HÖGSKOLAN
I HALMSTAD

Elektroingenjör 180hp

EXAMENSARBETE

Simulator för värmepump

Johan Persson och Kushtrim Veseli

Examensarbete 15hp

Halmstad 2017-01-18

Förord

Vi vill tacka vår handledare Stefan Byttner för all värdefull hjälp under projektet. Vi vill även rikta ett stort tack till EasyServ som kom med idén och har hjälpt oss för att få fram ett färdigt testsystem.

Sammanfattning

Rapporten beskriver projektet som genomfördes åt EasyServ, vars verksamhet grundar sig på att övervaka och diagnostisera värmepumpar. Deras önskan med projektet var att utveckla ett mer effektivt testsystem som gör det möjligt att på ett enkelt sätt simulera en värmepumps beteende. Syftet med detta var att möjliggöra effektivare tester med bättre exakthet för EasyServs produkt. Detta förenklar kvalitetssäkringen av mjukvaran för deras produkt.

De stora problemen som projektet stod inför var hur testsystemet skulle designas samt hur simuleringen av värmepumpens temperaturgivare skulle ske för att efterlikna en värmepump. Ett annat frågetecken var vilket kommunikationsgränssnitt som lämpade sig bäst för testsystemet.

Metoden för att konstruera testsystemet grundade sig i användning av simuleringstekniken Hardware-in-the-loop(HIL). Projektet delades därmed upp i delsystemen Elektronikkonstruktion och Programmering. Under Elektronikkonstruktion genomfördes designvalen och konstruktionen av testsystemet. Delsystemet Programmering behandlar utvecklingen av simuleringsprogrammet.

Resultatet blev ett testsystem bestående av en Raspberry Pi och ett tillverkat I/O-kort, där kommunikationen sker via I^2C. I/O-kortet består av åtta digitala potentiometrar som används för att efterlikna värmepumpens temperaturgivare. Simuleringsprogrammet som har utvecklats i Raspberry PI använder programmeringsspråket Python.

Slutsatsen är att projektets testsystem ger EasyServ en bra grund för att kunna testa diagnosverktyget på ett enkelt sätt. En potentiell och önskvärd vidareutveckling som ger testsystemet ännu större användbarhet för Easyserv är att kunna återkoppla kompressorsignalen.

Abstract

The report describes the project carried out for EasyServ, whose business is based on the monitoring and diagnosis of heat pumps. Their desire for the project was to develop a more efficient test system that makes it possible to easily simulate a heat pump behavior. The purpose of this was to enable more efficient tests with better accuracy for EasyServs product. This simplifies the quality assurance of the software for their product.

The main problems the project were facing was how the test system should be designed and how the simulation of the heat pump's temperature sensors would be to mimic a heat pump. Another question mark was which communication interface was best suited for the test system.

The method for constructing the test system was based on the use of the simulation technology Hardware-in-the-loop (HIL). The project was thus divided into subsystems Electronics Design and Programming. In Electronic design the decisions regarding the design were taken and the construction of the test system was made. The Programming subsystem deals with the development of the simulation program.

The result was a test system consisting of a Raspberry Pi and a manufactured I/O board, where communication takes place through I^2C. The I/O board has eight digital potentiometers which are used to simulate the heat pump's temperature sensors. The simulation program developed in Raspberry Pi uses Python as programming language.

The conclusion is that the project's test system provides a good basis for EasyServ to test their diagnostic tool in a simple way. A potential and desirable development that makes the test system even more useful for EasyServ would be to feedback the compressor signal.

Innehåll

1 Inledning

I dagens samhälle är det möjligt att värma upp de svenska husen på flera olika sätt. Ett sätt som blir mer och mer populärt med tiden är användning av olika typer av värmepumpar[1]. Värmepumpens popularitet har en del bakomliggande faktorer såsom bland annat dess effektivitet av elförbrukning då endast en fjärdedel av den värmeenergi som utvinns kommer från el-energi[1]. En annan faktor är dess goda miljöpåverkan[2], då lagrad solenergi används i processen till färdig värme. De här faktorerna kan och eftersträvas till att bli större. Genom att övervaka och analysera beteendet från flera värmepumpar av samma typ kan man hitta mönster. De mönster som identifieras kan bl.a. leda till att tidigt upptäcka fel som finns i värmepumpen och hur de uppkom. Det här gör det enklare att optimera driften, förlänga livslängden och minska på slitaget på värmepumpar.

Ett företag som ägnar sig åt verksamheten med övervakning av värmepumpar är EasyServ AB. EasyServ är beläget i Halmstad där deras verksamhet grundar sig på att utveckla och marknadsföra ett diagnosverktyg för värmepumpar. Syftet med diagnosverktyget är effektivisering av service och drift av värmepumpar. Deras kunder är återförsäljare/ombud för de större tillverkarna som exempelvis Nibe och IVT.

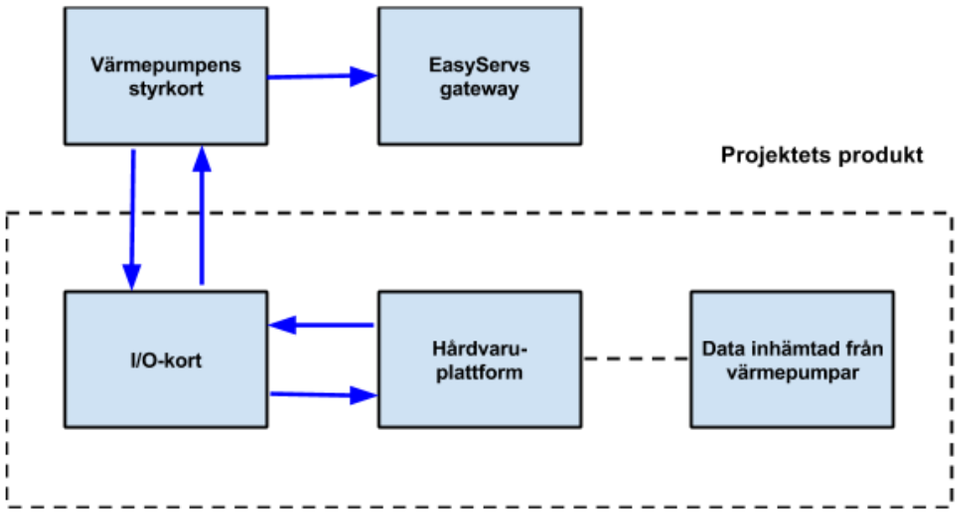
I dagsläget är EasyServs största problem att åstadkomma ett effektivt testsystem för deras kvalitetssäkring. En stor anledning till detta är att värmepumpar är dyra att köpa in och utrymmeskrävande. Detta medför att EasyServ inte har möjlighet att ha värmepumpar i deras labb. Lösningen som finns vid testning idag är användning av värmepumpens styrkort, tillhörande I/O-kort och diverse elektronik för simulering av värmepumpens beteende. Elektroniken som används är bl.a. mekaniska potentiometrar som ställs in manuellt eller vanliga NTC-motstånd. Denna lösningen gör det väldigt svårt att efterlikna en värmepump främst eftersom att många signaler hanteras samtidigt. Detta medför en opålitlig och tidsödande process.

EasyServ vill kringgå problemet genom konstruktion av ett mer effektivt och pålitligt testsystem som är digitalt. För att åstadkomma detta kommer projektet bestå av att konstruera ett system som på ett bra sätt simulerar en värmepumps beteende. Simuleringen ska bygga på användningen av historik från befintliga värmepumpars temperatur-givare.

1.1 Syfte

Syftet med projektet är att utveckla ett effektivt testsystem som ökar möjligheten för EasyServ att snabbare och bättre kunna kvalitetssäkra sitt diagnosverktyg. Diagnosverktyget kan då analysera och övervaka värmepumpar mer exakt, vilket gör att värmepumparna används mer optimalt. Ett annat syfte är underlättande av testning

då man undviker ex. inköp av värmepumpar som medför stora kostnader och är utrymmeskrävande.Detta undviker även möjligheten att en värmepump förstörs under test. Figur 1 visar EasyServs önskan av testsystemets design.



Figur 1: EasyServ önskar ett system som använder historik från befintliga värmepumpars temperaturgivare. Utifrån historiken används en hårdvaruplattform för simulering. De simulerade temperaturvärdena skickas till I/O-kortet som omvandlar signalen till en signal som styrkortet förstår. Styrkortet tolkar värdena och återkopplar tillbaka vid behov när kompressor eller växelventil ska sättas på/av.

1.2 Avgränsningar

Projektet avgränsas från att simulera tryckgivarna i värmepumpen. Avgränsningen finns då simulering av dessa gör projektet för komplext i dagsläget och är en möjlig vidareutveckling om testsystemet blir användbart. I projektet kommer ett grafiskt gränssnitt inte prioriteras.

1.3 Mål och frågeställningar

Målet är att utveckla en simulator för värmepumpar som enkelt går att använda och vidareutveckla i EasyServs testmiljö. För att uppnå detta finns det frågor som måste besvaras. Här nedan följer de största frågorna som identifierats och ska besvaras:

- Hur ska testsystemet designas för att uppfylla kraven?
- Hur ska användargränssnittet utformas för att uppfylla kraven?

- Hur ska temperaturgivarna simuleras för att efterlikna värmepumpens temperaturgivare?
- Vilken kommunikation lämpar sig bäst i testsystemet beroende på faktorer som exempelvis vidareutveckling, prestanda och hastighet?

1.4 Krav

Projektets huvudkrav listas nedan och i Bilaga 1 i rapporten finns samtliga krav.

- Testsystemet ska utifrån historik simulera åtta temperaturgivare
- Temperaturgivarna som simuleras ska minst ha 2°C upplösning
- Testsystemet ska bestå av ett inbyggt system
- Testsystemet ska kunna kommunicera med värmepumpens styrkort
- Testsystemet ska kunna simulera följande tre driftlägen: normal sommardag, normal vinterdag och ett felaktigt driftläge.
- Testsystemets olika driftlägen skall kunna modifieras

2 Bakgrund

Kapitlet behandlar den nödvändiga teorin bakom de tekniska delarna i projektet. Här genomförs även en undersökning av relaterade arbeten.

2.1 Liknande arbete

Projektet som genomförs åt EasyServ kan relateras till projektet "Utveckling av en testmiljö för Thermias värmepumpar"[3]. Även i detta projekt ersätts värmepumpens givare med en elektronisk enhet som ska testa och verifiera värmepumpens styrsystem. Den elektroniska enheten består av bl.a. digitala potentiometrar, vilka simulerar värmepumpens NTC-motstånd. Resultatet av projektets design blev två system, ett kopplat till värmepumpens styrsystem och ett som agerar användargränssnitt. Systemen består bl.a. av mikrokontrollers som kommunicerar via Inter-Integrated Circuit(I^2C).

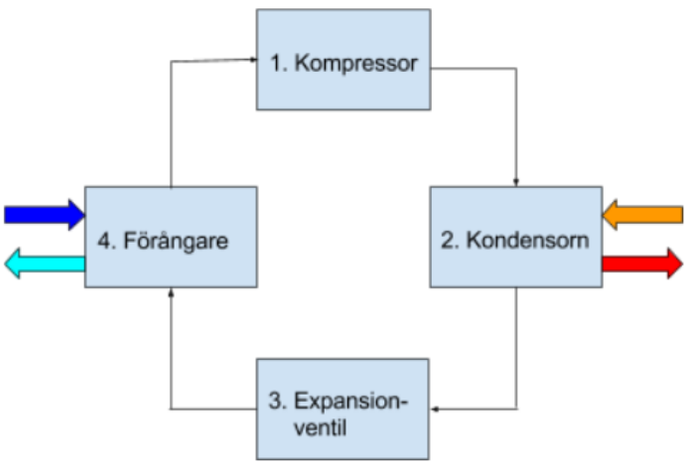
I det relaterade projektet finns saker som uppmuntrar till idéer för projektet som genomförs åt EasyServ. Det är t.ex. hur simuleringen och kommunikationen för värmepumpens temperaturgivare kan genomföras. I detta projekt används ex. Serial Peripheral Interface(SPI) för kommunikation med de digitala potentiometrarna. Detta och liknande gränssnitt kommer undersökas för tillämpning i projektet.

Ett annat projekt som kan relateras till är "Research on Thermistor Simulation Based on Digital Potentiometer for Microsatellites"[4]. I detta projekt testas ett delsystem för termiska satelliter genom att simulera temperaturvärden m.h.a digitala potentiometrar. Ett krav som finns för projektet är att det simulerade värdet högst får avvika 1.23°C från den önskade temperaturen. Detta krav uppfylls och kan relateras till krav 9(se Bilaga 1). Ett annat krav var att temperaturer inom ett visst temperaturområde skulle vara möjliga att simulera och även detta kan relateras till krav 8(se Bilaga 1).

2.2 Värmepumpar

Värmepumpen[5] är en värmekälla som använder elenergi för utvinning av värmeenergi från bl.a. marken, luften eller berggrunden. Värmen som levereras är ungefär tre gånger större än den elenergi som används för drift. Den tekniska anordningen[6] för en generell värmepump syns i Figur 2 och består av följande fyra huvudkomponenter:

- **Förångare**- i denna del av processen tar det kalla köldmediet[1]upp värme från värmekällan eller köldbäraren och förångas.
- **Kompressor**- gasen från förångaren sugs in och pressas ihop i kompressorn, detta medför ökning av tryck och temperatur.
- **Kondensor**- hit kommer gasen från kompressorn och överför värme till värmesystemet via en värmebärare. Den heta gasen kyls ned efterhand och kondenserar till vätska för att sedan flyta vidare till förångaren på nytt.
- **Expansionsventil**- reglerar köldmediets flöde från kondensorn till förångaren och ser till att tryckskillnaden mellan den varma och kalla sidan upprätthålls.



Figur 2: Värmepumpens arbetsprocess med de fyra huvudkomponenterna.

[1]Köldmediumet är vätskan som transporterar runt energi till de olika huvudkomponenterna[6].

Det finns olika typer av värmepumpar[7] på marknaden idag och exempel på dessa är mark-, sjö- och bergvärmepumpar. Dessa går under kategorin slutna vätskesystem och såldes mest år 2013. Under samma år låg frånluftsvärmepumpar och luft/vatten-värmepumpar på andra respektive tredje plats. Största skillnaden mellan värmepumparna, som också kännetecknas av namnen, ligger i vart värmekällan finns.

För att värmepumpens mekanism ska fungera på ett tillfredsställande sätt används olika givare som mäter parametrar som exempelvis tryck och temperatur. Några av de mest väsentliga temperaturerna[8] som mäts är värmesystemets returvatten, inomhus-, utomhus- och varmvattentemperaturen. Syftet med mätningarna är att värmepumpens styrenhet ska trigga igång eller stänga av olika komponenter i värmepumpen.

Det som orsakar att olika komponenter triggas igång eller stängs av i en värmepump är bl.a. värmekurvan[9]. Värmekurvan bygger på att det ska vara en jämn inomhustemperatur. Dess funktion är att när utomhustemperaturen ökar/sjunker så agerar framledningstemperaturen tvärtom. Hur mycket framledningstemperaturen ska öka/sjunka beror på kurvans lutning som i sin tur beror på faktorer som bl.a. radiatorer, golvvärme och isolering i huset.

2.3 Simulator

Simulering[10] handlar om eftersträvan att så bra som möjligt imitera ett verkligt system eller en process över tid. Det finns en mängd olika användningsområden för simulatorn, bl.a. används det i utvecklingen av ett systems design men också för att beskriva och analysera ett systems uppföranden. Ett exempel på uppförande som kan simuleras är utsättande för kritiska gränser där systemet i verkliga fall hade kunnat brista. En simuleringsmodell[10] ökar kunskapen om ett system och är väldigt värdefull när potentiella förändringar ska undersökas. Simulering är även ett bra sätt att hålla ner kostnaden och tiden vid produktutveckling då behovet av att ta fram en serie prototyper undviks[11].

Datorsimulering[11] bygger på en modell som matematiskt beskriver en process eller system. Kvalitén av modellen är grunden för hur väl det verkliga systemet kan efterliknas. Datorsimulering används för att exempelvis träna piloter inför både väntade och oväntade händelser under flygning. Det används även för att undersöka processer som i princip är omöjliga att testa p.g.a. bland annat kostnaden eller hastigheten.

Ett annat tillvägagångssätt för undersökning av ett systems uppförande är m.h.a. beräkningsvetenskap[12]. Detta är ett vetenskapligt område som mynnar ut i tre vetenskapliga grenar där komplexa beräkningsproblem behandlas. Grenen “Computer and information science” behandlar bl.a. biomedicinska-och energi-problem.

“Algorithms and modeling and simulation software” använder programvara för att lösa bl.a. komplexa vetenskapliga och humaniora problem. Fysiologiska och sociologiska problem är några av de områden som “Computing infrastructure” behandlar.

Några andra tekniker som används för att simulera ett system är HIL[13] och software-in-the-loop-simulation(SIL)[14].

2.3.1 Hardware-in-the-loop

Under detta avsnitt beskrivs grundidén bakom HIL och dess arkitektur utifrån tidskriften [13]. HIL är en teknik som faller under metoden realtidssimulering p.g.a. användandet av verkliga komponenter tillsammans med realtidssimulerade komponenter i samma system. Realtidssimulering innebär att de simulerade komponenternas in- och utsignaler klockas i samma takt som de faktiska komponenterna. Vid användning av HIL förblir styrenheten ofta densamma medans den kontrollerade processen(ställdon, sensorer och fysiska processer) kan utgöra olika kombinationer av simulerade och verkliga komponenter. Den vanligaste kombinationen är att behålla styrenheten och ställdon ihop med simulerade sensorer och fysiska processer.

I en artikel[15] från tidsskriften “Embedded Systems Design” beskrivs det att användning av HIL är lämpligt ur olika aspekter, bl.a. är det kostnadseffektivt då dyra och misslyckade systemtest undviks. Det är även effektivt ur den tekniska aspekten då antalet drifttest minskas. Ett exempel på detta är vid testning av bilens låsningsfria bromssystem som testas på olika väglag. En annan fördel med korrekt utformad HIL är att utvecklingsprocessen av produkter påskyndas. Detta medför även att systemtest blir mer noggranna till en lägre kostnad i jämförelse med användning av traditionella testmetoder.

Ett exempel på tillämpning av HIL i industrin beskrivs i artikeln[16] som undersöker användningen av HIL för testning av bilens elektroniska styrsystem. Det som styrsystemet ska styra är bilrutan, låset och backspegeln, vilka också är de olika delsystemen som projektet delades upp i. Denna uppdelning görs för att varje del ska kunna testas var för sig och på så vis spara både tid och pengar i utvecklingsprocessen. Slutligen integreras alla delsystem ihop till ett system. Utvärderingen av resultatet var bl.a. att denna simuleringsteknik har stor potential att snabbare få ut produkten på marknaden och även minska kostnaden för produktutvecklingen.

2.3.2 Software-in-the-loop

SIL[14] är en simuleringsmetod som inte använder något elektrisk gränssnitt. Det som används istället är operativsystemets mjukvarugränssnitt som möjliggör direkt kommunikation med simuleringen. Detta medför stor flexibilitet till att utföra tester

då ingen hårdvara behövs och därför kan SIL användas tidigt i mjukvaruutvecklingen.

I denna rapport, som ligger till grund för testsystemet som utvecklas, finns kravet att testsystemet ska bestå av ett inbyggt system. Med avseende på detta genomförs ingen djupgående fördjupning av SIL eftersom det slutliga testsystemet inte kommer använda den här simuleringsmetoden. Metoden nämndes utav anledningen att det är en vanlig teknik vid simulering.

2.4 Inbyggda system

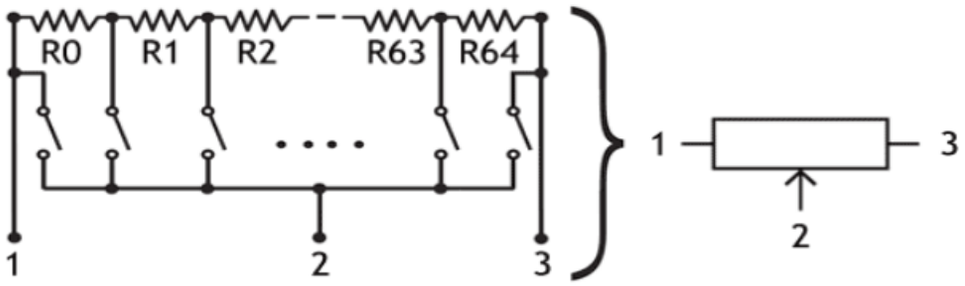
Ett inbyggt system[17] är ett datorbaserat system vars komplexitet, som sitter i en mikrokontroller eller mikroprocessor, ofta inte syns för användaren. Skillnaden mellan inbyggda system och en vanlig hemdator är bl.a. att ett inbyggt system endast kan utföra en eller flera förutbestämda uppgifter. Några exempel på tillämpning av inbyggda system är ABS-systemet i en bil, mobiltelefonen och mikrovågsugnen. Inbyggda system finns även i hårdvaruplattformarna Arduino[18] och Raspberry Pi[19], där den sistnämnda även kan användas som en vanlig dator när ett operativsystem installeras och körs.

Arduino är en mikrokontroller med öppen hårdvara som är konstruerad med en lättanvänd hård- och mjukvara. Detta har följt med sedan utvecklingen av den första Arduinon som var avsedd för studenter utan bakgrund från elektronik och programmering. En typisk Arduino plattform är Arduino UNO[20] som baseras på mikrokontrollern ATmega 328P. Den består bl.a. av 14 digitala in- och utgångar, sex analoga ingångar och har en klockhastighet på 16MHz. Plattformen stödjer bl.a. kommunikationsgränssnitten SPI och I^2C. Arduinon har en stor uppsamling av funktioner[18], ex. kan allt från ljuständning till publicering av saker på internet utföras. Detta, ihop med dess användarvänlighet, har gjort att den över åren använts i tusentals olika projekt.

Raspberry Pi[19] är en funktionell och användarvänlig minidator som kräver ett operativsystem för att köras. Den kan utföra de flesta funktioner som en vanlig dator kan och alla versioner av Raspberry Pi använder metoden System on a Chip(SOC). Detta innebär att all nödvändig elektronik för att datorn ska fungera samlas på ett chip. Raspberry Pi består av antingen 26 eller 40 General Purpose Input/Output(GPIO) och är kompatibelt för bl.a. kommunikationsgränssnitten SPI och I^2C. Genom inkoppling av en skärm och ett tangentbord kan den bl.a. användas till elektronikprojekt, spela spel och surfa på internet.

2.5 Digitala potentiometrar

Digitala och mekaniska potentiometrar har samma funktion som ett variabelt motstånd. Skillnaden är att digitala potentiometrar styrs av digitala signaler och mekaniska potentiometrar styrs av mekanisk rörelse. I Figur 3 syns principen för en digital potentiometer. Denna bygger på användningen av en motståndsstege, vilket kan ses som en sträng av små motstånd. Vid varje steg av denna sträng finns en elektronisk brytare där endast en kan slutas åt gången. Motståndet bestäms av den slutna brytaren och värdet för varje steg beror på upplösningen för vald digital potentiometer.[21] Genom användning av I^{2}C eller SPI kan digitala potentiometrar styras, men



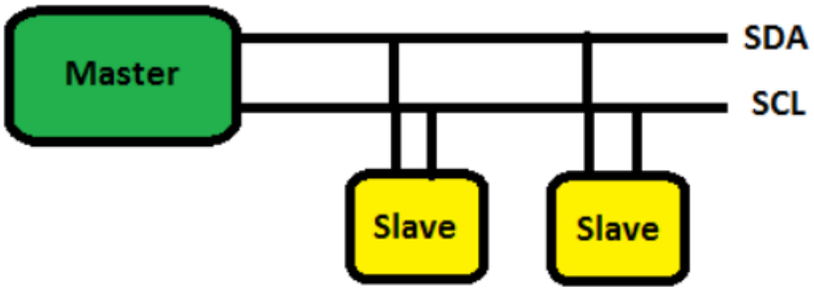
Figur 3: Principen för digitala potentiometerns motståndsstege där endast en brytare kan slutas åt gången. I figuren syns en digital potentiometer där ände-till-ände-motståndet[2] delas upp i 64 steg[21].

även enkla upp- och nedsignaler kan användas. En digital potentiometer kan ersätta en mekanisk potentiometer eller ett fast motstånd.[21] Fördelarna över mekaniska potentiometrar är bl.a. inget mekanisk slitage, inget behov av skruvmejsel samt en mindre fysisk storlek[22]. Nackdelen är att matningsspänningen, som vanligen är fem volt, kan försvåra en ersättning av mekaniska potentiometrar[21].

2.6 Kommunikationsgränssnitt

I^{2}C[23] består av två ledare, vilka benämns Serial Data(SDA) och Serial Clock(SCL). Figur 4 visar hur uppkopplingen för I^{2}C ser ut. Kommunikationen bygger på ett master/slav-protokoll, där masterenheten kommunicerar med slavenheter. Detta sker genom att varje slavenhet tilldelas en unik adress och därefter startar masterenheten kommunikationen. En fördel med I^{2}C är att endast två ledare krävs, vilket är lämpligt när många enheter är anslutna. Fördelen av två ledare kan dock vara en nackdel om endast ett fåtal enheter är anslutna, vilket ger onödigt hög komplexitet vid hantering av adressering och bekräftelser.

[2] Ände-till-ände-motståndet är det maximala motståndet för en digital potentiometer.



Figur 4: Principen för I^2C med masterenhet och slavenheter, där kommunikationen sker genom två ledare benämnda serial data(SDA) och serial clock(SCL)[23].

Gränssnittet SPI[24] används för kommunikation med en eller flera kringutrustningar över korta avstånd. SPI består av en masterenhet vilket kontrollerar slavenheterna. Huvudsignalerna som SPI använder är Master In Slave Out(MISO), Master Out Slave In(MOSI) och Serial Clock(SCLK). En valfri signal som också används ofta är Chip Select(CS).

Uppgifterna de olika signalerna har är följande[24]:

- **MISO**- Skickar data från slavenheten till masterenheten.
- **MOSI**- Skickar data från masterenheten till slavenheten.
- **SCLK**- Klocksignalen som är förknippad till dataöverföringen.
- **CS**- Indikerar, med hjälp av en spänningsnivå, om masterenheten pratar med slavenheten.

3 Metod

Projektets metod för utveckling av testsystemet är uppdelad i tre faser. Faserna genomförs i ordningen förstudie, utvecklingsfas och utvärderingsfas.

I förstudien undersöks förutsättningarna för projektet och dess omfattning klarläggs. Här görs en projektplan och kravspecifikation, där utifrån den sistnämnda även en testspecifikation skapas. Under utvecklingsfasen köps alla nödvändiga komponenter in och konstruktionen för testsystemet påbörjas. Projektet håller ingen bestämd budget då EasyServ finansierar det som behövs, dock är en låg totalkostnad önskvärd. I denna fasen genomförs även tester för säkerställning av att testsystemets delar följer kraven som finns. Under sista fasen sammanställs alla dokument och en granskning, av att testerna i testspecifikationen är uppfyllda, genomförs. Detta ger en översikt på hur väl testsystemet fungerar, vilket underlättar för en utvärdering av projektet. I slutskedet ska testsystemet och all dokumentation överlämnas till EasyServ.

3.1 Utvecklingsfas

Testsystemets design består av en HIL-uppsättning. I denna ingår en hårdvaruplattform, ett I/O-kort och ett värmepump-styrkort. Anledningen till att hårdvara ingår i simuleringen grundas på krav 2(se Bilaga 1). Kravet ställdes från EasyServ för att testsystemet ska efterlikna värmepumpens verkliga system ännu mer än med endast mjukvaru-simulering.

Eftersom att både hård- och mjukvara är involverade i ett HIL-system delas projektet upp i två delmoment, dessa är Elektronikkonstruktion och Programmering. Det förstnämnda berör design och konstruktion av testsystemet. Detta innefattar val av hårdvaruplattform och övrig hårdvara som är nödvändig för konstruktionen. Dessa val ligger till grund för hur bra simuleringen av värmepumpens temperaturgivare kan bli. Utöver dessa givare skall även möjligheten för att inhämta två digitala signaler(från värmepumpens styrkort till I/O-kortet) undersökas. Delmomentet programmering är "hjärnan" i projektet där tillvägagångssättet för simuleringen bestäms. I detta delmoment genomförs även programmeringen av de olika beteendena för en värmepump. Dessa är "normal sommardag", "normal vinterdag" och "felaktig drift". Under båda delmomenten utförs tester under utvecklingens gång. Dessa utförs för säkerställning av att både hård- och mjukvara ska kunna uppfylla kraven som finns.

3.2 Elektronikkonstruktion

I konstruktionen av testsystemet genomförs en undersökning av vilka komponenter som ska väljas(se avsnitt 3.2.1). Komponenterna som krävs för att projektets testsystem ska vara funktionellt är: hårdvaruplattform, digitala potentiometrar och

övrig elektronik. Digitala potentiometrar har valts som strategi för simulering av värmepumpens temperaturgivare. Givarna är NTC-motstånd, vilket innebär att dess motstånd förändras beroende på temperatur. Av denna anledning är digitala potentiometrar lämpliga att använda, då det digitalt går att ställa in olika motståndsvärden. En annan uppgift under delmomentet är att försöka inhämta två digitala signaler till I/O-kortet från värmepumpens styrkort. Dessa signaler är kompressor-signalen och växelventilen. Kompressor-signalen ger feedback när kompressorn satts igång och växelventilen ger feedback när parametern "Varmvatten" har sjunkit för lågt. När elektroniken är vald påbörjas designen och konstruktionen av testsystemet.

Val av komponenter och kommunikationsgränssnitt

Valet av hårdvaruplattform grundade sig främst på anledningarna att det skulle vara lätt att komma igång, billigt, stödja SPI och I^2C. Ett annat kriterium var att det skulle vara en hårdvaruplattform som EasyServ är bekanta med, för flexibiliteten att vidareutveckla testsystemet. Det konstaterades, utifrån anledningarna, tidigt att en lämplig hårdvaruplattform för testsystemet fanns i Arduino UNO eller Raspberry Pi 1 Model B. Utifrån jämförandet sattes Tabell 1 upp.

Tabell 1: Jämför Raspberry Pi 1 Model B mot Arduino UNO.

Attribut	**Raspberry Pi 1 Model B**	**Arduino UNO**
Pris	345 kr	215 kr
Stödjer SPI och I^2C	Ja	Ja
Användarvänlig	Ja	Ja
Antal in- och utgångar	26 st	14 st

Trots att de båda systemen är förhållandevis lika och att Arduino UNO är lite billigare föll valet på Raspberry Pi. Anledningen till valet är att testsystemet får fler in- och utgångar, vilket underlättar för en vidareutveckling samt för att fler funktioner blir tillgängliga med minidatorn Raspberry. En annan anledning som har vägt tungt i valet är att EasyServ känner sig mer bekanta med Raspberry Pi.

Valet av digital potentiometer grundas på bl.a. krav 9(se Bilaga 1), som finns från EasyServ, där en upplösning på minst 2°C ska uppnås. För att uppfylla detta krävs en mätosäkerhet som inte överstiger ± 3%. Anledningen till kravet är att simuleringen ska vara tillförlitlig och efterlikna en värmepump så mycket som möjligt. Detta kräver en hög upplösning och en låg mätosäkerhet som håller nere medelkvadratfelet[3] samt det maximala felet för mätpunkterna. Motståndet ska även ha en räckvidd som motsvarar temperaturer mellan -5 till 90°C för ett 4.7 kOhms NTC-motstånd(se Bilaga 6). Egenskapen sökes då temperaturerna normalt ligger inom detta intervall

[3]Ett mått på hur stort det totala felet är i genomsnitt.

i värmepumpssystemet. Temperaturintervallet är dock lite begränsat för att kunna bibehålla upplösningen. Denna upplösning beräknas genom en division mellan ände-till-ände-motståndet och antalet steg i potentiometern. Exempelvis har en digital potentiometer med samma antal steg, fast med högre ände-till-ände motstånd, en sämre upplösning än en med lägre ände-till-ände motstånd. En annan egenskap som söks är att SPI eller I^2C ska stödjas eftersom att projektgruppen vill bredda kunskaperna inom detta.

Övriga val av omkringliggande komponenter görs för att funktionen för systemet ska fungera på ett tillfredsställande sätt.

Tabell 2: Jämför I^2C mot SPI.

I^2C	**SPI**
Skickar bekräftelse vid varje överföring	Skickar ingen bekräftelse
Maximal hastighet 3.4 Mbps	Maximal hastighet 10 Mbps
Kräver endast 2 ledare	Ledare som krävs är 3 + antalet slavenheter
-	SPI är mer störningskänslig än I^2C

Valet av seriellt kommunikationsgränssnitt[25] gjordes utifrån attributen tillförlitlighet, hastighet, störningskänslighet samt fysisk komplexitet av fler inkopplade slavenheter. I Tabell 2 syns jämförelsen mellan SPI och I^2C utifrån ovannämnda attribut. Anledningen till dessa önskade attribut är för att testsystemet bl.a. ska vara tillförlitligt, vilket kräver en kommunikation som inte är störningskänslig. Det är även viktigt för en hög tillförlitlighet att kommunikationsgränssnittet skickar bekräftelser. Ett annat önskemål är att ha en låg fysisk komplexitet, vilket gör att fler slavenheter enkelt kan kopplas in. Detta förenklar en eventuell vidareutveckling av testsystemet. SPI har en fördel i överföringshastighet eftersom att det bl.a. inte skickas bekräftelser. Denna egenskap prioriterades inte p.g.a att datan i EasyServs diagnosverktyg loggas under långsammare tidsintervall än SPI och I^2C:s överföringshastighet. Värden loggas exempelvis var femte minut när kompressor är igång och var 15:e minut när den är av. Fördelarna med I^2C är att det är mindre störningskänsligt än SPI och kräver endast två ledningar oavsett antal slavenheter. Dessa fördelar, tillsammans med tillförlitligheten i skickandet av bekräftelser, gjorde att valet föll på I^2C.

Konstruktion och design

I projektet kommer två mönsterkort designas, ett för testning av den digitala potentiometern och ett som ska agera I/O-kort i testsystemet. Syftet med testkortet är att få kunskap om hur den digitala potentiometern fungerar. I/O-kortets design ska bestå av åtta uppkopplade digitala potentiometrar med tillhörande komponenter samt stiftlist för enkel in- och urkoppling. Detta kort kommer vara ett tvålagers

mönsterkort med jordplan på båda sidorna för att undvika störning av signalerna. Det slutgiltiga I/O-kortet kommer beställas från företaget 3PCB. Processen för framtagning av mönsterkort inleds med att designa ett kretsschema som visar hur komponenterna kopplas ihop. Detta görs i programmet OrCAD Capture[26]. När kretsschemat är klart kommer layouten för mönsterkortet designas i programmet OrCAD PCB Designer[26]. Innan layouten skapas måste alla komponenter ha footprints. De som finns i OrCAD:s bibliotek hämtas för användning och resterande skapas i programmet PCB Library EXPERT[27]. Vid beställning av mönsterkortet skickas layoutfilerna till företaget. Tillverkningen, av mönsterkortet i högskolan, sker genom processen förberedelse av mönsterkort, etsning, borrning och lödning.

3.3 Programmering

I detta delsystem väljs ett programmeringsspråk som är kompatibelt med Raspberry Pi. I denna plattform ska mjukvaran, som behandlar datahistoriken från värmepumpens temperaturgivare, utvecklas. Mjukvaran ska främst programmeras till att simulera följande tre driftlägen för värmepumpen: normal sommardag, normal vinterdag och ett felaktigt driftläge. Driftlägena väljs ut genom en enklare analys av datahistoriken som erhålls från EasyServ. Analysen utförs genom granskning av månaderna juli och januari för att finna en normal dag med stabila temperaturer i respektive månad. Datan för det felaktiga driftläget erhålls från företaget, där ett felaktigt beteende kommer väljas ut. Tillvägagångssättet för utveckling av mjukvaran kommer utformas efter ett flödesschema som görs i början av utvecklingsfasen. Flödesschemat kommer beskriva processen från simuleringens början till dess slut. I mjukvaran kommer bl.a. en algoritm skapas för omvandling av temperaturer till dess respektive motståndsvärde. Under detta delsystem kommer även ett användargränssnitt väljas ut och utformas.

Val av användargränssnitt

Användargränssnittet kommer bestå av en konfigureringsfil som användaren väljer för att simulera ett visst läge i testsystemet. Filerna, som det ska vara möjligt att välja mellan, placeras i ett bibliotek på Raspberry Pi. Programmet ska, från detta bibliotek, läsa in den valda konfigureringsfilen och genomföra simuleringen. Valet av att använda konfigureringsfiler som användargränssnitt gjordes eftersom att det uppfyller krav 5(se Bilaga 1), då modifiering av driftlägena kommer vara möjlig. Andra gränssnitt hade också kunnat uppfylla kravet men hade krävt mer tid att utveckla. Utveckling av ett sådant gränssnitt kommer ske om tid finns över, när den övriga delen av testsystemet är färdigt.

Val av programmeringsspråk

I valet av ett programmeringsspråk fanns bl.a. alternativen: Python, C/C++ och Java. EasyServ önskar ett program i Python, då deras övriga mjukvara är skriven i detta språk. Valet föll därför på Python som kommer underlätta framtida modifieringar av mjukvaran, vilket också finns som krav 5(se Bilaga 1).

3.4 Tester

Tester kommer utföras under utvecklingsfasen, för säkerställning av att båda delsystemen uppfyller kraven som finns. Genomförandet av dessa tester förklaras utförligt i testspecifikationen(se Bilaga 2). Fokuset för testerna kommer ligga på: prestandan för de digitala potentiometrarna, integrering av delsystemen och kommunikationen mellan I/O-kortet och värmepumpens styrkort. I slutet av utvecklingsfasen ska ett acceptanstest genomföras, för verifiering av att resultatet uppnår de krav som ställts. Testet kommer utföras hos EasyServ, där I/O-kortet kopplas till Raspberry Pi och värmepump-styrkortet. En konfigureringsfil, som skapats från datahistorik, ska sedan läsas in och köras i simuleringsprogrammet. Under testets gång kommer testsystemet vara uppkopplat till EasyServs gateway, där värden loggas och plottas till en graf.

3.5 Utvärdering

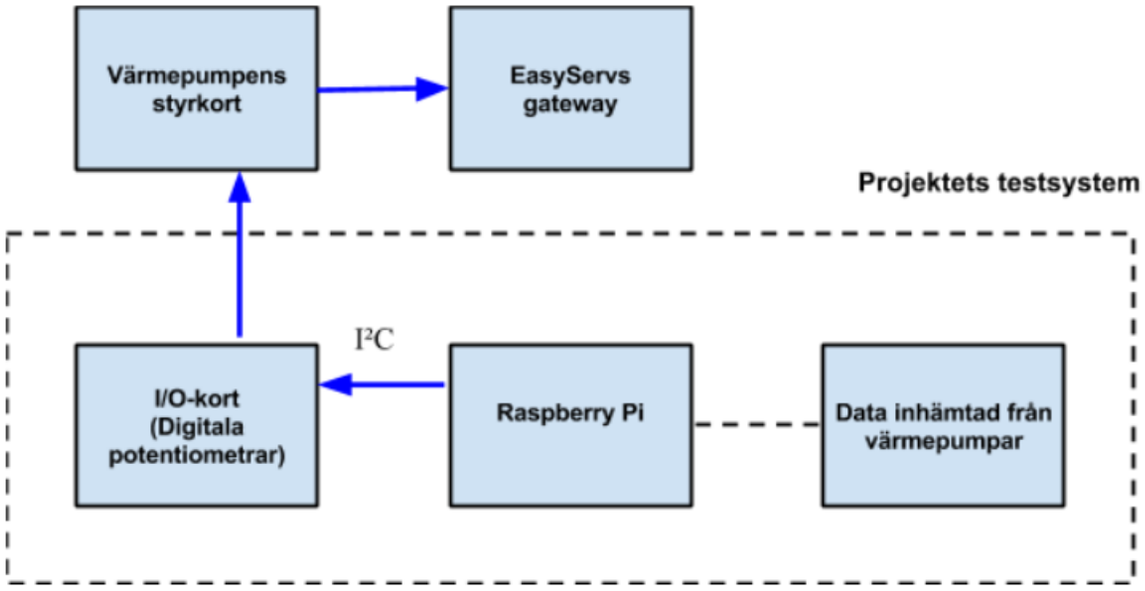
Under utvärderingsfasen kommer testsystemet utvärderas genom en granskning av testerna som utförts i testspecifikationen(se Bilaga 2). Denna granskning går ut på att bl.a. jämföra hur bra kvalité simuleringen håller relativt de verkliga temperaturgivarna. Kommunikationen mellan styrkortet och testsystemet kommer utvärderas efter noggrannheten av signalerna som mottages på styrkortet. Efter att denna fas har genomförts ska testsystemet överlämnas till EasyServ.

4 Resulat

I detta kapitel redovisas alla uppnådda resultat i projektet.

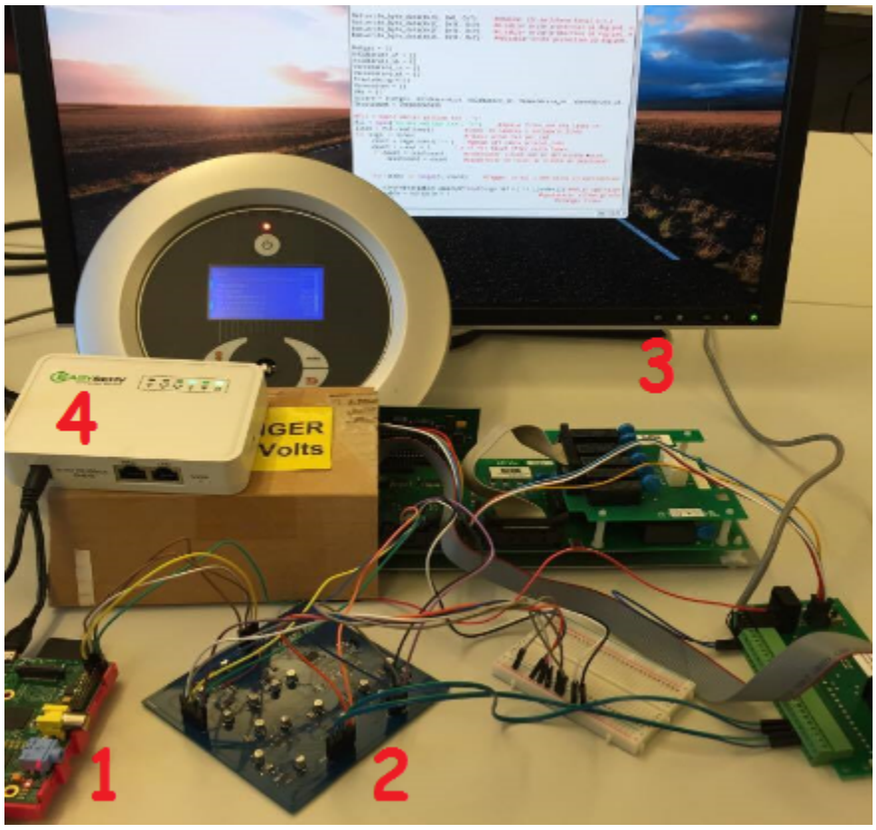
4.1 Testsystemets design

Under detta avsnitt beskrivs resultatet av testsystemets design. Den slutliga designen



Figur 5: Översikt av testsystemets slutgiltiga design

för testsystemet blev enligt Figur 5. I detta ingick utveckling av delarna som finns inom det streckade området. Resultatet blev ett system där en Raspberry Pi läser in en konfigureringsfil och simulerar värmepumpens temperaturgivare. I/O-kortet tar emot det simulerade värdet från Raspberry Pi:n via I^2C och skickar vidare detta till värmepumpens styrkort. Detta beskrivs mer utförligt under avsnitt 4.2. Figur 6 visar hur det uppkopplade testsystemet ser ut i labbmiljön.



Figur 6: Översikt av testsystemets labbuppställning. 1. Raspberry Pi, 2. I/O-kort, 3. Värmepumpens styrkort, 4. EasyServs Gateway. Testsystemet kompletteras med en kopplingsplatta för att sammankopplas till styrkortets CANbuss-system. En monitor, som tillhör styrkortet, visar de simulerade temperaturerna. En extern datorskärm och ett USB-kopplat tangentbord(syns ej i figuren) används för att kunna köra programmet i Raspberry Pi:n.

Valet av digital potentiometer

Den digitala potentiometern som har valts är en AD5272(se Bilaga 5) från Analog Devices. Andra alternativ som har undersökts har haft brister i antingen antalet steg eller ände-till-ände-motståndet. Den bästa kombinationen av dessa egenskaper hittades i AD5272. Komponenten har ett ände-till-ände motstånd på 20 kOhm och en upplösning på 1024 steg(10 bitar). Detta möjliggör en ändring på 19,53 Ohm/steg. Mätosäkerheten är $\pm$ 1% vilket motsvarar ett maximalt fel på 200 Ohm. Komponenten är även kompatibel med I^2C och kan tilldelas tre unika I^2C adresser.

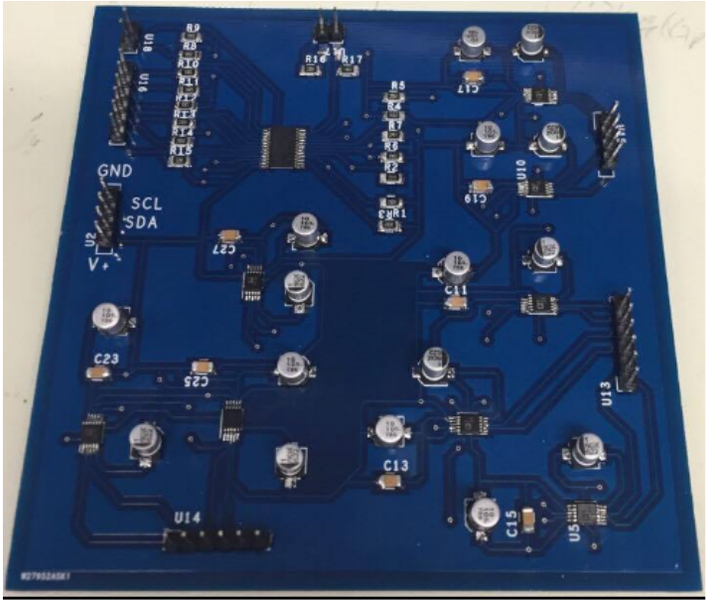
Då den digitala potentiometern endast har tre unika I^2C-adresser, kompletterades testsystemet med I^2Cbuss-switchen PCA9548A(se Bilaga 5). Denna möjliggör användning av åtta kanaler vilket undviker kollision av likadana adresser.

4.2 Implementation av testsystemet

I detta avsnittet redovisas resultat från projektets testsystem.

Projektets testsystem

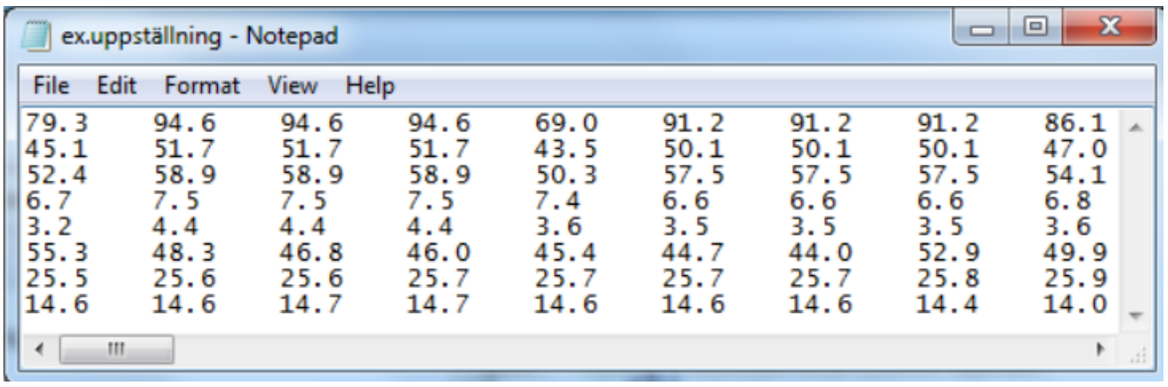
Figur 7 visar I/O-kortet som har skapats för interagering mellan värmepumpens styrkort och Raspberry Pi. Kortet består av två lager och har ett jordplan som täcker den större delen av båda sidorna. Alla komponenter finns på kortets övre lager. Dessa är: åtta digitala potentiometrar, avkopplingskondensatorer, I^2Cbuss-switch, pull-up-motstånd och stiftlister. Avkopplingskondensatorer finns kopplade nära respektive digital potentiometer för att undvika störningar. Tre av I^2Cbuss-switchens kanaler används för de åtta digitala potentiometrar. Uppställningen för dessa är: tre styck i kanal ett, tre i kanal två och två i kanal tre. De resterande kanalerna är lediga för inkoppling av externa I^2C-kompatibla enheter. Pull-up-motstånd finns kopplade till varje I^2C-kanal. In- och utgångarna från kortet är kopplade till stiftlister som enkelt går att koppla in sig på kanterna av kortet.



Figur 7: Ovansidan av I/O-kortet som konstruerats för testsystemet.

Utifrån *Data inhämtad från värmepumpar*(Figur 5) har två txt-filer skapats. Filerna består av två dagars utvalda temperaturer, för givarna i Bilaga 7. Den ena symboliserar en vinterdags temperaturer och den andra en sommardags. Vinterdagens txt-fil innehåller mer varierande värden än sommardagens eftersom kompressorn går igång mer frekvent under denna årstid. Filen består av åtta raders temperaturvärden där varje rad tillhör en särskild givare. Uppställningen av givarna är enligt

de åtta första raderna i Bilaga 7, där även givarnas placering ges och dess innebörd förklaras. Temperaturerna måste vara åtskilda med en tab för att algoritmen skall kunna läsa in alla värden från raden. I Figur 8 visas ett utdrag ur en txt-fil som visar på hur uppställningen av temperaturerna ska se ut för att läsas in korrekt av algoritmen. Under avsnitt 3.2.1 nämndes det att EasyServs diagnosverktyg loggar värden under två tidsintervall. I ett tidsintervall loggas värden var femte minut, detta endast när kompressorn är igång. De berörda givarna för detta är: Hetgas, Köldbärare ut, Köldbärare in, Värmebärare ut, Värmebärare in. I det andra tidsintervallet loggas värden kontinuerligt var 15:e minut, detta berör givarna: Ute, Framledning och Varmvatten. Detta innebär att givarna i det ena tidsintervallet loggas mer/mindre frekvent beroende på hur ofta kompressorn är igång. Detta gör att de båda tidsintervallens loggningstider måste synkroniseras tidsmässigt. Synkroniseringen görs för att algoritmen är uppbyggd på att värdena som läses in från txt-filen, ligger tidsmässigt i fas. Denna synkronisering görs genom att fördröja värdena för tidsintervallet som loggas var femte minut.

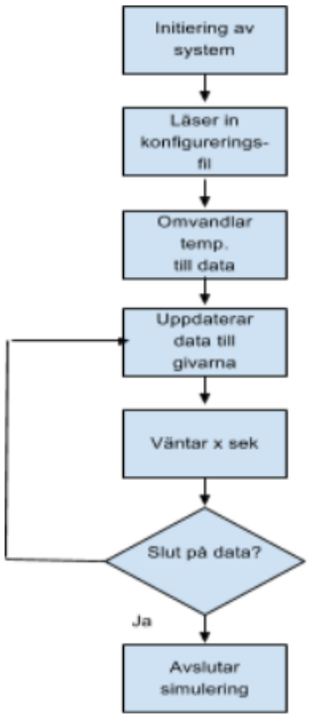


Figur 8: Här syns ett utdrag ur en txt-fil som visar på hur uppställningen ska se ut för att läsas in korrekt.

I *Raspberry PI*(Figur 5) läses en utvald txt-fil in i programmet av en algoritm. Algoritmen läser in txt-filen rad för rad och utdelar varje rad till en specifik vektor, exempelvis kommer den översta raden(Hetgas) i txt-filen tilldelas en vektor i programmet. Värdena, i den skapade vektorn, omvandlas till data som skickas till en specifik digital potentiometer. Denna process beskrivs mer utförligt längre ner i detta avsnitt.

Flödesschema

Flödesschemat som har skapats syns i Figur 9 och beskriver hur simuleringen genomförs. Under varje etapp tillkommer programmering.



Figur 9: Flödesschemat för simuleringsprogrammet.

Följande underrubriker, i fetstil, förklarar de olika etapperna(Figur 9) i ordningen uppifrån och ned.

Initiering av system

Här sker initiering av variabler och de digitala potentiometrarnas skriv-skydd inaktiveras. Vektorer för de olika givarna skapas.

Läser in konfigureringsfil

Driftläget som väljs för att simuleras, läses in av programmet. De driftlägen som txt-filer tagits fram för är en vinterdag(Winter edition.txt) och en sommardag(Summer edition.txt). Det finns ingen rimlig begränsning på hur många txt-filer som kan skapas eller hur lång tid som går att simulera i projektets fall(1,5GB minne)

Omvandlar temp. till data

I programmet skapades en algoritm för att omvandla temperatur till den digitala potentiometerns motsvarande datavärde. I denna algoritm sker först en omvandling av temperaturen till motsvarande motstånd och sedan till det motsvarande datavärdet. I omvandlingen uppstår fel, då ekvation (1) inte motsvarar NTC-motståndets graf helt fullkomligt. Ett fel uppstår även när omvandlingen från motstånd till data görs, då den digitala potentiometern har en mätosäkerhet samt en begränsad stegupplösning. Simuleringen bygger på temperaturgivare av varianten NTC-motstånd som har motståndet 4,7 kOhm vid 25°C. En generell ekvation för ett NTC-motstånd ges av formeln[28]:

$$R = R_{25^\circ\mathrm{C}} \cdot e^{\beta \cdot ((1/T_\mathrm{k}) - (1/298.15)))} \qquad \textbf{(1)}$$

där R är motståndet för en specifik temperatur, $R_{25^\circ C}$ är temperaturgivarens motstånd vid 25°C, β är temperaturgivarens betavärde(3832 Kelvin i projektets temperaturgivare), $\mathrm{T_k}$ är den specifika temperatur(Kelvin) som motståndet beräknas fram för. Ekvationen(1) beräknar ett motstånd som representerar en specifik temperatur för NTC-motståndet(se Bilaga 6).

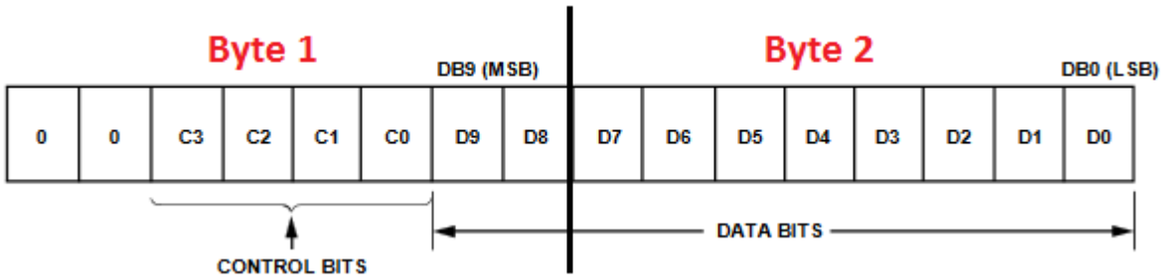
Motståndet som erhålls från från ekvation(1) behöver omvandlas till motsvarande datavärde(0-1023). Detta görs i följande ekvation:

$$D = \frac{1024}{R_\mathrm{p}} \cdot R \qquad \textbf{(2)}$$

där D är värdet på datan, $\mathrm{R_p}$ är den digitala potentiometerns ände-till-ände motstånd, R är specifikt motstånd som beräknades fram i ekvationen(1).

Uppdaterar data till givarna

Skiftregistret för AD5272(Figur 10) består av två bytes som skickas en byte i taget. I dessa bytes finns tio databitar(D0-D9), fyra kommandobitar(C0-C3) och två oanvända bitar. Databitarna sätter motståndsvärdet och är uppdelade i två bytes, av denna anledning har en algoritm skapats för att sätta rätt bitar(Figur 11). C0-C3 bestämmer vilken funktion som ska utföras, exempelvis aktiveras skriv-funktionen genom att sätta C0 hög.



Figur 10: AD5272's skiftregister.

Funktionen "bus.write_byte_data" har tre inparametrar som i Figur 11 namnges "address", "cmd" och "data". De två sistnämnda motsvarar respektive Byte 1 och Byte 2 i Figur 10. I "address" anges I^2C-adressen för den digitala potentiometern. Datavärdet från (2) är ett tal mellan 0-1023 som, vid större värde än 255, medför ett bit-skiftande i Byte 1. Algoritmen i Figur 11 visar hur bit-skiftandet i denna byte sker, där C0 alltid sätts hög för att aktivera skriv-funktionen. Algoritmen som har tagits fram bygger på information från databladet(se Bilaga 5).

```
cmd = 4

if data > 255:
    if data > 511:
        cmd = 6
        data =  data - 512
        if data > 255:
            cmd = 7
            data = data - 256
    else:
        cmd = 5
        data = data - 256

bus.write_byte_data(address, cmd, data)
```

Figur 11: Algoritmen beskriver uppdelningen av data innan den skickas till digital potentiometer.

4.3 Tester

I detta avsnitt beskrivs resultaten av de olika testerna som gjorts. För utförlig beskrivning av testernas genomförande, se testspecifikation(se Bilaga 2).

4.3.1 Test av potentiometer

Resultatet av mönsterkortet som tillverkades på högskolan blev bra då det enkelt gick att koppla det till en testplatta. Ett funktionstest gjordes som gick ut på att ändra motståndsvärdet för den digitala potentiometern. För kontroll av att motståndet ändras, användes en digital multimeter. Testet finns som test 6 i testspecifikationen(se Bilaga 2). Testet blev avklarat då den digitala potentiometern fungerade, detta dukade för att vissa av kraven skulle bli uppfyllda.

Testet gick bra då den digitala potentiometern fungerade enligt vad projektgruppen trodde. Detta uppfyller krav 5(se Bilaga 1)

4.3.2 Test av I/O-kort

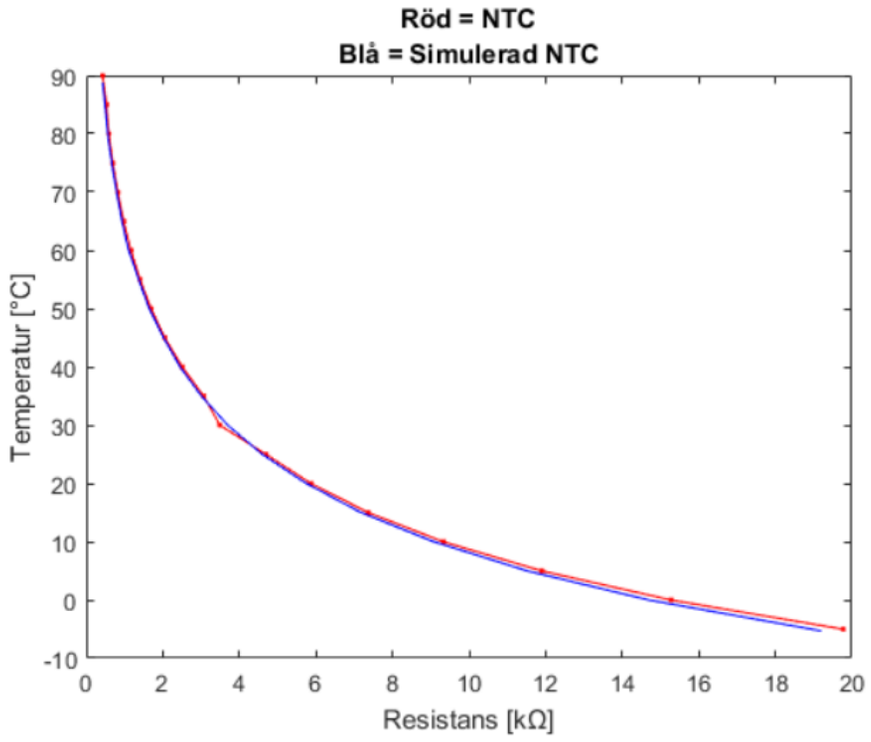
Testsystemets funktionalitet testades genom att först detektera I^2Cbuss-switchen och därefter alla digitala potentiometrar. Det sista testet genomfördes för kontroll av att I/O-kortet klarar ändra samtliga digitala potentiometrars motståndsvärden. Testet finns som test 6 i testspecifikationen(se Bilaga 2).

Funktionaliteten godkänns då de olika testerna har klarats av. Detta medför att krav 3, 7 och 11 i kravspecifikationen(se Bilaga 1) är uppfyllda.

4.3.3 Test av simuleringens noggrannhet

Noggrannheten av simuleringen testades genom att skicka in temperaturer mellan -5°C till 100°C med fem graders mellanrum. Detta kontrollerades visuellt(på styrkortets skärm) och genom mätning av motstånd på de digitala potentiometrarna.

Ett stationärt fel på -0,5°C upptäcktes mellan temperaturområdet 45°C till -5°C. Detta åtgärdades genom att subtrahera -0,5°C från algoritmen där temperaturen omvandlas. Detta görs endast inom temperaturområdet där felet upptäcktes. Detta ledde till att resultatet blev bättre, då medelkvadratfelet blev 0,24°C jämfört med föregående test som låg på 0,37°C. Det maximala felet på mätpunkterna blev 1.3°C och låg på 85°C. I Figur 12 har en graf tagits fram från testet på den optimerade algoritmen. Denna graf visar på att noggrannheten av simuleringen är god och att krav 6, 8 och 9 uppfylls(Se Bilaga 1).

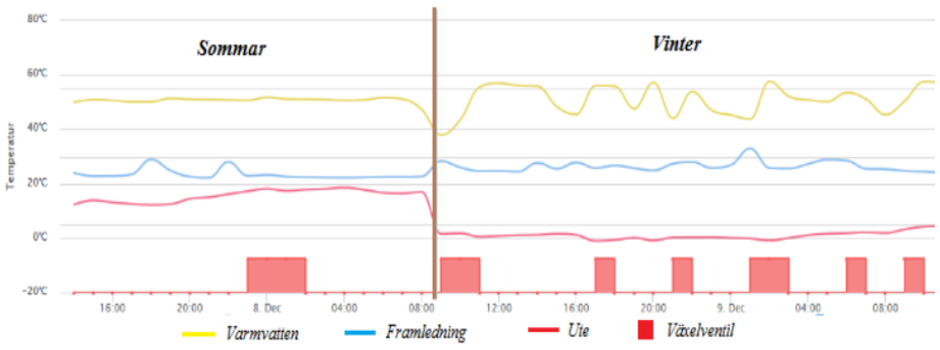


Figur 12: Grafen jämför det riktiga NTC-motståndet(röd) mot det simulerade NTC-motståndet(blå).

4.3.4 Acceptanstest

Acceptanstestet gick ut på att kunna simulera en hel dags temperaturer med ett tidsintervall på 15 minuter. Två txt-filer kördes i simuleringsprogrammet: “Summer edition.txt” och “Winter edition.txt”. Den förstnämnda filen representerar en normal sommardags temperaturer och den andra, en normal vinterdags. Detta kunde, med hjälp av EasyServs diagnosverktyg, följas via en graf på deras hemsida.

Resultatet av acceptanstestet blev enligt Figur 13, där man ser ett utdrag av en sommardags och en vinterdags simulerade temperaturer. Endast tre temperaturer finns plottade, då kompressorsignalen inte kunde triggas igång. De stora skillnaderna, som även syns i grafen, är bl.a. att framledningen har en högre temperatur på vintern. Detta för att kunna ha en stabil inomhustemperatur på vintern, som på sommaren. Detta visar på att en simulering av temperaturgivarna kan genomföras med testsystemet och att krav 1(se Bilaga 1) därmed blir uppfyllt.



Figur 13: Grafen visar ett urklipp av simuleringen av ett antal parametrar, där vänster om det bruna strecket representerar sommardagen och höger om strecket representerar vinterdagen.

4.4 Utvärdering av testsystem

Testsystemet kan, enligt kraven som satts upp i början av projektet, anses vara godkänt. Detta då de tester som klarats av var nog för att visa på funktionen av testsystemet. Testet som gjordes på testkortet gav projektgruppen den kunskap som behövdes för att förstå hur kommunikationen med I^2C fungerar och därmed kunna gå vidare. Utifrån detta kunde ett I/O-kort skapas till testsystemet. Detta kort kan ta emot simulerade värden från en Raspberry Pi och skicka vidare dem till ett värmepump-styrkort. Kvalitén av simuleringen är god, då medelkvadratfelet visade på 0,24°C från de verkliga givarnas temperaturer. Ett slutligt omfattande acceptanstest genomfördes där två olika driftläges-program kunde köras och visa på två olika årstiders temperaturer. Den slutliga kostnaden av projektet landade på låga 696 kr eftersom att företaget fanns till finansiering för projektet med bl.a. en Raspberry Pi. I Bilaga 4 finns en lista med de komponenter som finns med på I/O-kortet och vad dessa kostat.

5 Diskussion

Resultatet från testerna, utförda i projektet, ger intrycket av ett tillförlitligt testsystem. Detta för att testsystemet bl.a. efterliknar NTC-motstånden i värmepumpen bra, då medelkvadratfelet endast är 0.24°C samt att det maximala felet är 1.3°C på mätpunkterna. Detta resultat står sig ganska bra jämfört med kravet som finns i det relaterade arbetet " Research on Thermistor Simulation Based on Digital Potentiometer for Microsatellites"[4]. Kravet i det relaterade arbete var att det simulerade temperaturvärdet maximalt får avvika 1.23°C från den önskade temperaturen. Testsystemet som har utvecklats under detta projekt känns även som ett mer lättanvänt system jämfört med det relaterade arbetet "Utveckling av en testmiljö för Thermias värmepumpar"[3]. Detta bl.a. för att testsystemet som har utvecklats i den här rapporten består endast av ett och samma system, vilket jämfört med det relaterade arbetet som består av två system.

Svagheterna som testsystemet har är att ingen detektering finns om kompressorn är igång eller inte. Detta gör att krav 13 i kravspecifikationen inte kan uppfyllas. I projektet genomfördes en grundlig undersökning för att få med kompressorn i testsystemet. Detta genomfördes m.h.a. elschemat på värmepumpens styrkort där signalen kunde identifieras vart den fanns. I samband med detta konstaterades att om inte hela värmepumpssystemet är anslutet är det väldigt svårt att detektera kompressorsignalen. Styrkorna i projektet är främst noggrannheten som testsystemet har på temperaturerna och även att dem på ett bra och smidigt sätt kan simulera ex. en dags temperaturer. Detta är något som tydligt syns i de resultat som har fåtts.

Testsystemet är bra ur miljöperspektiv då dess effektivitet och pålitlighet att genomföra tester på EasyServs diagnosverktyg är god. Detta medför att Easyserv kan optimera diagnosverktyget och på ett bättre sätt övervaka värmepumpar med mer exakthet. Genom ett bättre sätt att övervaka värmepumpar går det snabbare och enklare att se när en värmepump används ineffektivt eller är sönder. Datahistoriken som används för att simulera en sommar- och vinterdag i projektet, har valts från en slumpartad värmepump, vilket innebär att det inte finns någon risk att någon persons integritet kränks.

6 Slutsats

Projektet har uppnått de mål som sattes upp i början av projektet. Detta då testsystemet som utvecklats enkelt kan kopplas ihop och simulera de temperaturgivarna som värmepumpens styrkort vill få in. Bakgrunden till projektet är önskan från EasyServ att utveckla ett effektivt testsystem. Det effektiva testsystemet möjliggör att de kan optimera sitt diagnosverktyg mer. Testningen EasyServ genomför idag bygger endast på användning av bl.a. mekaniska potentiometrar och NTC-motstånd. Denna testningsmetod är både osäker och invecklad då många temperaturgivare behandlas. Genom projektet har en grund skapats för EasyServ att testa deras diagnosverktyg på ett effektivt och bättre sätt än tidigare. EasyServ kan m.h.a det utvecklade testsystemet enkelt och snabbt ställa in flera digitala potentiometrar på samma gång med stor precision. De kan med konfigureringsfilerna simulera några enklare beteenden för en värmepump.

Syftet med testsystemet har varit att förbättra EasyServs diagnosverktyg. Detta genom att testa deras diagnosverktyg på ett mer effektivt sätt. De frågeställningar som ställdes utifrån syftet i början av projektet har, genom en kunskapsfördjupning inom projektets berörda områden, gjort det möjligt att besvara alla frågor. Genom en fördjupning i simulering bestämdes det att designen av testsystemet skulle bestå av en HIL-uppsättning bestående av: Raspberry Pi, I/O-kort och ett värmepump-styrkort. Kommunikationsgränssnittet som valts till detta är I^2C, då detta skapade goda vidarutvecklingsmöjligheter för EasyServ. En annan förutsättning, för en eventuell vidarutveckling, finns i användningen av konfigureringsfiler som användargränssnitt, då dessa är väldigt enkla att modifiera. Simuleringsprogrammet skapades genom att programmera funktioner i Python, efter ett framtaget flödesschema från projektgruppen. Python, för övrigt, är också en förutsättning för EasyServ, då mycket av deras programmering sker i detta språk. Strategin som används för att efterlikna värmepumpens temperaturgivare blev användning av digitala potentiometrar. Metoden för konstruktion av I/O-kort blev att designa hur det skulle se ut och beställa.

Vidareutveckling

Testsystemets potential att vidareutvecklas är stor eftersom att I/O-kortet möjliggör inkoppling av fler I^2C-kompatibla enheter. Andra potentiella vidareutvecklingar är att på något sätt återkoppla kompressor- och växelventilssignalen till testkortet. Detta skulle göra testsystemet mer likt ett riktigt värmepumpssystem vilket leder till en bättre simulering. På mönsterkortet hade förbättringar som att skapa en gemensam jordpunkt mellan testkortet och CAN-bussen kunnat göras. Även resterande uppkopplingar mellan de olika korten kan förbättras. Lägre temperaturer än -5°C går enkelt att simuleras då den utvalda digitala potentiometern ingår i en serie-familj där motstånden 50 kOhm och 100 kOhm ingår. Detta medför dock en nackdel i att

noggrannheten på temperaturen försämras. En annan potentiell vidareutveckling är att utveckla ett mer användarvänligt användargränssnitt där det ex. går att ändra lite inställningar för hur simuleringen ska genomföras. Detta skulle kunna ske m.h.a inkoppling av en keyboard och att skapa ett GUI.

7 Referenser

Referenser

[1] Sverige världsledande på värmepumpar - men hur miljövänliga är de?
Använd 2016-09-18
`http://sverigesradio.se/sida/artikel.aspx?programid=3345I&artikel=4353594`

[2] Solenergi ur marken och luften,
Använd 2016-09-18.
`http://www.thermia.se/varmepump-kunskap/hur-fungerar-en-varmepump/varmepump-solene`

[3] Johan Gustafsson och Sebastian Witkowski, "Utveckling av en testmiljö för Thermias värmepumpar", Institutionen för teknik och naturvetenskap, Linköpings universitet, C-uppsats, 2007-03-01.

[4] Z. Guangquan, Z. Jiwei, X. Sirui and X. Wei, Research on Thermistor Simulation Based on Digital Potentiometer for Microsatellites, 2014 Fourth International Conference on Instrumentation and Measurement, Computer, Communication and Control, Harbin, 2014, pp. 31-35. doi: 10.1109/IMCCC.2014.15

[5] Ordlista
Använd 2016-09-21.
`http://www.energikunskap.se/FAKTABASEN/Ordlista/#V/`

[6] Energimyndigheten, Välj rätt värmepump, ET2010:02, 2010 .

[7] Statens energimyndighet, Värmepumparnas roll på uppvärmningsmarknaden Utveckling och konkurrens i ett föränderligt energisystem, ER2015:09, 2015, sid.168.

[8] Lindsay Porter, *The Renewable Energy Home Handbook.*
Veloce Publishing Ltd, 1 mars 2015

[9] Värmekurva,
Använd 2016-10-11.
`http://www.nibe.se/support/FAQ1/FAQ-bas-Service/Varmekurva/`

[10] Jerry Banks,Handbook of Simulation: Principles, Methodology , Advances, Applications, and Practice, John Wiley & Sons, 14 sep 1998
Definition of Simulation sid. 3-4, Verification, validation and testing techniques sid. 371

[11] Burdett, Arnold BCS Academy Glossary Working Party Bowen, Dan Mett, Percy. BCS Glossary of Computing and ICT [Internet]. : BCS Learning & Development Limited; 2013. [Citerad: 2016 December 9]. Tillgänglig från: ProQuest Ebook Central, s.105-108

[12] Computational Science: Ensuring America's Competitiveness. President's Information Technology Advisory Committee. June 2005

[13] R. Isermann, J. Schaffnit, S. Sinsel, Hardware-in-the-loop simulation for the design and testing of engine-control systems, Control Engineering Practice, 1999, Vol.7, Nummer 5. Introduction sid. 1, Hardware-in-the-loop simulation sid. 644-645.

[14] Software in the Loop for Embedded System Test(SiLEST)
Använd 2016-10-06.
`http://www.dlr.de/sc/en/desktopdefault.aspx/tabid-1262/1765_read-3186/`

[15] Ledin, Jim A., Hardware-in-the-loop simulation, Embedded Systems Programming 12 (1999): 42-62.

[16] Kendall. I.R, Jones. R.P, An investigation into the use of hardware-in-the-loop simulation testing for automotive electronic control systems, Control Engineering Practice, 1999, Volym 7, Nummer 11

[17] Embedded Computer Systems
Använd 2016-12-11.
`http://www.ece.ncsu.edu/research/cas/ecs`

[18] What is Arduino? *Använd 2016-10-05.*
`https://www.arduino.cc/en/Guide/Introduction`

[19] Raspberry Pi
Använd 2016-10-09.
`https://www.raspberrypi.org/help/faqs/#intro`

[20] Arduino UNO
Använd 2016-10-05.
`https://www.arduino.cc/en/Main/ArduinoBoardUno`

[21] Digital potentiometer
Använd 2016-09-28.
`http://www.resistorguide.com/digital-potentiometer/`

[22] Analog Devices Inc. Engineeri, Data Conversion Handbook(1), Saint Louis, US: Newnes, 2004, ProQuest ebrary, Web.10 okt 2016 (s.581)

[23] I2C
Använd 2016-09-28.
`http://www.totalphase.com/support/articles/200349156`

[24] Özgun Ayaz, Wireless Low Power Data Acquisition Device Embedded design and application, Bachelor Thesis, School of Information Science, Computer and Electrical Engineering, Halmstad University, 2012 May

[25] Selecting Between I2C and SPI
Använd 2016-10-16.
`https://www.lifewire.com/selecting-between-i2c-and-spi-819003`

[26] Orcad
Använd 2016-12-11.
`www.orcad.com/`

[27] PCB Library Expert
Använd 2016-12-11.
`http://www.pcblibraries.com/LibraryExpert/`

8 Bilagor

Bilaga 1

.

Kravspecifikation

Version 0.1

Granskad Namn: Johan Persson Datum: 2016-12-12

Godkänd Namn: Kushtrim Veseli Datum: 2016-12-12

Inledning

Alla krav som finns specificerade i detta dokument presenteras i tabeller likt den nedan. Den första kolumnen anger identifikationsnummer för kravet. Andra kolumnen anger status för kravet och kan vara original, revidering eller utgånget. Om statusen är reviderad eller utgånget återfinns även datum för beslut i andra kolumnen. Tredje kolumnen ger en kort beskrivning av kravet. I den sista kolumnen finns kravets prioritet där 1 har högst prioritet.

Krav nr x	Kravtext för krav nr x	**Prioritet**

Parter

Projektets kund är EasyServ AB. Handledaren för projektet är Stefan Byttner och projektet utförs av 2 studenter från högskolan i Halmstad med inriktning elektroingenjör.

Syfte och mål

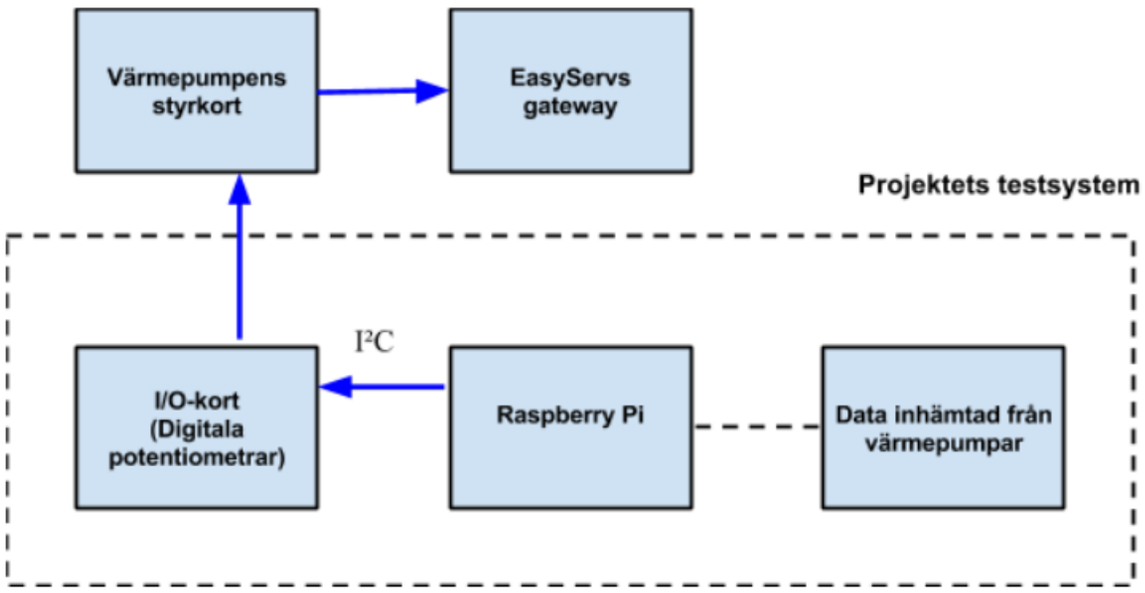
Syftet med projektet är utveckling av ett effektivt testsystem åt EasyServ. Genom ett effektivare och även ett pålitligare testsystem kan EasyServ kvalitetssäkra sitt diagnosverktyg bättre och snabbare. Detta leder till att deras diagnosverktyg kan analysera och övervaka värmepumpar mer exakt, vilket i sin tur gör att värmepumparna används mer optimerat. Ett annat syfte med testsystemet är underlättande av testning, detta då man undviker att ex. köpa in värmepumpar för testning vilket leder till stora kostnader och är utrymmeskrävande. På så vis undviker man även möjligheten att en värmepump förstörs under test.
Målet är att utveckla en simulator för värmepumpar som går att använda och vidareutveckla i EasyServs testmiljö.

Bakgrund

I dagsläget är EasyServs större problem att åstadkomma ett effektivt testsystem för deras kvalitetssäkring. En stor anledning till det är att värmepumpar är dyra att köpa in och utrymmeskrävande. Därför har EasyServ inte möjlighet att ha värmepumpar i sitt labb. Lösningen de har idag vid testning är användning av värmepumpens styrkort, tillhörande I/O-kort och ansluta diverse elektronik för simulering av värmepumpens beteende. Lösningen är en tidsödande och opålitlig process, främst då det är många signaler som hanteras. EasyServ vill kringgå problemet genom konstruktion av ett mer effektivt och pålitligt testsystem. För att åstadkomma detta

kommer projektet bestå av att konstruera ett system vilket på ett bra sätt ska simulera en värmepumps beteende. Simuleringen bygger på användning av historik ifrån befintliga värmepumpars signaler.

Översikt av systemet



Figur 14: Grov beskrivning av testsystemet

I projektet ingår utveckling av dem delarna som finns inne i lådan Testsystem som syns i Figur 3 vilket är en översikt av systemets design. Testsystemet använder historik från befintliga värmepumpars givarsignaler. I hårdvaruplattformen används historiken för simulering av värmepumpens beteende och sänder dessa signaler till I/O-kortet via ett gränssnittet. I/O-kortet omvandlar signalerna till en signal för att värmepumpens styrkort ska kunna läsa och förstå datan. I styrkortet sitter elektronik som tolkar värmepumpens beteende och genomför åtgärder vid behov och kan även återkoppla till I/O-kortet om ex. kompressorn ska starta. I/O-kortet i sin tur återkopplar via digitala signaler till hårdvaruplattformen. Gatewayen inhämtar information från värmepumpens styrkort om bl.a. givarnas signaler och är den information som Easyservs diagnosverktyg bygger på.

Testsystemets komponenter

Testsystemet består av en hårdvaruplattform, ett I/O-kort och ett värmepumps styrkort.

Beroenden till andra system

Har inga beroenden till andra system.

Ingående delsystem

Elektronikkonstruktion
Programmering

Avgränsningar

Projektet avgränsas från att simulera tryckgivarna i värmepumpen. I projektet kommer ett grafiskt gränssnitt inte prioriteras utan är ett extra moment om tid finns.

Designfilosofi

Testsystemet ska på ett enkelt och effektivt sätt genomföra tester av EasyServs diagnosverktyg. Testerna bygger på att simulera olika dagar men även olika fel.

Generella krav på hela systemet

Krav nr 1	Testsystemet ska utifrån historik simulera temperaturgivarna som finns i värmepumpar	**1**
Krav nr 2	Testsystemet ska bestå av ett inbyggt system	**1**
Krav nr 3	Testsystemet ska kunna kommunicera med värmepumpens styrkort	**1**
Krav nr 4	Testsystemet ska kunna simulera normal sommardag, normal vinterdag och ett felaktigt driftläge	**1**
Krav nr 5	Testsystemets olika driftlägen skall kunna modifieras	**1**

Delsystem - Elektronikkonstrukton

Inledning

Konstruktionen av elektroniken påbörjas med undersökning av vilka komponenter som ska väljas. De komponenter som krävs för konstruktionen är en hårdvaruplattform och övrig elektronik som krävs för testsystemets funktionalitet. Om tiden finns kommer testplattan ersättas av ett kretskort som antingen skickas in för tillverkning eller konstrueras på högskolan.

Gränssnitt

Krav nr 6	Omvandla signalen från hårdvaruplattformen till en signal som värmepumpens styrkort förstår	**1**
Krav nr 7	Kommunikationen mellan hårdvaruplattformen och I/O-kortet ska ske med hjälp av en buss	**1**

Funktionella krav för delsystem elektronikkonstruktion

Krav nr 8	Givarna som simuleras ska efterlikna 4,7 Kohm NTC-motstånd inom området -5°C till 90°C	**1**
Krav nr 9	Givarna som simuleras ska minst ha 2°C upplösning	**1**
Krav nr 10	Kommunikationen som används ska enkelt kunna modifieras för vidareutveckling	**1**

Delsystem - Programmering

Inledning

Efter att valet av hårdvaruplattform gjorts väljs en mjukvara för kodning. Mjukvaran som utvecklas i projektet ska främst simulera en värmepumps beteende, detta genom användning av historik från befintliga värmepumpars givarsignaler.

Gränssnitt

Krav nr 11	Skicka simulerade givarsignaler till I/O-kort	**1**
Krav nr 12	Kunna ta emot signaler från styrkortet	**1**

Funktionella krav för delsystem programmering

Krav nr 13	Koda funktion som kan läsa in txt-filer	**1**

Vidareutveckling

Krav nr 14	Allt arbete skall dokumenteras noggrant	**2**

Tillförlitlighet

Krav nr 15	Testsystemet ska ha godkänts på samtliga punkter i testspecifikationen	**1**

Ekonomi

Krav nr 16	Varje person ska lägga minst 400 timmar	**1**

Leveranskrav

Krav nr 18	Vid slutleverans till kund skall minst alla krav med prioritet 1 vara uppfyllda i enlighet med denna kravspecifikation	**1**
Krav nr 18	Efter avslutat projekt skall en demonstration av testsystemet genomföras för företaget	**1**

Underhåll

Krav nr 18	All källkod lämnas till EasyServ vid slutleverans	**1**
Krav nr 18	Alla kretsscheman lämnas i digital form till EasyServ vid slutleverans	**1**

Bilaga 2

.

Testspecifikation

Version 0.1

Granskad Namn: Johan Persson Datum: 2016-12-13
Godkänd Namn: Kushtrim Veseli Datum: 2016-12-13

Inledning

Testspecifikationen beskriver de test som ska utföras på enskilda delsystem samt på hela testsystemet. Dessa utförs bland annat på funktionaliteten och designen för att se till att testsystemet klarar att uppfylla de krav som ställts i kravspecifikationen(se Bilaga 1). Testprotokollet beskriver hur testaren har gått tillväga och vilket resultat testet gav.

Ej godkända test

Tester som inte godkänns ska diskuteras vid gruppmöten där beslut om testmetodens giltighet tas. Om metoden anses felaktig så ska en ny testmetod utformas. Anses testmetoden korrekt ska det diskuteras om kravet inte går att uppfylla och EasyServ kontaktas för beslut om möjliga ändringar.

Delsystem - Elektronikkonstruktion

Test nr 1	Kontroll av att givarna ska kunna simulera temperaturområdet -5°C till 100°C för ett 4,7 Kohms NTC-motstånd	Berör krav: 6,9
Test nr 2	Testsystemet ska visa temperatur från -5 till 90°C	Berör krav:8
Test nr 3	Testsystemet ska ha en upplösning på minst 2°C	Berör krav:9
Test nr 4	Kontrollera att I^2C-switchen kan skicka data till de digitala potentiometrarna.	Berör krav:7
Test nr 5	Testsystemet ska använda txt-filer som simulerar normal sommardag, normal vinterdag och felaktigt läge	Berör krav:4

Delsystem - Programmering

Test nr 4	Kontrollera att I^2C-switchen kan skicka data till de digitala potentiometrarna.	Berör krav:7
Test nr 5	Testsystemet ska använda txt-filer som simulerar normal sommardag, normal vinterdag och felaktigt läge	Berör krav:4
Test nr 6	Funktionstest av potentiometern.	Berör krav:7
Test nr 7	Funktionstest av I/O-kort	Berör krav:3,7,11

Tester

Test 1, 2, 3:

Kontrollera att potentiometrarna efterliknar NTC-motstånd på 4.7 kOhm. Under samma test kontrolleras testsystemets upplösning och räckvidd i temperatur.
Testet har utförts på en potentiometer, där resistansen uppmättes på I/O-kortets utgångar och där den skickade temperaturen kontrolleras på värmepumpens skärm. Testet utfördes genom att öka temperaturen som skickas in med 5°C per gång och där mätningen skedde mellan -5 till 90°C. Ändrade i formeln(-0.5°C) för 45°C och lägre, plottat upp en tabell nedan.

Test 4:

Kontrollera att I^2C-switchen kan skicka data till de digitala potentiometrarna. Genom att koppla I/O-kortet till Raspberry Pi via I^2C och programmera så undersöktes varje digital potentiometer för sig och det gick att ändra värde.

Test 5:

Testsystemet ska använda txt-filer som simulerar normal sommardag, normal vinterdag och felaktigt läge Skapat två olika txt-filer(Winter edition.txt , Summer edition.txt) baserade på historik och båda kan läsas av programmet.

Test 6:

Funktionstest av potentiometer Testet, för undersökning av funktionen på den digitala potentiometern, genomfördes med uppkoppling enligt tillhörande datablad(Se Bilaga 5). Genom Raspberry Pi:s I^2C pinnar programmerades i Python för att skicka ut önskad signal. Signalen som skickades undersöktes m.h.a. ett oscilloskop som kopplades till de två olika signalerna, vilka är SCL och SDA. För att veta vilken signal som ska skickas ut användes databladet. Testets syfte var att undersöka och skaffa sig bekantskap av hur I^2C och digitala potentiometrar fungerar. Där själva testningen gick ut på att ändra motståndsvärdet för den digitala potentiometern och för kontroll att motståndet ändrar sig användes en digital multimeter.

Test 7:

Funktionstest av I/O-kort Testsystemets funktionalitet testades genom att först detektera I^2Cbuss-switchen och därefter alla digitala potentiometrar. Det sista testet genomfördes för kontroll av att I/O-kortet klarar ändra samtliga digitala potentiometrars motståndsvärden Funktionaliteten godkänns då de olika testerna har klarats av. Detta medför att krav 3, 7 och 11 i kravspecifikationen(se Bilaga 1) är uppfyllda.

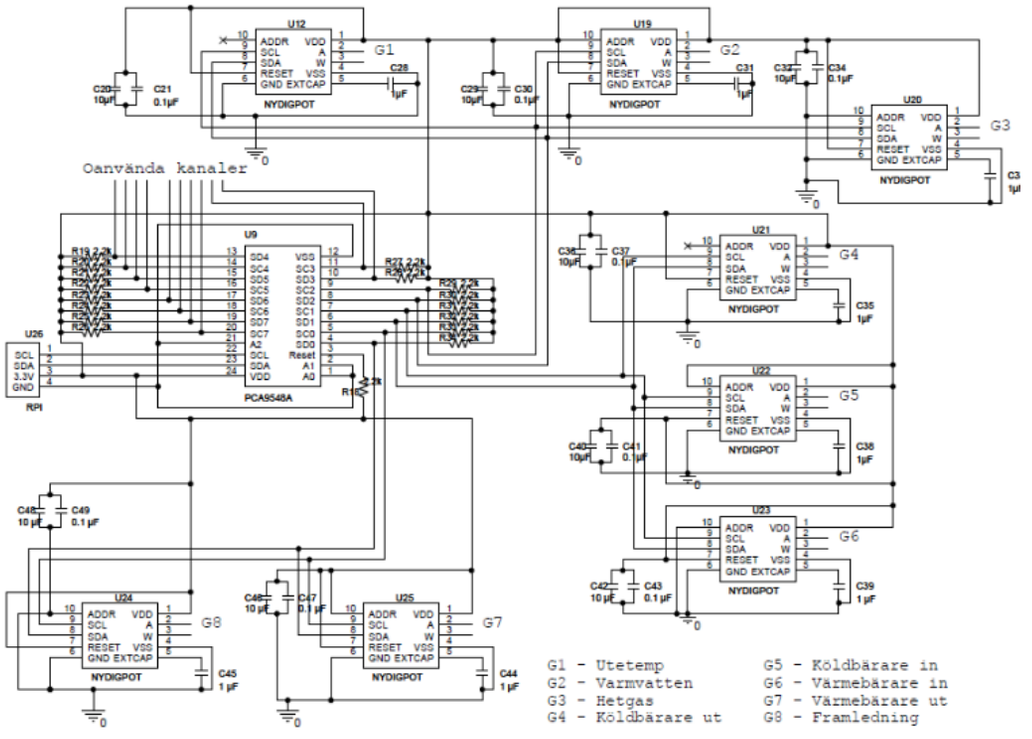
Acceptanstest

För att veta när testsystemet fungerar korrekt och uppfyller kraven så har ett acceptanstest skapats. Acceptanstestet utför en serie handlingar, dessa är följande:

- Koppla upp värmepump-styrkort till EasyServ gateway
- Inläsning av txt-fil(Sommar eller vinter)
- Omvandla från temperatur till datavärde
- Starta uppdatering av datavärde(15 minuters intervall)
- Plotta värden på EasyServ diagnosverktyg under en dag

Resultatet av acceptanstestet blev enligt Figur 13, där man ser ett utdrag av en sommardags och en vinterdags simulerade temperaturer. Endast tre temperaturer finns plottade, då kompressorsignalen inte kunde triggas igång. De stora skillnaderna, som även syns i grafen, är bl.a. att framledningen har en högre temperatur på vintern. Detta för att kunna ha en stabil inomhustemperatur på vintern, som på sommaren. Detta visar på att en simulering av temperaturgivarna kan genomföras med testsystemet och att krav 1(se Bilaga 1) därmed blir uppfyllt.

Bilaga 3 - Kretsschema



Bilaga 4 - Komponentlista

Komponent:	**Antal:**	**Pris:**
Ytmonterat chipmotstånd(2.2Kohm, 200 V, 500mW, ± 1%, 1206)	17st	1.97 kr/st
SMD keramisk kondensator(0.1µF, 50 V, ± 10%, 1206)	8st	0.673 kr/st
SMD aluminium-elektrolytkondensator(10µF, 10 V)	8st	2.97 kr/st
SMD aluminium-elektrolytkondensator(1µF, 50 V)	8st	1.83 kr/st
Digital potentiometer(20Kohm, Enkel, I²C, Linjär, ± 1%, 2,7 V)	8st	34.35 kr/st
Stiftlist (2 stift, 4 stift, 6 stift)	6st	-
PCB-kort(10*10cm)	5st	345 kr
Totalsumma:	-	**696 kr**

Komponent:	**Antal:**	**Pris:**
Raspberry PI 1 model B 512mb	1st	Tillhandahålls av EasyServ

Bilaga 5 - Datablad

Digital potentiometer(Analog devices AD5272):

http://www.analog.com/media/en/technical-documentation/data-sheets/AD5272_5274.pdf

I^2C-buss switch(NXP Semiconductors PCA9548A):

http://www.nxp.com/documents/data_sheet/PCA9548A.pdf

Bilaga 6 - Givartabell för NTC-motstånd 4,7Kohm

Temperatur (°C)	kΩ
-40	154,300
-35	111,700
-30	81,700
-25	60,400
-20	45,100
-15	33,950
-10	25,800
-5	19,770
0	15,280
5	11,900
10	9,330
15	7,370
20	5,870
25	4,700
30	3,490
35	3,070
40	2.510
45	2,055
50	1,696
55	1,405
60	1,170
65	0,980
70	0,824
75	0,696
80	0,590
85	0,503
90	0,430

Bilaga 7 - Beskrivning av temperaturgivarnas och komponenternas placering och funktion

Givare/Komponent	**Funktion/Placering:**
Hetgas	Givaren sitter vid kompressorn och mäter temperaturen på gasen ut från kompressorn
Köldbärare ut	Givaren sitter efter där köldmediumet går ut från förångaren och mäter temperaturen på köldmediumet efter att värme har överförts till systemet
Köldbärare in	Givaren sitter innan där köldmediumet går in i förångaren och mäter temperaturen på köldmediumet innan värme har överförts till systemet
Värmebärare in	Givaren sitter innan där den uppvärmda gasen går in till kondensorn och mäter temperaturen på gasen innan värmen överförts till värmesystemet
Värmebärare ut	Givaren sitter efter där gasen har kondenserats i kondensorn och mäter temperaturen på den kondenserade gasen efter att värme har överförts till systemet
Framledning	Givaren sitter innan mediumet som går ut till uppvärmningen till huset via element exempelvis där givaren mäter den temperaturen innan mediumet går ut till uppvärmningen
Varmvatten	Givaren sitter på väg ut till varmvattnet i systemet och mäter temperaturen på varmvattnet som går ut till systemets duschar och kranar
Ute	Givaren sitter utomhus och mäter utetemperaturen
Kompressor	Kompressorn är en av huvudkomponenterna i värmepumpen och funktionen är att vid behov öka temperaturen i systemet vilket leder till ökning av temperatur i antingen varmvatten eller framledning
Växelventil	Växelventilen är en komponent i värmepumpen som ändrar läge beroende på om temperaturen ska ökas på varmvattnet eller framledningen

Johan Persson och Kushtrim Veseli

HÖGSKOLAN
I HALMSTAD

Besöksadress: Kristian IV:s väg 3
Postadress: Box 823, 301 18 Halmstad
Telefon: 035-16 71 00
E-mail: registrator@hh.se
www.hh.se