Fysisk realtidsmodellering av ett reglerbart vindkraftverk

Daniel Haverås och Adam Richert

***Sammanfattning*—Den här rapporten ämnar beskriva utvecklingen och konstruktionen av en Linuxbaserad hårdvaruprototyp som i realtid simulerar det kommersiella vindkraftverket Enercon E-44. Projektet är delvis en vidareutveckling av projektet "Fysisk modellering av IEEE 14-bus test system" som genomfördes våren 2014 på avdelningen för ICS, Industriella Informations- och Styrsystem vid KTH, Stockholm. Där utvecklades en fysisk modell av ett virtuellt AC-nät avsett för simuleringar i utbildningssyfte. Syftet med detta projekt är att utveckla ett virtuellt vindkraftverk som kan anslutas till elnätsmodellen. Modelleringen kan då göras mer verklighetstrogen eftersom energikällornas dynamik tas med i simuleringen. En virtuell modell av ett vindkraftverk skapades i programmet Simulink, för att sedan konverteras till C-kod. Koden anpassades för realtidskörning och kompilerades på enkortsdatorn Raspberry Pi. Resultatet blev en fysisk enhet som i realtid tar in en vindhastighet och matar ut värden på producerad aktiv effekt samt förbrukad reaktiv effekt. En delvis egenutvecklad A/D-omvandlare ansluten till en 220 kΩ vridpotentiometer användes för att leverera den simulerade vindinsignalen, som kan variera mellan 0 och 31 m/s. Utdata skrivs i Linuxkonsolen 10 ggr/sekund. Tester visade att vindkraftsmodellens beräkningar gav värden som ligger mycket nära de hos Enercon E-44. Anpassningar behöver göras i form av mindre ändringar i C-koden och tillägg av analoga komponenter för att modellen ska kunna anslutas till den befintliga elnätsmodellen hos ICS.**

I. INTRODUKTION

DAGENS samhälle är beroende av elektricitet. I Sverige ligger elförbrukningen på drygt 130 TWh [1] årligen, vilket motsvarar ungefär 14 000 kWh per capita. Med en sådan förbrukning är det av stor vikt att ha ett väl fungerande elnät som transporterar rätt mängd el dit det behövs. Samtidigt måste nätet klara av att det sker förändringar i både elförbrukningen och elproduktionen från år till år. Om ett kärnkraftverk läggs ned måste motsvarande el produceras av en annan källa för att få in samma mängd i systemet. På något sätt måste följderna av dessa förändringar förutspås och planeras för innan kärnkraftverket väl tas ur bruk. Detta innebär ofta komplicerade och utförliga beräkningar av aktiva och reaktiva effekter samt komplexvärda strömmar och spänningar.

Ett sätt som underlättar och begränsar beräkningarna är att skapa simuleringar av elnätet. Med rätt simuleringsverktyg kan modeller av verkligheten konstrueras och följder av förändringar studeras för att få en uppfattning av exempelvis vad som kommer ske när källor och laster läggs till eller plockas bort från elnätet.

Fördelen med traditionella icke-förnybara energikällor som exempelvis kol, olja och gas är att de är fullständigt reglerbara. Vid ökat energibehov från nätet kan kraftverken snabbt gå upp till maxkapacitet genom att öka mängden bränsle till generatorerna. Effekten som utvinns är alltså i högsta grad deterministisk, beroende endast på bränsletillgång och kraftverkens kapacitet. Utvinning av förnybara källor, exempelvis sol- och vindkraft, ger istället en delvis stokastisk effekt som är beroende av väder och miljö. Den utvunna effekten kan endast regleras nedåt, inte uppåt. Detta leder till att områden med varierande väder och klimat inte lokalt kan förlita sig på vindkraft. Nätet måste kunna byta energikälla vid utebliven vind, vilket ger mer osäkerhet och en större dynamik i nätet. En ingenjör vill kunna förutspå och vara beredd på dessa förändringar i nätet, vilket gör att simulering blir ett mycket viktigt verktyg.

Även om simuleringar oftast görs i mjukvara på virtuella nät är det i utbildningssyfte användbart att ha fysiska komponenter och block för att få en god överblick och lära sig hur de olika beståndsdelarna fungerar och interagerar med varandra. År 2014 dedikerades ett kandidatexjobb på KTH till att modellera elnätet i en mindre stad [2]. Arbetet utfördes på Elektroskolan EES vid ICS, avdelningen för Industriella Informations- och Styrsystem. Modellen av ett vindkraftverk som beskrivs i denna rapport kan med mindre modifieringar användas till att utöka tidigare nämnda modell.

A. Syfte

I det här projektet utvecklas en metodik för att skapa en ny komponent till ICS simuleringsplattform, i form av ett virtuellt vindkraftverk. Denna ska ha en reglerbar vindhastighet som indata, vilket exempelvis kan ställas in från historiskt uppmätta vinddata. Komponenten ska sedan utföra beräkningar på indatat och mata ut avgiven aktiv effekt samt förbrukad reaktiv effekt, i enlighet med hur ett kommersiellt vindkraftverk ska bete sig. Innan en vindkraftverksmodell kan skapas måste ett mål för modellen fastställas. I detta projekt valdes det att modellera ett verkligt vindkraftverk av typen E-44 från företaget Enercon [3]. Detta vindkraftverk har en märkeffekt på 900 kW vilken den når vid vindhastigheter större än eller lika med 16 m/s. Det här kommer förklaras närmare i II. *A. Vindkraftverk*. För att implementera simuleringsmodellen används Arduino och Raspberry Pi, som är billiga, kommersiella och allmänt tillgängliga (COTS) enkortsdatorer. Hur simuleringsmodellen implementerats i hårdvaran förklaras i II. *B. Simulink* och II. *C. Raspberry Pi*, där hårdvarans funktion-

alitet även förklaras närmare. Komponenten, som hädanefter kommer kallas vindkraftsmodulen, måste kunna göra beräkningar i realtid. Eftersom resten av systemets simuleringar sker i realtid är det viktigt att även vindkraftsmodulen arbetar i realtid, för att undvika felaktiga resultat. Det här kommer detaljeras noggrannare i II. *D. Realtidsanpassningar*.

B. Rapportens utformning

I kapitel II. *A. Vindkraftverk* ges teorin bakom ett vindkraftverk, samt hur denna teori implementerats kommersiellt och i Simulink. Efter detta följer hur modellen konverterats till hårdvara och hur programvaran samt hårdvaran modifierats för att uppfylla specifikationen. Kapitel III. RESULTAT detaljerar resultatet och tester som gjorts för att verifiera det. Kapitel IV. DISKUSSION och V. SLUTSATS diskuterar resultatet och redogör för tillägg, ändringar och brister i den färdiga vindkraftsmodulen.

II. METOD OCH TEORI

A. Vindkraftverk

I Fig. 1 nedan visas förhållandet mellan hastigheten på vinden som träffar vindkraftverkets rotorblad och den aktiva effekten som produceras för det kommersiella vindkraftverket Enercon E-44. Målet är att hårdvarumodellen ska följa denna kurva så nära som möjligt.

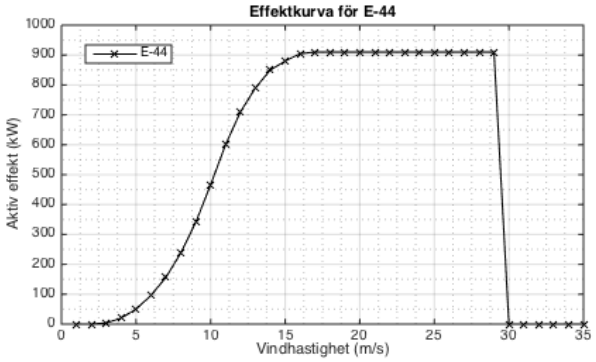

Fig. 1. Effektkurva för Enercon E-44 med 900 kW märkeffekt [3], vilken vill följas.

Som kan ses i Fig. 1 överstiger effekten aldrig 910 kW. Anledningen är att vindhastigheter över 16 m/s ger turbinhastigheter som ligger över generatorns optimala arbetspunkt. Vid höga vindhastigheter vinklas därför rotorbladen av en inbyggd mekanism så turbinens rotationshastighet inte blir för hög. Om vindhastigheten når 30 m/s, vilket skulle motsvara svår storm, stängs vindkraftverket av automatiskt och den aktiva effekten går ner till 0 W, vilket även det ska implementeras i modellen.

Den färdiga modellen ska endast ta vindhastigheten som inparameter men mata ut både aktiv och reaktiv effekt. Den tillgängliga effekten i vinden är beroende av vindhastigheten [4] med sambandet

$$P_W = c_p A_R \frac{\rho}{2} v^3, \tag{1}$$

där c_p är vindkraftverkets koefficient för prestanda, A_R är rotorbladens sveparea, ρ är luftdensiteten och v är vindhastigheten. Anledningen till att egenskaperna hos ett tredjegradspolynom inte är så prominenta i effektkurvan för Enercon E-44 beror på den variabla koefficienten c_p som skär ner kurvan så den inte växer obegränsat. Detta görs för att leverera konstant effekt och skydda generatorn i vindkraftverket vid högre hastigheter [5]. Sveparean är den area som täcks av rotorbladen när de roterar, dvs. en cirkelskiva med vindkraftverkets noskonsarea subtraherad.

En av de vanligaste typerna av vindkraftverk idag är dubbelmatad induktionsgenerator även kallad Doubly Fed Induction Generator (DFIG) [6]. En induktionsmaskin konsumerar reaktiv effekt, vare sig om den arbetar som motor eller generator, eftersom rotorfältet alltid ligger efter statorfältet i fas [7]. Den reaktiva effekten som matas ut ur vindkraftsmodulen bör därför vara negativ för modellen om denna är verklighetstrogen. Svårigheter att hitta information om hur mycket reaktiv effekt ett Enercon E-44 konsumerar medförde att kraven för modellen inte var speciellt höga.

B. Simulink

Verktyget som valdes för simulering var Simulink från MathWorks som är ett grafiskt påbyggnadsprogram till MATLAB för att skapa modeller och simulera system. Eftersom Simulink är ett grafiskt programmeringsspråk kopplas block ihop för att representera system medan koden kompileras och körs internt när simuleringen väl utförs. I programmets bibliotek finns både logiska elementära block och större sammansatta block, varför mycket efterforskning gjordes i förtid för att fastställa vilka block som är relevanta för användningsområdet. Dessa block kan jämföras med funktioner i andra programmeringsspråk eftersom de tar ett inargument, utför operationer på detta, och ger ett returvärde.

Det valdes att utgå från det färdiga blocket av en vindturbin av typen dubbelmatad induktionsgenerator (“Wind Turbine Doubly-Fed Induction Generator”). Det kommer inte förklaras om hur vindturbinen är uppbyggd i denna rapport, utan den kommer behandlas som ett block med parametrar specificerade senare i denna del. Blocket kan ta de två insignalerna “Wind (m/s)” och “Trip” (se Fig. 7). I “Wind (m/s)” ska, som namnet anger, vindhastigheten matas in mätt i meter per sekund. Målet är att vindkraftsmodulen ska ta in en extern input, därför kopplades ett sådant block dit. Koefficienten c_p från (1) implementeras i form av Simulinkblocket “Saturation” som begränsar vindhastigheten till 16 m/s. Om paralleller dras till ett verkligt vindkraftverk kan mättnaden ses som vinklingen av rotorbladen för höga hastigheter. Insignalen “Trip” fungerar som en på och avstängningsswitch till vindkraftsmodulen. När signalen sätts till 1 kommer vindkraftverket att stängas av. För att efterlikna Enercon E-44 kopplades därför ett komparatorblock in som sätter “Trip” till 1 om vindhastigheten är 30 m/s eller högre.

Likt en verklig asynkronmaskin måste vindturbinsmodellen matas med reaktiv effekt från ett trefas-nät. I Simulinkmodellen modelleras nätet med en trefas-källa på 25 kV som Fig. 7 visar. Nätspänningen måste därefter transformeras ned till en spänning som lämpar sig bättre för turbinen. En trefas Y-Yn-

transformator med omsättningen 1000:23 valdes till detta, vilket ger spänningen 575 V. För att minska den reaktiva effekten [6] som vindkraftverket drar från elnätet parallellkopplades även en trefas-kondensatorbank på 400 kvar in mellan transformatorn och turbinen.

1) Wind Turbine Doubly-Fed Induction Generator

Det tidigare nämnda blocket för vindturbinssimulering kommer hädanefter förkortas till WTDFIG. Användaren får stora möjligheter att manipulera blockets inställningar så att vindturbinen beter sig enligt specifikation. I det här projektet skulle vindturbinen ge en effekt-vindhastighet-kurva som så nära som möjligt efterliknar den hos Enercon E44. Den viktigaste parametern är vektorn "Tracking characteristic speeds" som ställer in fyra värden ABCD. Sedan ges en kurva mellan de olika värdena, där värdet på A-D kan avläsas från x-axeln (Fig. 2). Detta motsvarar turbinrotationshastigheten mätt i pu (per unit) vilket är den relativa hastigheten i förhållande till det maximala värdet. 1 pu motsvarar alltså maxhastighet. På y-axeln avläses motsvarande avgiven mekanisk effekt mätt i pu.

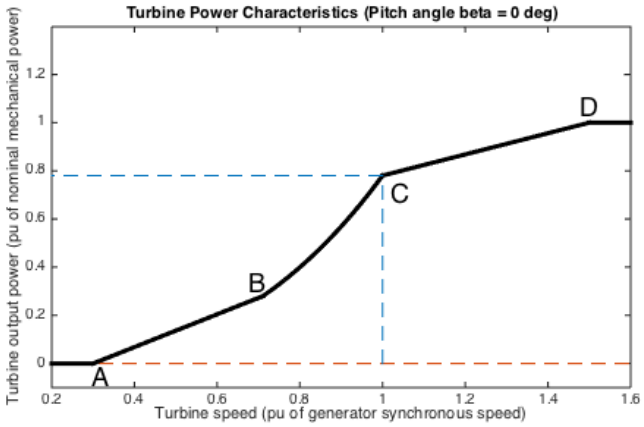

Fig. 2. Egenskaper hos vindturbinens utmatade effekt som funktion av turbinhastigheten.

Punkten A motsvarar vindkraftverkets "cut-in speed", d.v.s. vid vilken turbinrotationshastighet effekt ska börja produceras. Mellan punkt A och punkt B interpoleras en rät linje. Punkt B ger värdet där effektkurvan ska börja följa en given geometrisk ort, fram till punkt C. Dokumentation saknas om vilken ort som följs, men det kan antas att orten är baserad på (1) eftersom den visar ett beteende likt en tredjegradsfunktion. Punkt C används för att välja en referensvindhastighet, som här sattes till 12 m/s. Vid denna punkt ansätts även den avgivna elektriska effekten i pu (per unit) av den mekaniska. Empiriska tester visade att 71/91 pu gav ett lämpligt resultat. Mellan punkt C och punkt D interpoleras en rät linje. Punkt D ger den turbinrotationshastighet där 1 pu elektrisk effekt matas ut. För att finjustera kurvan sattes den elektriska nominella effekten till 870 kW istället för 900 kW. Detta behövde göras eftersom de linjära interpolationerna gav för låga effekter vid mellanhöga vindhastigheter. Nedan visas turbinhastighet-effektkurvan för [A B C D] = [0.3 0.71 1 1.5], som bör ge önskad vindhastighet-effektkurva i den färdiga simuleringsmodellen. Turbinhastigheten är dock inte direkt proportionell mot vindhastigheten eftersom inbyggda regulatorer ser till att den upptagna vindeffekten minskas för högre vindhastigheter. Alltså ger Fig. 2 endast en indikation om hur vindkraftverkets simulering kommer bete sig.

WTDFIG-blocket har ett antal utsignaler som väljs med hjälp av en bussväljare. De signaler som var relevanta för projektet var den aktiva respektive reaktiva effekten, vilka därför valdes. En oönskad egenskap hos vindturbinblocket är dock att dessa effekter matas ut som pu, dvs. nerskalade med märkeffekten 900 kW för att anta ett värde mellan 0 och 1 för effekter mellan 0 och 900 kW. Omvandling till enheterna watt respektive voltampere reaktiv gjordes med enkelhet med hjälp av en förstärkning på $9 \cdot 10^5$ gånger. Målet var för vindkraftsmodulen att mata ut både den aktiva och reaktiva effekten, men kodgeneratorn i Simulink som senare kommer användas klarar bara av att returnera en enda utsignal. Lösningen på detta problem var att endast ha den aktiva effekten som extern utsignal och istället sätta en markör för den reaktiva effekten. Markören har som syfte att underlätta identifiering av den reaktiva effekten i den senare genererade C-koden. En terminator sattes i slutet på utsignalen för den reaktiva effekten för att undvika problem med icke inkopplade ledningar vid kodgenereringen.

2) Simulinkmodell till C-kod

Simulinkmodellen använder en proprietär kod som anropar MATLAB-bibliotek och inbyggda funktioner. För att kunna köra modellen på annan hårdvara, i synnerhet hårdvara som inte klarar att köra MATLAB, måste modellen paketeras om till allmän kod. För att göra detta har Simulink en inbyggd kodgenerator, som kan konvertera modellen till ett antal olika programmeringsspråk. C valdes som språk för det här projektet, eftersom det möjliggör maskinnära programmering på låg nivå, med liten overhead och fullständig kontroll över hårdvarans beteende [8].

Kodgeneratorn ställdes in för Embedded Real Time (ERT), för att skapa kod optimerad för inbäddade realtidssystem. Koden ska ta in vindhastigheten som ett värde, och mata ut effekterna som värden. Ändringar gjordes i kodgeneratorns gränssnittshantering för att möjliggöra detta, då standardinställningen använder pekare för in- och utsignal. Eftersom modellen innehåller beräkningar på t.ex. komplexa tal, flyttal och kontinuerliga tal behöver stöd för dessa väljas till. Simulink använder numeriska metoder för att lösa differentialekvationer, vilket i den här modellen görs med Bogacki-Shampine's metod med steglängd 0,001 s. Exempel på differentialekvationer som behöver lösas är de som modellerar transienterna som orsakas av trögheten i turbinen, dock är dessa ekvationer inte nödvändiga att känna till då Simulink både ställer upp och löser dem. Även om differentialekvationer kan lösas effektivare med varierbar steglängd måste konstant steglängd användas eftersom kodgeneratorn inte kan framställa kod som både arbetar i realtid och beräknar icke-konstanta steglängder [9]. Koden kunde genereras efter att ovanstående inställningar konfigurerats. Detta skapar ett flertal header-filer och C-program som definierar funktioner, datatyper och inställningar. Alla dessa filer anropas i sin tur

från tre funktioner, som beskrivs nedan. Observera att den genererade C-koden definierar egna datatyper som skiljer sig från de inbyggda i C. Exempelvis används datatypen `real_T` för att definiera variabler som består av reella tal.

Dessutom har en funktion skrivits och lagts till för att läsa det binära 5-bitarstalet från Raspberry Pi:s GPIO-pins. Innan funktionen anropas måste `wiringPiSetupGpio()` anropas för att förbereda GPIO. Utförliga beskrivningar av nämnda funktioner finns att hitta i Tabell 1 i APPENDIX.

I filen `Finalmodel_main.c` körs ovanstående funktioner enligt följande tillvägagångssätt:

1. Programmet initieras för modellen och wiringPi
2. Minnet låses och prioritering sätts högt för att undvika avbrott. Mer om detta i *2)*.
3. Realtidsaspekten utförs genom att logga tiden och "stalla" programmet tills 1 ms har passerat, inklusive exekveringstiden för en iteration. Mer om detta i *2)*.
4. Vindhastighetsvärde hämtas, och momentaneffekten beräknas. Värden skrivs på skärmen.
5. Åter till steg 3 tills programmet avbryts.

C. Raspberry Pi

Valet av hårdvara föll på Raspberry Pi Model B, en billig enkortsdator kapabel till att köra fullständiga kommersiella operativsystem. Den är utrustad med en Broadcom BCM2835 SoC (System-on-Chip), vilket innebär att processorn (enkärnig 700MHz ARM1176JZF) och grafikkretsen (Broadcom VideoCore IV) är integrerade på samma krets. 512 MiB SDRAM är monterat ovanpå SoC:n. Datorn saknar lokal lagring, så operativsystem och programvara installeras på ett externt SD-kort som monteras i en dedikerad port. För kommunikation och anslutning till externa enheter har Raspberry Pi två USB 2.0-portar, en Ethernetport samt 26 stycken GPIO-pins.

Innan Raspberry Pi kan köra simuleringsprogramvaran behöver den ett operativsystem. I det här projektet användes Raspbian OS, en modifierad version av Linuxdistributionen Debian [10]. Dessutom patchades Linuxkärnan med RT-Preempt, som möjliggör äkta realtidskörning av programvara [12].

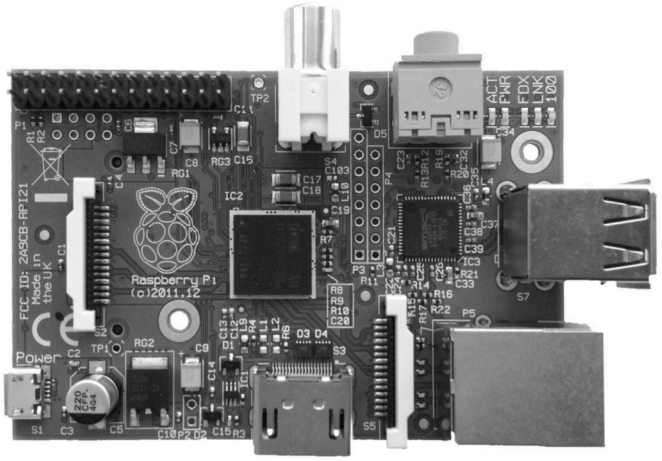

Fig. 3. Raspberry Pi Model B, hårdvaruplattformen för detta projekt.

D. Realtidsanpassningar

En utvecklare som önskar köra programvara i realtid stöter på ett flertal problem. För det första måste programmeringsspråket väljas med omsorg, eftersom högnivåspråk som exempelvis Python har oberäknelig "garbage collection" [11]. Det här orsakar jitter, "mikrofrysningar" som avbryter exekveringen i upp till ett flertal millisekunder. Realtidsapplikationer körs alltså med fördel i programmeringsspråk där man har fullständig kontroll över hårdvara och mjukvara, exempelvis C eller Assembly.

1) Operativsystemet

I rapporten "*A Comparison of Scheduling Latency in Linux, PREEMPT_RT, and LITMUS^{RT}*" [12], skriver Björn Brandenburg och Felipe Cerqueira om så kallad "scheduling", fördröjningar orsakade av växlingar mellan trådar i Linux. Programmet cyclictest används för att jämföra prestandan hos en ren Linux 3.0-kärna och de anpassade kärnorna PREEMPT_RT samt LITMUS^{RT}. Från testerna framgår det att den rena Linuxkärnan stundvis ger fördröjningar på flera millisekunder samtidigt som I/O-baserade processer körs, medan de två modifierade kärnorna ger en värsta fall-fördröjning på enstaka mikrosekunder. Av denna anledning patchades Raspbian-systemet med den skräddarsydda Linuxkärnan PREEMPT_RT 3.12.36-rt50+. Detta möjliggör dessutom större kontroll över interrupts, eftersom de hanteras i kärntrådar istället för i hårdvara [13]. Utöver detta ökades processorns klockfrekvens till 800 MHz, för ökad prestanda.

2) Programkoden

a) Minne

För att låsa minnet och garantera att programmet inte växlas till virtuellt minne anropas den inbyggda C-funktionen `mlockall(MCL_CURRENT|MCL_FUTURE)`.

b) Timing

I koden anropas funktionen `sleep_until()`, som följer realtime-funktionen `CLOCK_MONOTONIC` och ser till att programmet väntar tills 1 ms passerat från att föregående iteration har startat.

c) Prioritering

För att programmets tråd inte ska flyttas undan och "stall-las", då operativsystemet försöker hantera andra processer, måste programmet ställa in sin prioritering högt. Prioriteringen sattes till nivå 30 med `pthread_setschedparam()`.

d) Kompilering

Realtidsfunktionerna, exempelvis `CLOCK_MONOTONIC`, i C-programmet kräver att realtidsbiblioteket rt länkas vid kompilering. Dessutom länkades bibliotekets wiringPi för att infoga stöd för GPIO. Optimering användes inte för att undvika eventuella problem med cachade variabler som inte uppdateras kontinuerligt. Det här hade kunnat göras om alla dynamiska variabler deklarerats som `volatile`, men denna ändring ansågs för tidskrävande eftersom programmet totalt innehåller över 3000 rader kod. För kompilering användes

GCC (GNU Compiler Collection). Det fullständiga Linux-kommandot för att kompilera koden finns i APPENDIX. Kompileringen skapade den exekverbara filen Windpower. Alla genererade filer inkluderades i `Finalmodel_main.c` och placerades i samma mapp innan kompilering. Github-länk till såväl autogenererad som egenskriven kod finns tillgänglig i APPENDIX.

III. RESULTAT

A. Simuleringar

I Fig. 7 visas den färdiga Simulinkmodellen. Detta är modellen som körs i form av C-kod på Raspberry Pi.

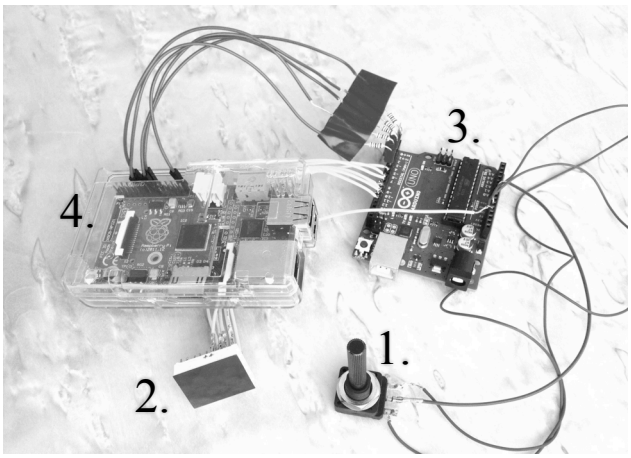

Fig. 4. Fysisk vindkraftverksmodell bestående av vridpotentiometer (1), sjusegmentsdisplay (2), A/D-omvandlare på Arduino (3), samt en Raspberry Pi (4).

Den slutgiltiga vindkraftsmodulen visas i Fig. 4 ovan. Produkten består av en vridpotentiometer, en Arduino, en sjusegmentsdisplay, en Raspberry Pi, samt en datorskärm. Resultatet från simuleringar för vindhastigheter från 1 m/s till 35 m/s visas i Fig. 5. I figuren är även effektkurvan för Enercon E-44 inritad då det var denna som försöktes efterliknas. Värdena på den aktiva effekten hämtades efter 30 sekunders simulering då utsignalens transienter avklingat.

Även den kompletta fysiska vindkraftsmodulen testades med vridpotentiometern satt till vindhastigheter från 0 m/s till 31 m/s. Den aktiva effektens värde som visas i Fig. 6 erhölls då modellens utsignal nått stabilt tillstånd.

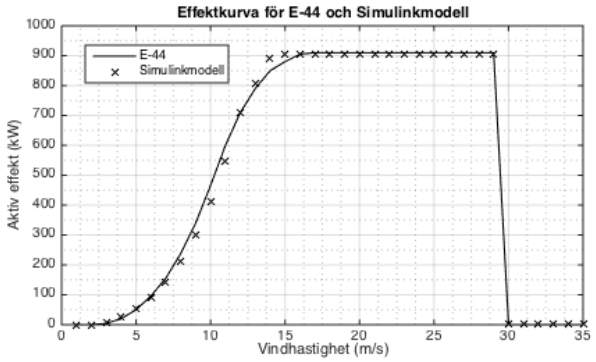

Fig. 5. Jämförelse mellan effektkurvor för Enercon E-44 (heldragen linje) och projektets modell i Simulink (markerat med "x").

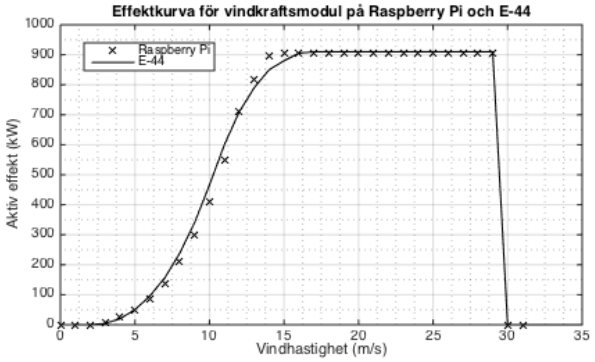

Fig. 6. Effektkurva för färdig fysisk vindkraftsmodul (markerat med "x") jämfört med Enercon E-44 (heldragen linje).

B. Modellens gränssnitt

Den färdiga modellen tar in vinddata som en extern insignal, vilket gör att en utomstående källa behövs för att ställa in vindhastigheten. Detta designval gjordes för att möjliggöra snabba källbyten av vinddata utan att behöva modifiera koden och/eller Simulinkmodellen. Då vindkraftsmodellen är designad för att stängas av vid vindhastigheter större än eller lika med 30 m/s, behöver den inte kunna simulera högre vindhastigheter än så. Därför bedömdes det tillräckligt att ta in en 5-bitars binärt tal via ett parallellt gränssnitt implementerat i Raspberry Pi:s GPIO-pins. Detta möjliggör vindhastigheter mellan 0 till 31 m/s, med upplösning 1 m/s.

1) Insignal

För projektets implementering tillverkades en extern A/D-omvandlare, som tar in en analog spänning levererad från en referensspänning via en 220 kΩ linjär vridpotentiometer. Användaren måste alltså manuellt ställa in vindhastigheten via vridpotentiometern. A/D-omvandlaren implementerades med C-kod som kördes på en Arduino Uno, en mikrokontroller baserad på ATmega328. Denna har en analog inport som relativ spänning kan hämtas från. Ett värde mellan 0 och 1023 returneras, där 0 motsvarar 0 V och 1023 motsvarar 5 V. Detta värde konverteras ned i programvaran till ett 5-bitars binärt tal, som skrevs på Raspberry Pi:s fem GPIO-pins via korta GPIO-sladdar. Dessutom visas talet (0-31) på en digital sjusegmentsdisplay, ansluten till Arduinos GPIO-pins. Koden till A/D-omvandlaren finns i APPENDIX.

2) Utsignal

Vindkraftsmodellens framräknade resultat skrivs i konsolen på skärmen var 100:e millisekund. Varje ny rad innehåller momentanvärden för framställd aktiv effekt, förbrukad reaktiv effekt samt vindhastigheten. Föregående värden bevaras på skärmen tills de ersatts av nya, för att få en uppfattning om förändringshastigheten.

C. Realtidstest

För att verifiera att programmet körs i realtid gjordes ett test. En ny version av programmet skrevs, kallad `Finalmodel_timetest.c`. Denna modifierades så att while-loopen begränsades till exakt 300 000 iterationer, istället för att iterera obegränsat som i standardversionen.

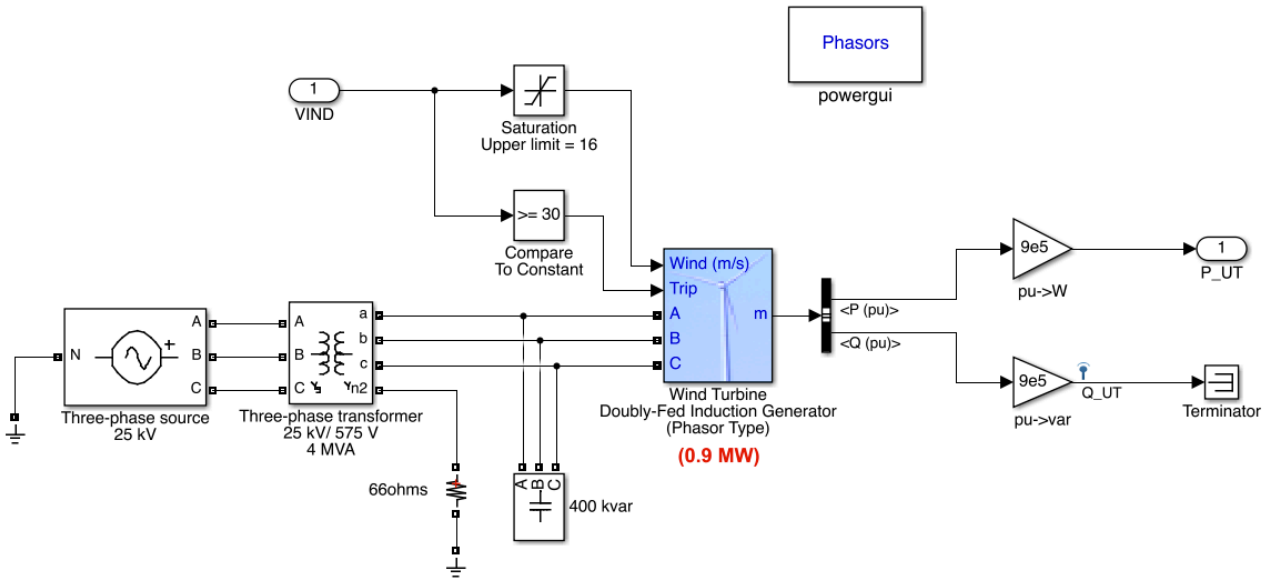

Fig. 7. Färdig modell av vindkraftverk i Simulink.

Då funktionen `sleep_until()` ska se till att varje iteration tar exakt en millisekund, bör detta resultera i att programmet kör i exakt 5 minuter.Det kompilerade testprogrammet döptes till Timepower. För att genomföra testet skrevs det kombinerade kommandot `>> date; sleep 2; ./Timepower; date` i linuxkonsolen. Kommandot skriver nuvarande tid, väntar två sekunder, kör programmet och matar sedan ut tiden igen. I konsolen skrevs tiderna 19:46:21 följt av 19:51:24. Eftersom kommandot väntade i två sekunder var den önskade sluttiden 19:51:23. Programmet har alltså totalt 1 sekund fördröjning efter ett test på fem minuter, vilket motsvarar ett tidsfel på 0,3 %.

IV. DISKUSSION

Som resultaten i Fig. 5 visar följer Simulinkmodellens effektkurva (markerad med "x") den given av Enercon E-44 (heldragen linje) till stora delar. Det största felet hittas vid vindhastigheten 10 m/s där E-44 producerar 466 kW, vilket är nästan 56 kW mer än vad modellen ger. Empiriska tester visade dock att detta var en uppoffring som var tvungen att göras för att få bäst anpassning till effektkurvan i flest punkter.

Med bättre källmaterial skulle modellen i Simulink kunna anpassas för att få bättre resultat. Eftersom Enercon E44 är en kommersiell produkt är det svårt att få tag på ritningar och data för komponenterna. Med hjälp av detta skulle man kunna konstruera en modell från grunden, som exakt motsvarar det verkliga vindkraftverket.

Om effektkurvan för den fysiska vindkraftsmodulen på Raspberry Pi (Fig. 6) studeras kan det ses att den är mer eller mindre identisk med den för Simulinkmodellen, vilket var önskat och förväntat resultat. Skillnaden är endast intervallet inom vilket modellen arbetar. Då det valdes att använda 5 bitar att representera vindhastigheten då denna förs över mellan Arduino och Raspberry Pi kan den fysiska modellen bara utföra simuleringar för vindhastigheter från 0 m/s till 31 m/s. Med 6 bitar skulle 64 olika hastigheter kunna simuleras, exempelvis alla heltal från 0 till 63 eller från 0 till 31,5 med steget 0,5. Anledningen till att detta inte implementerades var helt enkelt brist på utpins på Arduino. Av de 12 pins som lämpar sig för digital utsignal på Arduino krävs 7 pins per siffra som visas på sjusegmentsdisplayen. En lösning skulle vara slopning av displayen på bekostnad av ökad osäkerhet under utförda tester.

Om mindre modifikationer görs i C-koden som körs på Raspberry Pi kan den aktiva effekten skrivas som en pulsbreddsmodulerad spänning på GPIO 18. Värdet 0 motsvarar där $P = 0$ W och värdet 1 (3.3 V) motsvarar $P \geq 900$ kW. Vindkraftsmodulen kan då kopplas ihop med större modeller, exempelvis tidigare års kandidatexjobb på KTH [2] där elnätet i en mindre stad modellerades. Man använde då en 12 V växelspänning på 50 Hz. För att kunna ansluta vindkraftsmodulen till detta nät krävs analog elektronik i form av en D/A-omvandlare och reläer. Denna måste konstrueras så att 900 kW motsvarar maximal levererad effekt. Modifikationer i Simulink skulle kunna få vindmodulen att mata ut en effekt i trefas, vilket görs av verkliga vindkraftverk.

Konstruktionen i sin helhet är relativt ineffektiv kostnadsmässigt. Kostnaden för en Arduino One, Raspberry Pi Model B och ett 8 GB SDHC-kort uppgår i skrivande stund till nästan 630 SEK[1]. Majoriteten av Raspberry Pi:s funktioner används inte under körning, och faktumet att Raspbian är ett fullskaligt Linux-OS har visat sig vara ett hinder snarare än en möjlighet. Detta eftersom extra åtgärder var nödvändiga för att säkerställa realtidskörning.

För att minska tillverkningskostnaden kan man använda sig av en mikroprocessor som exempelvis Arduino för själva modellen. Detta är även vad som nu används för input, vilket är synnerligen ineffektivt eftersom Arduinon endast fungerar som A/D-omvandlare. Om modellen körs på Arduino skulle inmatningen kunna ske på samma krets, istället för att behöva hämtas externt. Raspberry Pi ska dock inte omedelbart

[1] Priser tagna från www.kjell.com för Raspberry Pi Model B (199 SEK), Arduino Uno (279 SEK) och 8 GB SanDisk SDHC-kort (149,90 SEK) den 27 april 2015 klockan 15:59.

avfärdas som simuleringsplattform eftersom den har mycket mer avancerade möjligheter för kommunikation över exempelvis Ethernet i ett simuleringsnätverk. Det skulle vara relativt enkelt att modifiera koden så att den skickar värdet som en UDP broadcast, varpå data kan användas av andra komponenter i nätverket.

A. Kommentar till realtidstest

Realtidstestet gav ett gott resultat, och den totala fördröjningen bör inte påverka simuleringsresultaten. Då simulationen når stabilt tillstånd efter cirka 30 sekunder blir fördröjningen mindre än en tiondels sekund. Det är även möjligt att fördröjningen kommer bli ännu mindre märkbar vid längre körningar, eftersom testet även mätte den tid som krävdes för att initiera och ladda programmet första gången. Ett bättre test skulle antingen genomföra testet under en längre tid eller utföra mätningen inuti själva programmet, i samband med att iterations-loopen startat.

V. SLUTSATS

En fysisk modell av ett vindkraftverk har skapats i Simulink och implementerats som realtidsapplikation på en Raspberry Pi. Modellen klarar av att ta indata i form av en vindhastighet och beräkna den aktiva samt reaktiva effekt som ett verkligt vindkraftverk producerar eller konsumerar vid denna hastighet. Effekterna samt vindhastigheten skrivs ut på en display med uppdateringsintervall på 100 ms. Insignalen kan regleras med en vridpotentiometer och effekten beräknas i realtid. Då information om verkliga vindkraftverks reaktiva effektkonsumtion sällan publiceras utelämnades denna i resultatet.

Med hjälp av samma verktyg som användes i detta projekt kan emellertid andra källor och laster till ett elnät modelleras. Som nämndes i introduktionen till denna rapport är miljöberoende källor ofta svårare att skapa verklighetsriktiga modeller av då de inte är fullt reglerbara. Om simuleringar med sådana källor ska göras krävs verkliga indata i form av exempelvis vindhastighet eller mängden solljus under ett dygn för att kunna studera påverkan på elnätet.

Modellen kan, efter mindre fysiska och kodmässiga anpassningar, användas i nästan vilket simuleringsnät som helst. Eftersom modellen beter sig som en “black box”, där endast insignal och utsignal är av värde, behöver modifikationerna endast involvera modellens gränssnitt. Man behöver alltså inte modifiera detaljerna för hur simuleringen går till för att anpassa modellen till nya nät.

SLUTORD

Vi vill passa på att tacka Daniel Brodén, som kunde hoppa in som handledare eftersom Nicholas Honeth fick förhinder. Tack riktas även till Davood Babazadeh, Hassan Fidai, och Mehrdad Kazemtabrizi för deras hjälp med tekniska aspekter.

REFERENSER

[1] M. Thorsén. (27 januari, 2015). *Elanvändning* [Online]. Hämtat: 27 april, 2015. Tillgängligt: http://www.svenskenergi.se/Elfakta/Elanvandning

[2] G. Falk Olson, och V. Gliniewicz, *Fysisk modellering av IEEE 14-bus test system*, Industriella Informations- och Styrsystem, KTH, Stockholm, 2014.

[3] (juli, 2010) *ENERCON Wind energy converters* [Online]. Hämtat 12 maj, 2015. Tillgängligt: http://www.enercon.de/p/downloads/EN_Productoverview_0710.pdf

[4] S. Heier, *Wind Energy Conversion Systems*, Chinchester, England: John Wiley & Sons Ltd, 1998.

[5] A. Miller, E. Muljadi, och D. S. Zinger, “A variable speed wind turbine power control,” *IEEE Transactions on Energy Conversion*, vol. 12, no. 2, 1997.

[6] E. Kilander, och N. Modig, *Frekvensderivataskydd för kraftsystemet*, Mastersexamensarbete, Industriella Informations- och Styrsystem, KTH, Stockholm, 2014.

[7] A. Alfredsson, K. A. Jacobsson, A. Rejminger et al., *Elkrafthandboken*, Stockholm: Liber AB, 2002.

[8] B. W. Kernighan, och D. M. Ritchie, *The C Programming Language*, 1st ed.: Prentice Hall, 1978.

[9] *Choose a Solver - MATLAB & Simulink - MathWorks Nordic* [Online]. Hämtat: 27 april, 2015. Tillgängligt: http://se.mathworks.com/help/simulink/ug/choosing-a-solver.html

[10] W. M. Traynor. (29 november, 2014). *Raspbian* [Online]. Hämtat: 27 april, 2015. Tillgängligt: http://elinux.org/Raspbian

[11] B. Croston. (15 februari, 2015). *RPi.GPIO 0.5.10* [Online]. Hämtat: 27 april, 2015. Tillgängligt: https://pypi.python.org/pypi/RPi.GPIO/0.5.10

[12] F. Cerqueira, och B. B. Brandenburg, *A Comparison of Scheduling Latency in Linux, PREEMPT RT, and LITMUSRT*, Max Planck Institute for Software Systems, Berlin, 2013.

[13] L. Fu och R. Schwebel. (7 mars, 2014). *RT PREEMPT HOWTO* [Online]. Hämtat: 27 april, 2015. Tillgängligt: https://rt.wiki.kernel.org/index.php/RT_PREEMPT_HOWTO.

APPENDIX

TABELL I
FUNKTIONER I C FÖR VINDKRAFTSMODELL PÅ RASPBERRY PI

Funktionsprototyp	Förklaring	Timing	Argument	Returnerat värde	Headerfil
`void Model-Copy2_initialize(void)`	Initierar modellen i programkoden	Ska anropas exakt en gång	Inget	Inget	ModelCopy2.h
`real_T Model-Copy2_custom(real_T arg_VIND)`	Utför en enstaka iteration i beräkningen av modellen. Anropas inuti funktionen `float rt_OneStep(int)`, som möjliggör inaktivering av interrupts under exekvering.	Ska anropas exakt 1000 gånger per sekund för steglängd 0,001. Timingen är kritisk för realtidsaspekten.	`real_T arg_VIND` (momentanvärdet för vindhastig-heten)	`real_T` (momentanvär-det för aktiv effekt) Observera att den reaktiva effekten hämtas manuellt med `ModelCopy2_B.QUT` efter att funktionen anropats, eftersom funktionen bara kan returnera ett enstaka värde.	ModelCopy2.h
`int GetWind(void)`	Hämtar det binära talet från GPIO och konverterar det till ett tal i bas 10	Anropas i samband med `rt_OneStep` för att hämta ett nytt vindhastighetsvärde.	Inget	`int Arg_VIND`	wiringPi.h
`void Model-Copy2_terminate(void)`	Avbryter modellen.	Ska anropas exakt en gång	Inget	Inget	ModelCopy2.h

Tabellen visar de fyra viktigaste funktionerna i progammet.

Kommando för kompilering av vindkraftmodellens C-program:
>> Gcc Finalmodel_main.c -DRT -o Windpower -lrt -lwiringPi -lm

Github-länk till filer med all C-kod som krävs för att köra modellen på Raspberry Pi: https://github.com/b12015/b1windpower
Läs README.txt för instruktioner om hur koden ska kompileras.

Kod som körs på Arduino för A/D-omvandling:

```
int c, k;
void setup()
{
  pinMode(2, OUTPUT);
  pinMode(3, OUTPUT);
  pinMode(4, OUTPUT);
  pinMode(5, OUTPUT);
  pinMode(6, OUTPUT);
  pinMode(A0, INPUT);
}

void loop()
{
  int x = analogRead(A0);
  int t = x;
  if (t & 1)
  {
    x++;
  }
  x = x/32;      // max 1024 -> max 32
  for (c = 4; c >= 0; c--)
  {
    k = x >> c;
    if (k & 1)
      digitalWrite(c+2, HIGH);
    else
      digitalWrite(c+2, LOW);
  }
```

Länk till beskrivande screencast som förklarar hur Simulinkmodellen är gjord:
https://www.youtube.com/watch?v=_5QEof5HXF8
Observera att ändringar gjorts i modellen sedan videon skapades. Blocket WTIG som användes i videon byttes senare ut mot WTDFIG. Parametrar till det nya blocket beskrivs under Metod och Teori.