körningen i 22 sekunder för att låta paket gå timeout
163     #om arduinon fortfarande inte får svar från mätarna
164 
165 
```

D Bromssystem: Arduinokod

mainBrake.ino

```
//Konstanter som definierar pinnummer
const int redLed = 12;
const int greenLed = 11;
const int stopBtn = 10;
const int startBtn = 9;
const int relayPin = 4;
const int brakePin = 13;

const float brakeFreq = 66; //Frekvens då bromsen aktiveras.
//66 Hz motsvarar 792 rpm
float fFreq = 0;
float tempFreq = 0;
float prevTime = 0;
float currTime = 0;
int lastBlink = 0;
unsigned long nInterrupted = 0;
char charReceived;
bool bBrake = false;
bool zero = false;
bool blink = false;

void setup()
//Kod som körs vid uppstart
{
  Serial.begin(9600); //Initierar serieporten

  //Sätter pinnar till utgångar
  pinMode(brakePin, OUTPUT);
  pinMode(relayPin, OUTPUT);
  pinMode(redLed, OUTPUT);
  pinMode(greenLed, OUTPUT);

  //Aktiverar ingångar
  pinMode(startBtn, INPUT);
  pinMode(stopBtn, INPUT);

  brake(); //Systemet startar alltid med bromsen aktiverad
  attachInterrupt(0, freq, RISING); //Aktivera avläsning av frekvens utifrån
  delay(3000); //Paus i tre sekunder stoppar felaktiga initiala triggningar
}

void loop()
//Kod som körs kontinuerligt
{
  if (nInterrupted >= 2){ //Om vi fått minst två positiva flanker kan
  //frekvensen mätas
    detachInterrupt(0); //Stänger av triggning för att förhindra att variabler
    //ändras under operationen
    tempFreq = 1/((currTime - prevTime)*0.000001); //Beräknar frekvensen
    //av periodtiden
    if ((tempFreq < 280) && (tempFreq >= 0)){ // Filtrerar bort störspikar över
    //280 Hz, samt förhindrar problem med
    //negativa frekvenser då micros() får overflow
      fFreq = tempFreq;
      zero = false;
    }
    //Sätter frekvensen till noll om periodtiden överstiger en halv sekund.
    if (fFreq < 2 || (1 / ((micros() - currTime)*0.000001)) < 2){
      if (!zero){
        nInterrupted = 0;
        fFreq = 0;
        zero = true;
      }
    }
    attachInterrupt(0, freq, RISING); //Återaktiverar triggning
```

```
  }

  //Lyssnar på serieporten för fjärrstopp och start
  if (Serial.available()){
    charReceived = Serial.read();
    if (charReceived == '1'){
      brake();
    }
    if (charReceived == '2'){
      start();
    }
    if (charReceived == '3'){
      Serial.println(fFreq); //Returnerar senast uppmätta frekvens
    }
  }

  //Börja bromsa om frekvensen blir för hög
  if (fFreq > brakeFreq){
    brake();
  }

  //Bromsa om stoppknappen trycks in
  if (digitalRead(stopBtn) == LOW){ //inverterad switch
    brake();
  }

  //Starta om starknappen trycks in
  if (digitalRead(startBtn) == LOW){ //inverterad switch
    start();
  }
  if (bBrake){
    blinkRed(); //Låt röd lampa blinka vid nedbromsning

    //Släpp relä om frekvens understiger 5 Hz och broms är aktiverad
    if (fFreq < 5){
        digitalWrite(relayPin, LOW);
        digitalWrite(redLed, HIGH); //Röd lampa lyser konstant
        blink = false;
    }

  }
}
void brake(){
  digitalWrite(brakePin, LOW); //Låter spänningsreferensen gå hög då
  //transistorn bryts, bromsprocessen påbörjas
  digitalWrite(greenLed, LOW); //Släcker grön lampa
  bBrake = true;
}

void start(){
  bBrake = false;
  digitalWrite(relayPin, HIGH); //Drar reläet och släpper därmed kortslutning
  digitalWrite(brakePin, HIGH); //För spänningsreferensen till jord,
  //bromsen inaktiveras
  digitalWrite(greenLed, HIGH); //Tänd grön lampa
  digitalWrite(redLed, LOW); //Släck röd lampa
}

void blinkRed(){
//Tänd och släck röd lampa med en sekunds intervall

  int blinkTime = millis();
  if (blinkTime - lastBlink >= 1000){
    lastBlink = blinkTime;
    blink = !blink;
    digitalWrite(redLed, blink);
  }
}

void freq(){
```

```
136 //Körs vid varje positiv flank, räknar ut tidsskillnaden mellan flankerna
137 //d.v.s periodtiden
138   prevTime = currTime;
139   currTime = micros();
140   nInterrupted += 1;
141 }
142
```

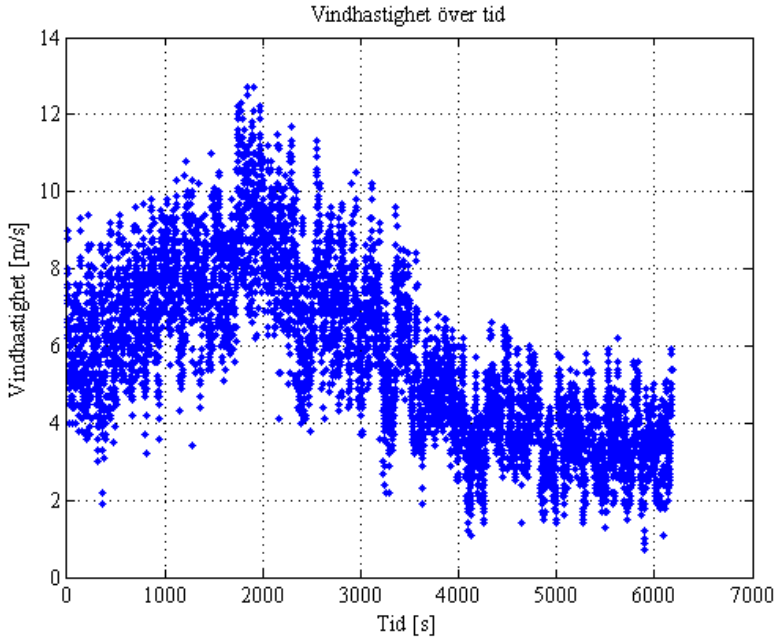
E MATLAB-kod för sammanställning av mätvärden från LabVIEW

```
1         %read_H4_1ss.m
2 %Uppläsningsprogram för Hönö Småprod 1s mätfiler.
3 %Genererade med programpaketet C
4 % De är sparade i textformat med mätvärdena i
5 %kolumner. Intervallet mellan mätvärdena är 1 sekund.
6
7 % 2015-05-05 Magnus Ellsen.
8
9 clear, close all, clc
10 workdir='C:\Users\jonwin\Dropbox\Kandidatarbete VT15\FrMagnus\matdata';
11 matdatadir='C:\Users\jonwin\Dropbox\Kandidatarbete VT15\FrMagnus\matdata';
12 cd (matdatadir);
13 [filnamn,stig]=uigetfile('*.lvm','Ange mätfil att läsa upp!');
14 name = [stig,filnamn];
15 % name = 'C:\Users\mellsen\Dropbox\Kandidatarbete
16 % VT15\FrMagnus\matdata\Windstar_1s_data_15-05-04_0000.lvm';
17
18 fs = 1;  % Scans/s
19
20 matdata=load(name);
21
22
23 cd (workdir);
24
25 %Channel separation:
26
27 Time=matdata(:,1);          % DC Current REFerence (mean) [A]
28 WD=matdata(:,2);             % DC Current measured (mean) [A]
29 WS=matdata(:,3);             % DC Voltage measured (mean) [V]
30 AIRTA=matdata(:,4);              % Rotor Speed Analog (mean) [rpm]
31 AIRPA=matdata(:,5);         % Rotor Teeter Angle level, max(RTA)-min(RTA) [degrees]
32 RINT=matdata(:,6);              %   (mean) [kNm]
33 Uhac_WT2=matdata(:,7);            %   (mean) [kNm]
34 Iac_WT2=matdata(:,8);             %   (mean) [kNm]
35 Udc_WT2=matdata(:,9);               %   (mean) [meas.volts]
36 Idc_WT2=matdata(:,10);              %   (mean) [degrees]
37 n_WT2=matdata(:,11);          %   max(abs(HYMP1-HYMP2))  [bar]
38 Pgrid_WT2=matdata(:,12);          %   max(abs(NAX1-NAX2))  [m/s2]
39 Pgrid_PV=matdata(:,13);          %   max(abs(NAY1-NAY2))  [m/s2]
40 Udc_WT3_AD=matdata(:,14);          %   max(abs(NAZ1-NAZ2))  [m/s2]
41 Idc_WT3_AD=matdata(:,15);            %   (mean) [kW]
42
43 t=(0:length(Time)-1);       % Time vector [s]
44
45 figure(1)
46 plot(t,WS,'.'),grid on
47 title('Vindhastighet över tid')
48 xlabel('Tid [s]')
49 ylabel('Vindhastighet [m/s]')
50
51 figure(2)
52 plot(t,n_WT2,'.'),grid on
53 title('Varvtal över tid')
54 xlabel('Tid [s]')
55 ylabel('Varvtal [rpm]')
56
57 figure(3)
58 plot(WS,n_WT2,'.')
59 grid on
60 title('Vindhastighet mot varvtal')
61 xlabel('Vindhastighet [m/s]')
62 ylabel('Varvtal [rpm]')
63
64 addpath('C:\Users\Jonwin\Dropbox\VindkraftlabbMatlab');
65 import bin.*
66
```

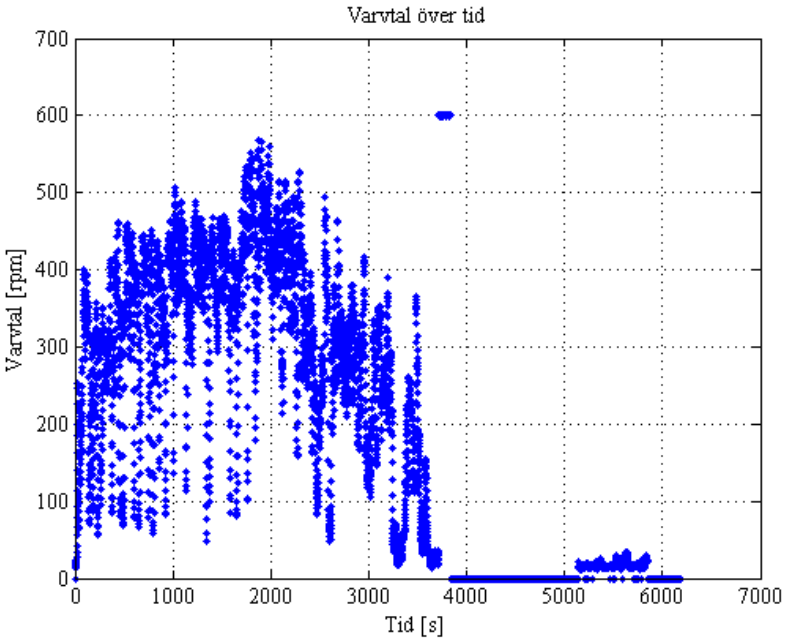
```
figure(4)
plot(WS,Udc_WT2)
grid on
title('Spänning mot vindhastighet')
xlabel('Vindhastighet [m/s]')
ylabel('Spänning [V]')

[xvect,yvect] = bin(WS,Udc_WT2);

figure(5)
plot(xvect,yvect)
grid on
title('Spänning mot vindhastighet (förenklad graf)')
xlabel('Vindhastighet [m/s]')
ylabel('Spänning [V]')

[xvect2,yvect2] = bin(WS,n_WT2);

figure(6)
plot(xvect2,yvect2)
grid on
title('Vindhastighet mot varvtal (förenklad graf)')
xlabel('Vindhastighet [m/s]')
ylabel('Varvtal [rpm]')

figure(7)
plot(t,Udc_WT2),grid on
title('Spänning över tid')
xlabel('Tid [s]')
ylabel('Spänning [V]')

[xvect3,yvect3] = bin(WS,-Pgrid_WT2);


```

F Mätvärden från LabVIEW, sammanställda i MATLAB

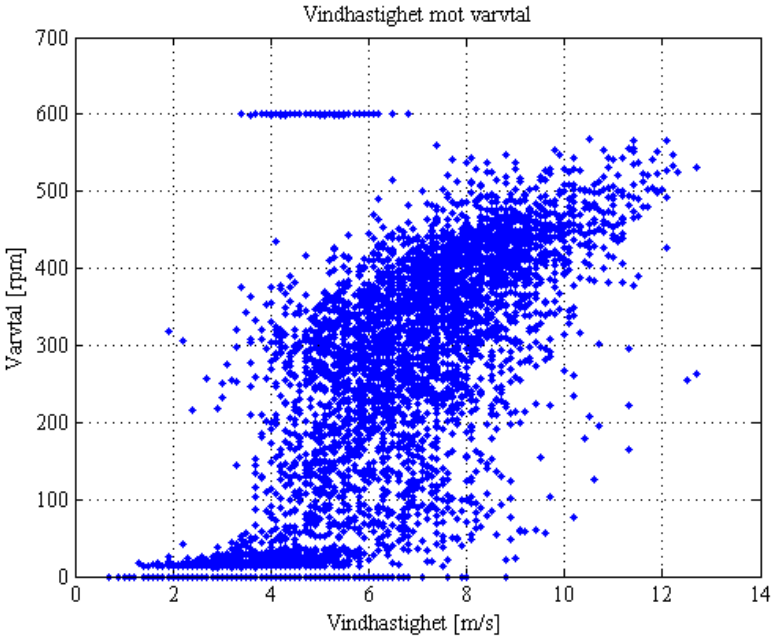
,



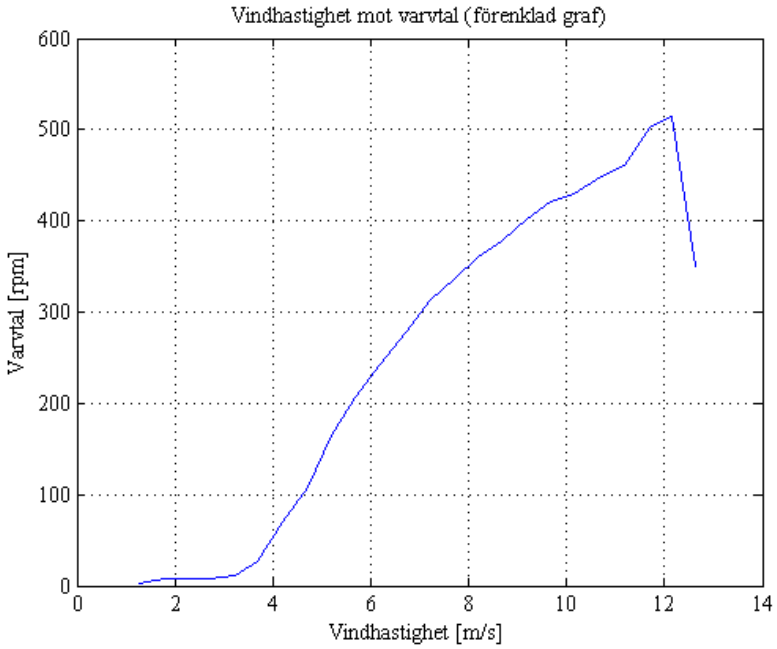
Figur 25: *Graf över uppmätt vindhastighet.*



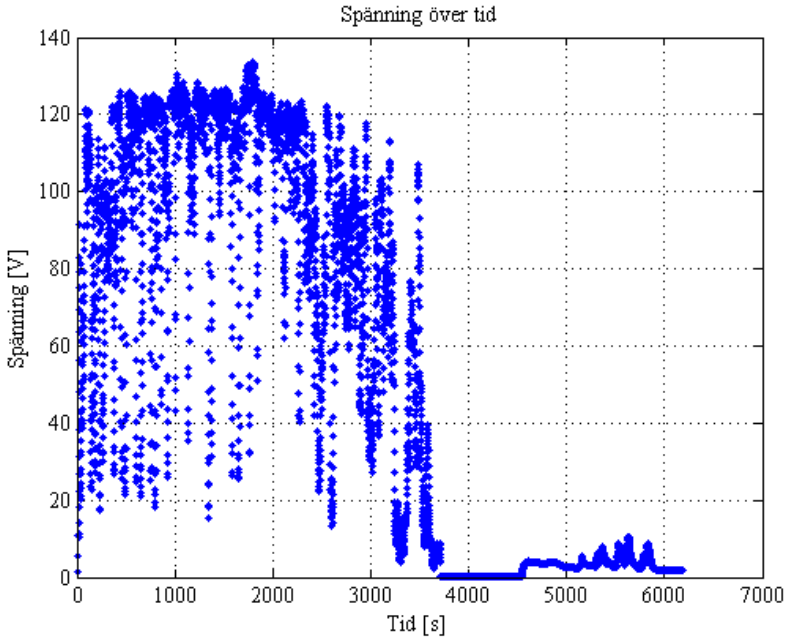
Figur 26: *Graf över uppmätt varvtal.*



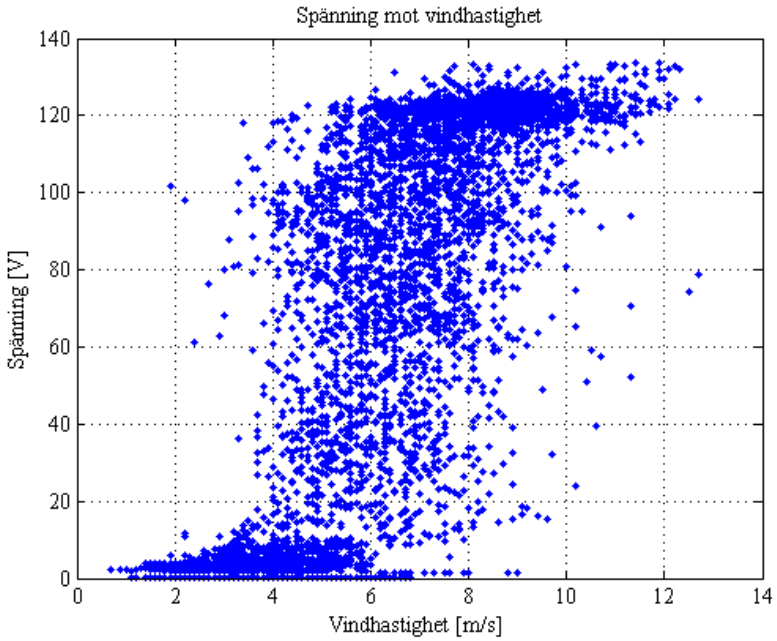
Figur 27: *Graf över vindhastighet mot varvtal.*



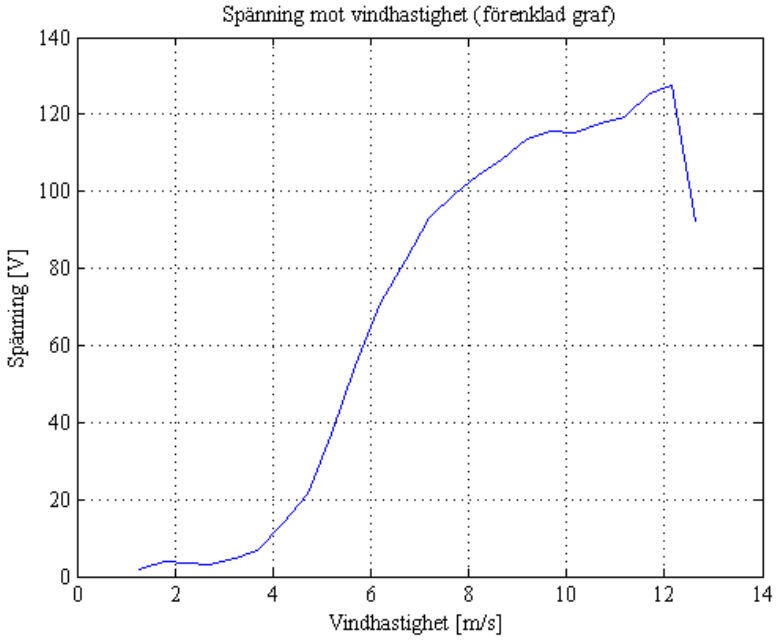
Figur 28: *Förenklad graf över vindhastighet mot varvtal.*



Figur 29: *Graf över uppmätt spänning.*



Figur 30: *Graf över spänning mot vindhastighet*



Figur 31: *Förenklad graf över spänning mot vindhastighet.*

G MATLAB-kod för avrundning av värden i grafer

```
function [xvect,yvect] = bin (x,y)
%function [xvect,yvect] = bin (x,y) calculates the averaged curve, xvect, yvect,
%of the input data points x,y according to the method of bins.

xbin(40)=0;
ybin(40)=0;
antal(40)=0;
for j=1:length(x)
  fack=floor(2*x(j));
  if fack > 1
    xbin(fack)=xbin(fack)+x(j);
    ybin(fack)=ybin(fack)+y(j);
    antal(fack)=antal(fack)+1;
  end
end
xvect=xbin ./ antal;
yvect=ybin ./ antal;
```

H MATLAB-kod för C_p/λ-kurvor

```
%% Regnbågsgraf Cp/lambda
clc
clf
Lambda = 0;
tickVector = [0:5:120];
cp=[0, 0.05,0.2, 0.3, 0.38, 0.45, 0.478, 0.45, 0.41, 0.38, 0.35, 0.32, 0.3, 0.25...
    0.2, 0.18, 0.15, 0.14, 0.13, 0.1, 0.1];
vind=[0:20];
n = [0:100:700];
r = 1.525;
omega = (n.*(2*pi))./60;
v_tip = omega.*r;
Lambda = zeros(8, 21, 'uint32');
for i=1:8
    for j=1:21
    Lambda(i,j) = v_tip(i)./vind(j);
    end
end
cmap = hsv(8);

figure(1)
for i=1:8
plot(Lambda(i,:),cp,'-*','Color',cmap(i,:))
hold on
end

grid on
axis([0 120 0 0.5])
xlabel('Lambda');
ylabel('C_p');
set(gca,'XTick',tickVector);
legend('n = 0','n = 100','n = 200','n = 300','n = 400','n = 500','n = 600',
       'n = 700')
```

I MATLAB-kod för framtagning av vind-effektkurva

```
%% Verklig vindhastighet/effekt-kurva

n = [0 100 200 300 400 500 600];                    % Varvtal [rpm]
Lambda = [0 3 5 6 7 7 7];                           % Lambda (0 ren formalitet)
Cp = [0 0.478 0.478 0.41 0.38 0.32 0.25];           % Cp
P = [0 321.8 557.1 931.58 1291.2 2300 3000];        % Effekt

r = 1.525;

%omega = (n.*(2*pi))./60;
%v_tip = omega.*r;
%vind_h = v_tip./Lambda;
vind_h = [];

for i = 1:7
    vind_h(end+1) = (n(i)*(2*pi)*r)/(60*Lambda(i));
end
vind_h(1) = 0;

figure(1)

plot(vind_h,P)
grid on

title('Verklig vindhastighet/effekt-kurva')
xlabel('Vindhastighet [m/s]')
ylabel('Effekt [W]')
```

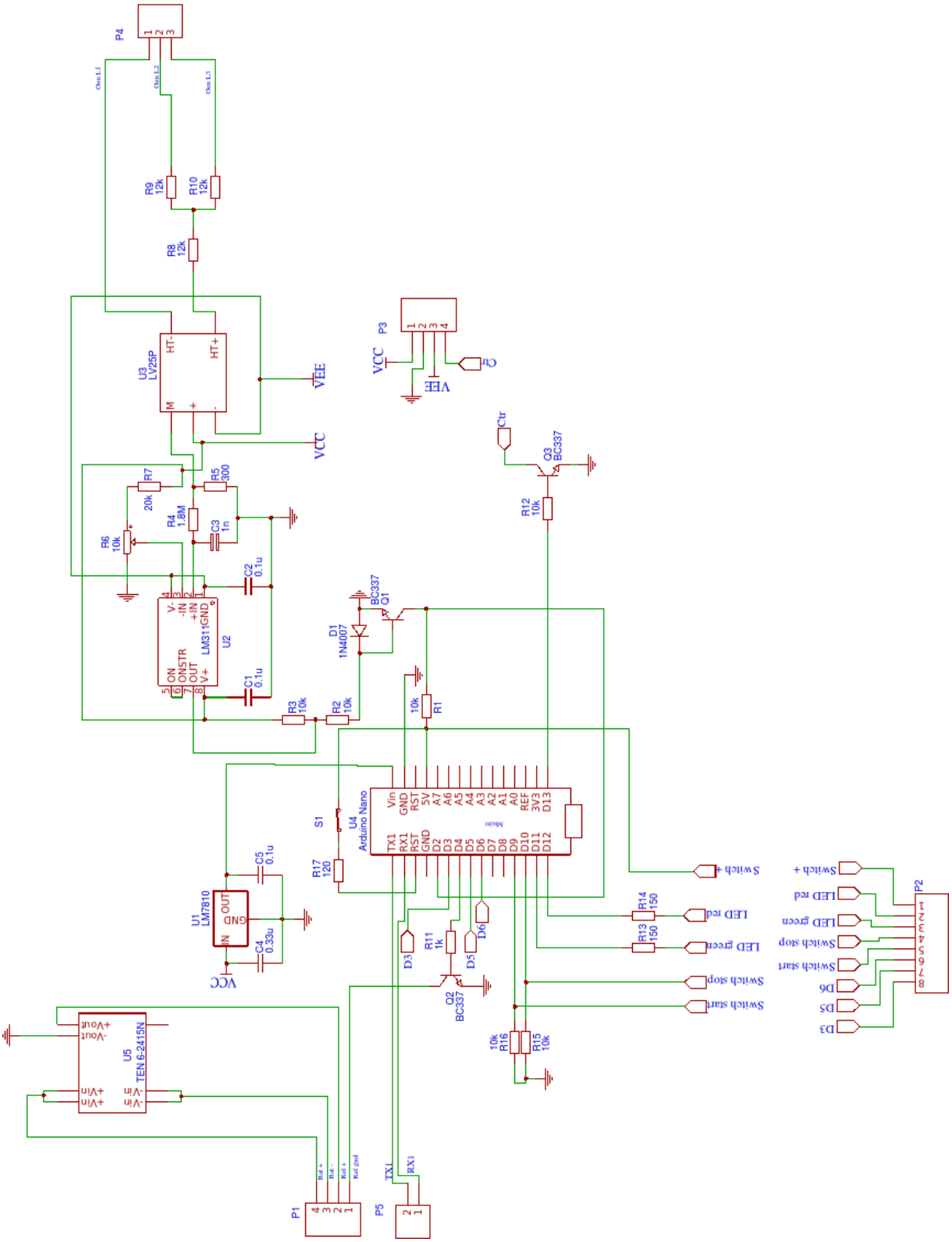
J MATLAB-kod för olika belastningskurvor

```
clear, close all
clc

tickVector = [0:25:300];
n = [0 100 200 300 400 500 600];                % Varvtal [rpm]
Lambda = [0 3 5 6 7 7 7];                       % Lambda (0 ren formalitet)
Cp = [0 0.478 0.478 0.41 0.38 0.32 0.25];       % Cp (0 ren formalitet)
v = [0 5.32 6.39 7.98 9.13 11.41 13.69];        % Vindhastighet [m/s]
U = [0 41 75 102.1 113.05 121.9 130];           % Spänning [V]
P = [0 321.8 557.1 931.58 1291.2 2300 3000];    % Effekt [W]
U_f = [0:10:300];
P_f = [0 5 15 50 75 100 150 175 200 350 700 1500 2500];
for i = 14:21
    P_f(end+1) = 25*U_f(i);
end
for i = 22:31
    P_f(end+1) = 5000;
end

U_om = [0 10 20 30 40 50 60 70 80 90 100 110 120 130 140 150 160 170 180 190...
200 210 220 230 240 250];
P_om = [0 78.5 157 235.5 314 384.1 453.3 522.5 626.2 764.4 902.5 1191 2084 3000...
3500 3750 4000 4250 4500 4750 5000 5000 5000 5000 5000 5000];

figure(1)
plot(U,P,'-*r'), hold on %Teoretiskt uträknad kurva
plot(U_om,P_om,'-*b'), hold on %Kurva anpassad till omriktaren
plot(U_f,P_f,'-*m') %Kurva framtagen genom tester
grid on
xlabel('Spänning [V]');
ylabel('Effekt [W]');
title('Belastningskurva');
legend('Uträknad kurva','Kurva anpassad till omriktaren',
       'Kurva framtagen genom tester','Location','southeast')
set(gca,'XTick',tickVector);
```

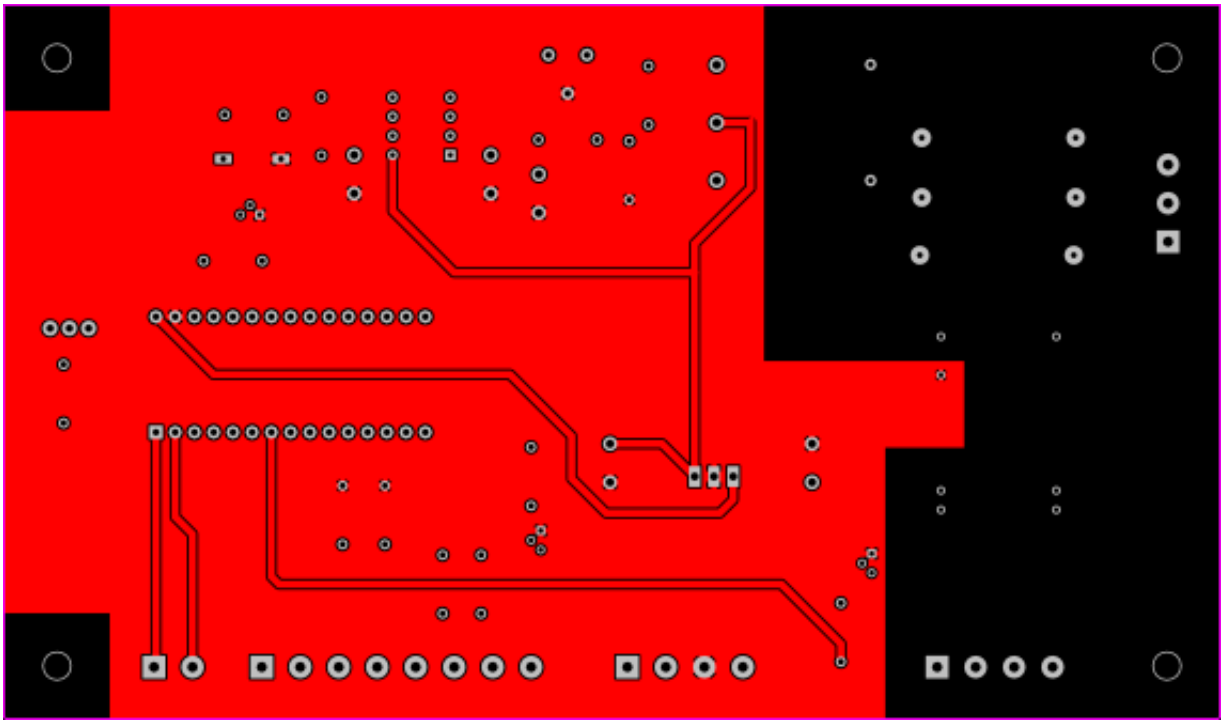
K MATLAB-kod för energiutvärdering

```
%% Sol-, vindenergi och konsumtion för ett år
clc
clf
Soltimmar_tot = [45,71.33333333,198,207,260.6666667,242,272.3333333,197.3333333,
173,77.33333333,36.33333333,36.33333333];
kWh_tot = 1.9.*Soltimmar_tot; %Det går 1.9 kWh/soltimme enligt tidigare räkning
x = {'Jan','Feb','Mar','Apr','Maj','Jun','Jul','Aug','Sep','Okt','Nov','Dec'};
Cons = [378.28, 413.6, 240.76, 184.18, 145.05, 220.44, 180.27, 186.97, 195.53,
301.85, 299.37, 406.78];
Vindhastighet = [5.3075 4.995 5.1575 4.596666667 5.59 5.403333333 5.2675 5.2825...
6.91 6.32 6.676666667 6.236666667];
P_vind = [320.85 302 311.78 277.875 380.8 339.49 318.43 319.34 679.5 542.1 624.8
523.67]./1000;
Timmar_manad = [744 672 744 720 744 720 744 744 720 744 720 744];
kWh_vind = P_vind.*Timmar_manad; %Beräkning för att få fram kWh för Windstar 3000.

Prodsol = sum(kWh_tot./1000);
Cons_tot = sum(Cons);
Wind_tot = sum(kWh_vind)

Sum_tot = Prodsol + Wind_tot - Cons_tot

figure(1)
plot([1:12],kWh_tot,'b'), hold on
plot([1:12],Cons,'r'), hold on
plot([1:12],kWh_vind,'m')
grid on
set(gca, 'XTick',1:12, 'XTickLabel',x)
legend('Solcellsproduktion','Konsumtion','Vindkraftproduktion')

title('Energiproduktion/konsumtion per månad (anläggning)')
ylabel('Energi [kWh]')
xlabel('Månad')
```

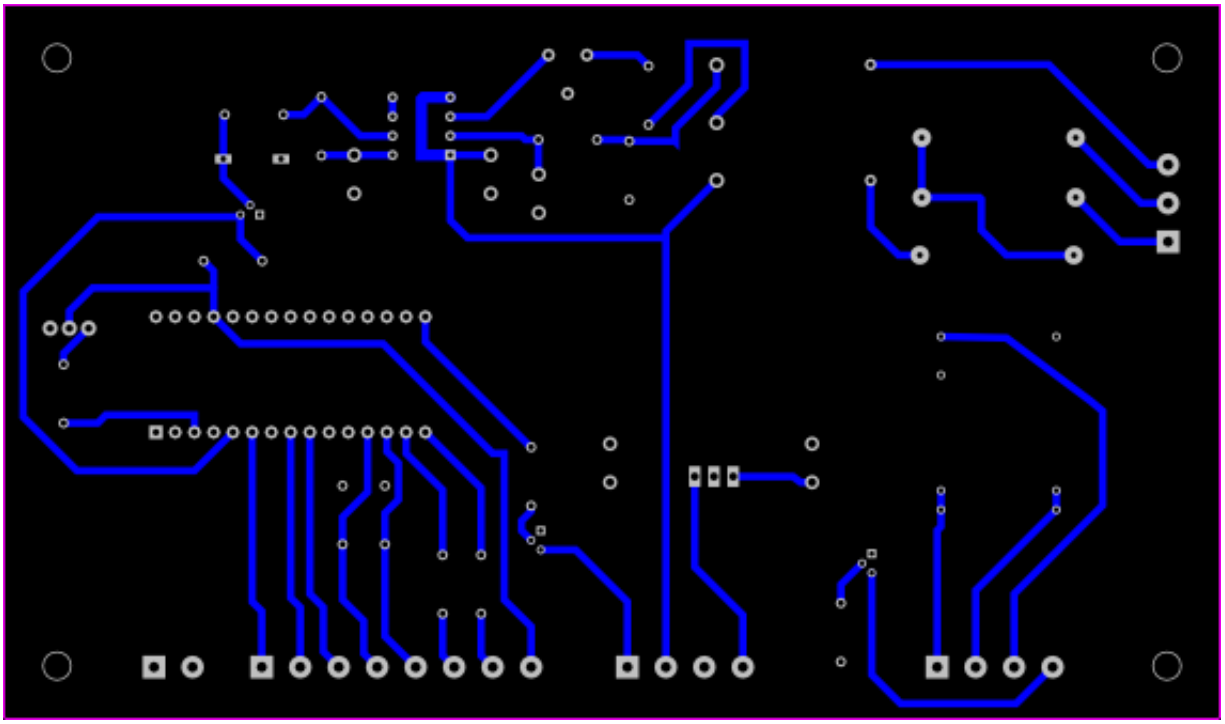
L Schema och PCB till frekvenskretsen



Figur 32: *Schemat till frekvenskretsen.*



Figur 33: *PCB topskickt till frekvenskretsen.*



Figur 34: *PCB bottenskikt till frekvenskretsen.*