Zonnecollectorsysteem & Datalogging

Auteur: Hessel Meulenbeld – 1584275

Eerste examinator: Joop Kaldeway

Bedrijf: TriOpSys

Managementsamenvatting

Als afstudeeropdracht voor de studie Technische Informatica is deze opdracht bij TriOpSys uitgevoerd. Deze opdracht bestond uit een onderzoek naar de mogelijkheid om een Raspberry PI te gebruiken als data-recorder voor een Siemens LOGO! PLC. Om de resultaten van dit onderzoek te steunen is een proof of concept ontworpen en ontwikkeld. Dit proof of concept stuurt een zonnecollectorsysteem aan en gebruikt een Raspberry PI als datarecorder.

Bij dit project is de volgende probleemstelling gedefiniëerd:

> Is het mogelijk om een kleine Linux gebaseerde computer, zoals de Raspberry PI, te gebruiken als data-recorder voor een Siemens LOGO! PLC?

Deze probleemstelling heeft geleid tot de volgende onderzoeksvragen:

> Wat zijn de interfacemogelijkheden van de Raspberry PI?
>
> Wat zijn de interfacemogelijkheden van de Siemens LOGO!?
>
> Hoe kan worden gecommuniceerd met de Siemens LOGO!?
>
> Wat voor informatie verstuurt de Siemens LOGO!?
>
> Wat is het data-format waarmee de Siemens LOGO! de data verstuurt?
>
> Welke alternatieven zijn er voor de RRD-tool(Round Robin Database)?

Uit de resultaten van het onderzoek is gebleken dat de Raspberry PI geschikt is om als data-recorder te gebruiken voor een Siemens LOGO! PLC. Beide hardwareonderdelen kunnen via dezelfde interface communiceren. Daarnaast biedt het protocol de mogelijkheid om de data op te vragen en deze data is duidelijk genoeg om gebruikt te kunnen worden door de data-recorder.

Om dit aan te kunnen tonen is een proof of concept gemaakt, waarbij een zonnecollector systeem wordt aangestuurd. De gegevens van dit systeem worden via een webserver op de Raspberry PI ontsloten. Uit de resultaten van het proof of concept is gebleken dat de Raspberry PI gebruikt kan worden als data-recorder. Het visualiseren van de verkregen gegevens met een webserver kan ook op de Raspberry PI plaatsvinden, maar dit geldt alleen als het aantal gebruikers van de webserver beperkt blijft.

Het is niet mogelijk om met de Raspberry PI de PLC aan te sturen. Dit komt doordat het protocol van de PLC niet toereikend is voor deze functionaliteit. Dit is met andere aansturingshardware wel mogelijk, er wordt daarom aanbevolen om een andere PLC te gebruiken als het nodig is om de PLC met de Raspberry PI aan te sturen.

Inhoudsopgave

1 Inleiding

Deze scriptie beschrijft de opdracht die ik heb uitgevoerd bij TriOpSys. TriOpSys is een ICT-bedrijf dat zich vooral richt op dataverzameling en datadistributie. Daarnaast ontwikkelt TriOpSys ook communicatiesystemen. Dit onderzoek valt onder dataverzameling en datadistributie. Hierbij wordt onderzocht of het mogelijk is om een Raspberry PI te gebruiken als data-recorder voor een Siemens LOGO! PLC.

Voor het proof of concept bij dit onderzoek moet met de Siemens LOGO! PLC een zonnecollectorsysteem van twee vakantiehuizen worden geregeld. Van deze PLC word door de Raspberry PI de gegevens uitgelezen en gelogd. Deze gegevens worden middels een webpagina met grafieken getoond. Dit proof of concept moet aantonen dat een kleine Linux machine zoals de Raspberry PI gebruikt kan worden om gegevens van een PLC zoals de Siemens LOGO! te loggen en te visualiseren.

Bij dit onderzoek is de volgende hoofdvraag gesteld:

Is de Raspberry PI geschikt als data-recorder voor een Siemens LOGO! PLC?

Deze hoofdvraag is opgedeeld in de volgende deelvragen:

1. Wat zijn de interfacemogelijkheden van de Raspberry PI?
2. Wat zijn de interfacemogelijkheden van de Siemens LOGO!?
3. Hoe kan worden gecommuniceerd met de Siemens LOGO!?
4. Wat voor informatie verstuurt de Siemens LOGO!?
5. Wat is het data-format waarmee de Siemens LOGO! de data verstuurt?
6. Welke alternatieven zijn er voor de RRD(Round Robin Database)?

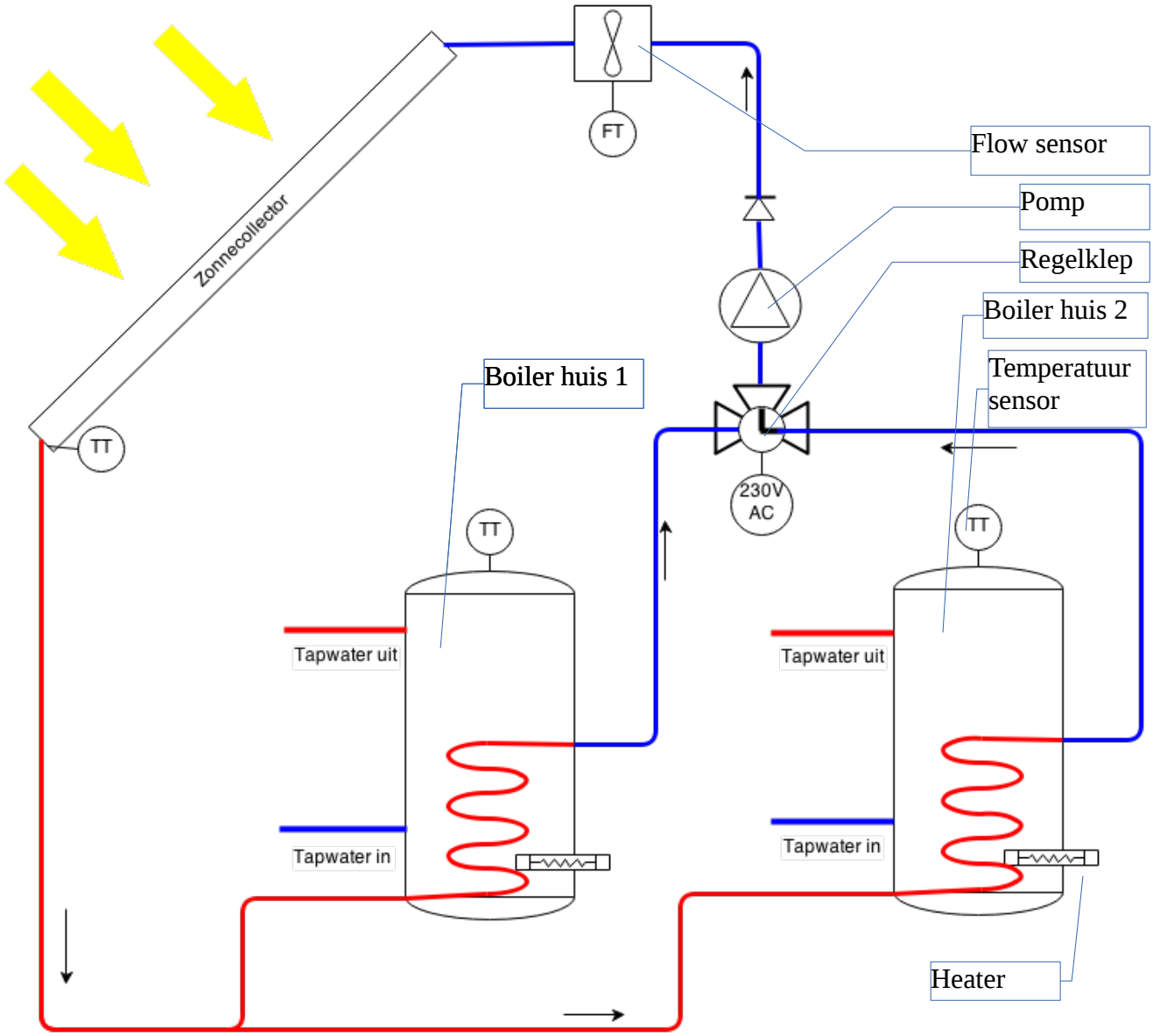
In dit document zullen onder andere de volgende onderdelen behandeld worden:

Het probleem en het doel van deze opdracht worden eerst beschreven. Waarna het bedrijf kort beschreven wordt. In het daaropvolgende deel worden de aanpak en de gebruikte tools nader toegelicht. De resultaten van het onderzoek en het proof of concept worden in het daaropvolgende hoofdstuk behandeld. Na de resultaten worden de implementatie en manier van testen per deel van het proof of concept beschreven. In het laatste hoofdstuk geef ik de conclusie en de evaluatie van het project en het onderzoek. Achter in het document is een woordenlijst te vinden waar alle afkortingen zijn toegelicht.

2 Probleemstelling en doel

In dit hoofdstuk worden het probleem en de context beschreven. Daarnaast wordt beschreven wat het doel van deze opdracht is en waarom deze opdracht wordt uitgevoerd.

2.1 Achtergrond



Figuur 1: Schematische weergave van het systeem
TT: Temperatuur sensor(Temperature Transducer)
FT: Flow sensor(Flow Transducer)

Een complex met twee woningen krijgt een installatie voor de warmwatervoorziening. Elke woning beschikt over een boiler van 300 liter die wordt verwarmd met zonne-energie. Daarom is een zonnecollector aanwezig waarvan de energie zodanig aan de boilers moet worden geleverd dat aan de comforteisen van de bewoners is voldaan. Voor het geval niet voldoende energie door de zonnecollector geleverd kan worden, zijn de boilers uitgerust met elektrische verwarmingselementen.

2.2 Probleemstelling

Is het mogelijk om een kleine Linux gebaseerde computer, zoals de Raspberry PI, te gebruiken als data-recorder voor een Siemens LOGO! PLC?

2.3 Kwestie

TriOpSys wil met behulp van de Siemens LOGO! PLC de output monitoren van bepaalde toepassingen. Deze gegevens moeten dan via een webserver te benaderen zijn. TriOpSys wil onderzoeken of het mogelijk is om de PLC te monitoren met een Raspberry PI. Dit wil TriOpSys testen door dit toe te passen op het zonnecollectorsysteem van twee vakantiehuizen.

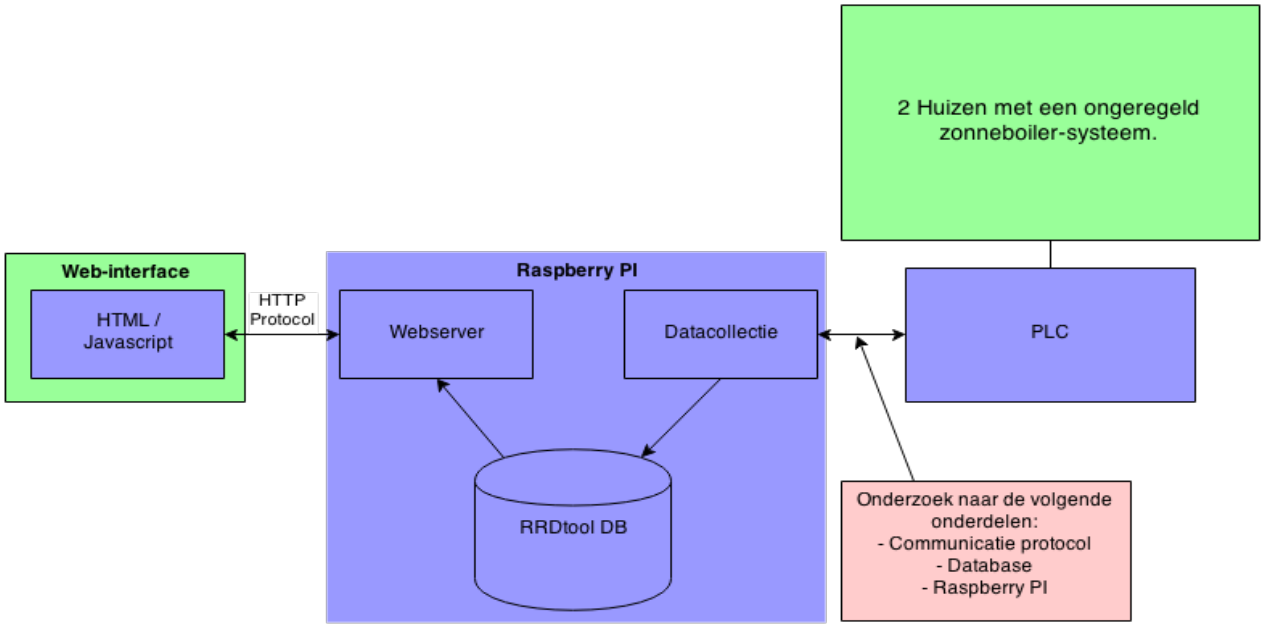
2.4 Opdrachtomschrijving

De opdracht die TriOpSys heeft gegeven, is het ontwerpen en ontwikkelen van software voor het regelen van een zonnecollector-systeem, de interface met het regelsysteem en een webserver. Daarnaast moet een testopstelling gebouwd worden om deze software te testen.

De data die het regelsysteem(PLC) genereert moet worden opgevraagd via de interface van de PLC en worden opgeslagen in een database. Omdat het communicatieprotocol tussen de PLC en de processing-unit(Raspberry PI) niet bekend is zal dit moeten worden uitgezocht en zal voor deze interface nieuwe software moeten worden geschreven.

Nadat de data is opgeslagen moet de gebruiker via een webbrowser de gegevens kunnen inzien. Het inzien van de gegevens moet mogelijk zijn via een webpagina met grafieken. Daarnaast kan dit uitgebreid worden met een simulatie/SCADA-model en een mobiele applicatie. De eigenaar en de bewoners moeten de gegevens ook op een lokaal display, dat gekoppeld is aan de PLC, kunnen inzien.

In de situatie zoals beschreven in 2.1 Achtergrond, zal door mij het volgende worden toegevoegd:



Figuur 2: Abstract schema van het systeem.
Groene onderdelen: Al bestaande onderdelen.
Blauwe onderdelen: Door mij toegevoegde onderdelen.
Rode onderdelen: Het door mij uitgevoerde onderzoek.

2.5 Designbeperkingen

De opdrachtgever heeft aan deze opdracht een aantal beperkingen gesteld:

1. Voor het regelsysteem moet gebruik worden gemaakt van een Siemens LOGO! PLC.
2. Voor de database moet gebruik worden gemaakt van een Round Robin Database.
3. De database en interface moeten worden gerealiseerd met een Raspberry PI.

2.6 Doelstellingen

Hieronder zijn de doelstellingen beschreven die bij deze kwestie zijn gesteld.

Warmteverdeling van de zonne-energie

De warmte, die van de zonnecollector komt, moet op een optimale manier verdeeld worden over de beide boilers. Hierbij moet zo min mogelijk warmte verloren gaan.

Warmtevoorziening van de boilers

De regeling moet zorgen dat de boilers de gewenste temperatuur bereiken, waarbij zo min mogelijk gebruik wordt gemaakt van de elektrische verwarmingselementen.

Bedieningsinterface

Het systeem moet een interface bieden waarmee de bewoningsstatus van de woningen ingesteld kan worden. Ook moet hiermee de gewenste temperatuur ingesteld kunnen worden.

Monitoren van de installatie

Het systeem moet mogelijkheden bieden om de procesparameters van het regelsysteem(PLC) en hun historie van maximaal twee jaar te bekijken. Dit moet zowel via een webbrowser en op een lokaal display kunnen.

2.7 Onderzoeksvragen

Uit de gestelde probleemstelling is de volgende hoofdvraag naar voren gekomen:

Is de Raspberry PI geschikt als data-recorder voor een Siemens LOGO! PLC?

Deze hoofdvraag is opgedeeld in de volgende deelvragen:

1. Wat zijn de interfacemogelijkheden van de Raspberry PI?
2. Wat zijn de interfacemogelijkheden van de Siemens LOGO!?

   Aan de hand van de antwoorden van deze vragen kan bepaald worden of de Raspberry PI en de Siemens LOGO! kunnen communiceren.
3. Hoe kan worden gecommuniceerd met de Siemens LOGO!?

   Het is belangrijk om te weten of het protocol het opvragen van data ondersteunt.
4. Wat voor informatie verstuurt de Siemens LOGO!?

   De gewenste informatie moet gestuurd worden voor het nut van de data-recorder.
5. Wat is het data-format waarmee de Siemens LOGO! de data verstuurt?

   De data die via het protocol verstuurd wordt moet bruikbaar zijn voor een data-recorder.
6. Welke alternatieven zijn er voor de RRD(Round Robin Database)?

2.8 Eisen

Het product moet voldoen aan de eisen zoals gespecificeerd in Bijlage B – SRS van Bijlage A – Plan van Aanpak. Een korte opsomming van deze eisen is hieronder weergegeven.

Eis	**Korte omschrijving**
CAP_UC01_01	Er moet kunnen worden ingesteld of een huis bewoond of onbewoond is.
CAP_UC01_02	De regeling moet voorzorgsmaatregelen nemen om te voorkomen dat de installatie wordt beschadigd.
CAP_UC01_03	De boiler van het meest bewoonde huis moet als eerste verwarmd worden als beide huizen bewoond/onbewoond zijn.
CAP_UC01_04	Als de eerste boiler de maximale temperatuur heeft bereikt moet de andere boiler verwarmd worden, hierdoor.
CAP_UC01_05	Tijdens het wisselen tussen de huizen moet de pomp uit zijn.
CAP_UC01_06	Bij het bereiken van de maximale temperatuur van de boiler moet de toevoer vanaf de zonnecollector gestopt worden.
CAP_UC01_07	Als de eerste boiler de ingestelde temperatuur heeft bereikt moet de andere boiler verwarmd worden.
CAP_UC01_08	Het electrische verwarmingselement moet gebruikt worden tot de boiler van het huis de gewenste temperatuur bereikt heeft.
CAP_UC01_09	De bewoner moet de temperatuur kunnen instellen.
CAP_UC02_01	(Specificatie van het gedrag van de klep)
CAP_UC02_02	(Specificatie van het gedrag van de verwarmingselementen)
CAP_UC02_03	(Specificatie van het gedrag van de pomp)
CAP_UC02_04	Er moet voorkomen worden dat de actuatoren pendelend gaan schakelen.
CAP_UC04_01	De gegevens moeten via grafieken kunnen worden bekeken.
CAP_UC04_02	(Specificatie van de gegevens die moeten worden opgeslagen)
CAP_UC04_03	De gebruiker moet via de interface de tijdsspanne van de grafieken kunnen instellen.
CAP_UC04_04	De actuele gegevens van het proces moeten op een lokaal display worden getoond.
CAP_UC05_01	De klok van de PLC moet gelijk lopen met de klok van de processing unit.
CAP_UC05_02	De te gebruiken bron voor de klok moet zo nauwkeurig mogelijk zijn.
CAP_UC05_03	De getoonde tijd moet de lokale tijd van de installatie zijn.
WEB_GUI_01	Het te gebruiken protocol is HTTP.
WEB_GUI_02	De GUI moet vrij toegankelijk zijn.
WEB_GUI_03	De koppeling met het internet moet via ethernet.
IDR_01	De tijdsspanne van de database moet twee jaar zijn.
IDR_02	De resolutie van de datalogging moet één minuut zijn.

3 Het bedrijf

“TriOpSys B.V. is in 1998 opgericht door 3 vrienden: At Hijwegen, Hans Schulz en Rob Timman. Ook voor de start van TriOpSys werkten zij al vele jaren samen, in een soortgelijk bedrijf. Het uitgangspunt bij de oprichting was een bedrijf te starten, waar zij zelf ook graag zouden willen werken, met technisch uitdagende projecten en waar kwaliteit, integriteit en collegialiteit hoog in het vaandel zouden staan. Vanaf het begin zijn deze waarden leidend geweest en dat heeft geresulteerd in prachtige, succesvolle projecten en een gestage groei van ons bedrijf. Momenteel werken we met ongeveer 70 mensen zowel vanuit ons kantoor in Utrecht als op locaties bij klanten. TriOpSys is een financieel zeer gezond, onafhankelijk en (bewust) niet beursgenoteerd bedrijf. Daardoor bent u verzekerd van een IT-partner, met wie u een langetermijnrelatie kunt aangaan. En de praktijk heeft dat ook bewezen: veel bedrijven doen al lang zaken met TriOpSys, sommigen sinds de oprichting in 1998.”(Over ons, n.d.)

De activiteiten van TriOpSys:

• Softwareontwikkeling & Systeemintegratie

• Advies op het gebied van architectuur en analyse

• Projectmanagement

• Testen & Audits

• Servicedesk & Systeembeheer

4 Aanpak

In dit hoofdstuk wordt de gebruikte aanpak beschreven.

4.1 Scrum

Bij dit project heb ik gebruik gemaakt van scrum in aangepaste vorm. Deze aanpassingen zijn hieronder kort beschreven met de reden waarom ze zijn doorgevoerd.

1. Geen daily-scrum.

   Omdat dit project door één persoon is uitgevoerd, is het niet nuttig om een daily-scrum te houden. Tijdens de daily-scrum bespreken de teamleden de voortgang met elkaar. Dat kan bij een eenmansproject niet. Dit is opgelost door wekelijks met een ander team een weekly-scrum te houden. Hierbij kan dat team feedback geven op de problemen en voortgang van het project.

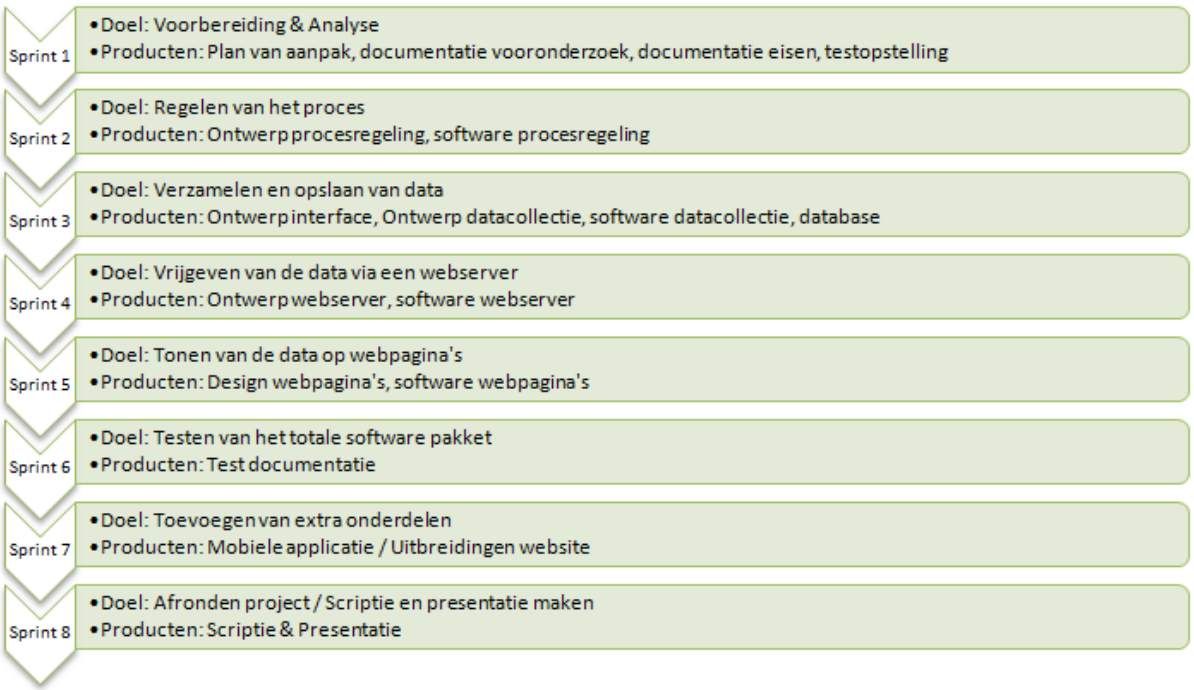
2. Andere werkvolgorde.

   Bij scrum is het normaal om bij een gelaagd systeem sprints verticaal op te delen, zodat er aan het eind van elke sprint nieuwe functionaliteit kan worden geleverd. Bij dit project kan op elk niveau binnen het lagenmodel iets zichtbaar worden gemaakt, omdat elke laag een apart stuk hardware is. Daarom is het niet nodig geweest om dit verticaal op te delen.

3. Voorbereidende sprint.

   Omdat vanuit school en het bedrijf nog veel documentatie moet worden opgeleverd is een voorbereidende sprint toegevoegd.

Het project is opgedeeld in acht korte sprints. Dit is te zien in Flowchart 1. De bedrijfsbegeleider is de product-owner en klant.



Flowchart 1: Flow van het project

Aan het eind is een sprint toegevoegd voor eventuele uitbreidingen en uitloop van de andere sprints. In de laatste sprint is extra veel tijd beschikbaar gemaakt om aan de scriptie te werken. Daarnaast is in elke sprint tijd ingepland om aan de scriptie te werken.

Tijdens elke sprint wordt ook veel aandacht besteed aan de documentatie voor het bedrijf. Deze documentatie moet voldoen aan de door het bedrijf gebruikte standaard[1]. Hieronder vallen de eisen van het systeem en de ontwerpen die daarvan gemaakt zijn.

4.2 Sprint opbouw

Elke sprint is opgedeeld in een aantal deelfasen. Een aantal van deze deelfasen komt één keer voor per sprint. Andere deelfasen komen meerdere keren voor. Elke sprint wordt begonnen met een planning en een vooronderzoek. Daarna zullen de volgende fasen herhaaldelijk worden uitgevoerd: ontwerpen, ontwikkelen en testen. Aan het eind van elke sprint is een opleveringsmoment.

4.2.1 Planning

Elke sprint wordt begonnen met een korte planning waarin wordt besloten wat in die sprint wordt gedaan. Deze planning hoort officieel nog niet tot de sprint maar wordt hier wel toegevoegd voor het overzicht. Tijdens deze planning worden de taken op prioriteit gesorteerd zodat de belangrijkste taken eerst worden gedaan.

4.2.2 Vooronderzoek

Bij elke sprint is een kort vooronderzoek gepland waarin de nieuwe hardware en software worden onderzocht. Deze fase kan worden overgeslagen als alles voor deze sprint al duidelijk is. Ook kan al begonnen worden aan de ontwerp- en ontwikkelfase omdat dit vaak veel inzicht in de taak geeft waardoor verder onderzoek niet nodig blijkt.

4.2.3 Ontwerpen

Nadat het vooronderzoek is afgerond wordt de nieuwe software ontworpen. Deze ontwerpen zijn alleen bedoeld als richtlijnen tijdens de ontwikkelingsfase. Deze ontwerpen zijn levende documenten die herhaaldelijk tijdens de sprint kunnen worden aangepast. Vaak gaan deze ontwerpen niet verder dan de basisstructuur van de functionaliteiten.

4.2.4 Ontwikkelen

De gemaakte ontwerpen worden ontwikkeld en daarmee wordt dus functionaliteit toegevoegd. Deze functionaliteit zal direct getest worden. De software kan dan aan de hand van nieuw ontdekte problemen worden aangepast. Nadat alle functionalteit is toegevoegd zal een nieuwe ontwerpfase worden gestart. Of als geen nieuwe ontwerpen nodig zijn, zal worden begonnen aan de testfase.

4.2.5 Testen

Aan het eind van de sprint zal de totale bundel van functionaliteiten nog een keer getest worden om zeker te zijn dat alles goed met elkaar kan samenwerken. Bij een fout tijdens het testen zal weer terug gegaan worden naar de ontwikkelfase. Als het probleem te groot is om nog in deze sprint op te lossen, wordt dit doorgeschoven naar de volgende sprint. Voor het testen worden de standaard testmethoden van het bedrijf toegepast. Hieronder vallen gebruikerstesten, functionaliteitstesten en unittesten. Voor uitgebreide informatie over het testen wordt verwezen naar hoofdstuk 8 Testen & Acceptatie.

1 Department of Defence, United States of America. (1995). J-STD-016 Standard for Software Development and Documentation

4.2.6 Opleveren

Naast het testen wordt aan het eind van de sprint de software ook opgeleverd. Deze software wordt aan de productowner geleverd voor goedkeuring. Deze zal de software dan presenteren aan de klant. Na het opleveren van de software zal direct worden begonnen aan de volgende sprint.

4.2.7 Documentatie

Gedurende elke sprint wordt documentatie gemaakt van de software. Dit zijn onder andere de ontwerpen. Daarnaast worden waar nodig handleidingen geschreven en andere benodigde documenten opgeleverd. Tijdens elke sprint wordt een korte periode beschikbaar gesteld waarin aan de scriptie gewerkt kan worden.

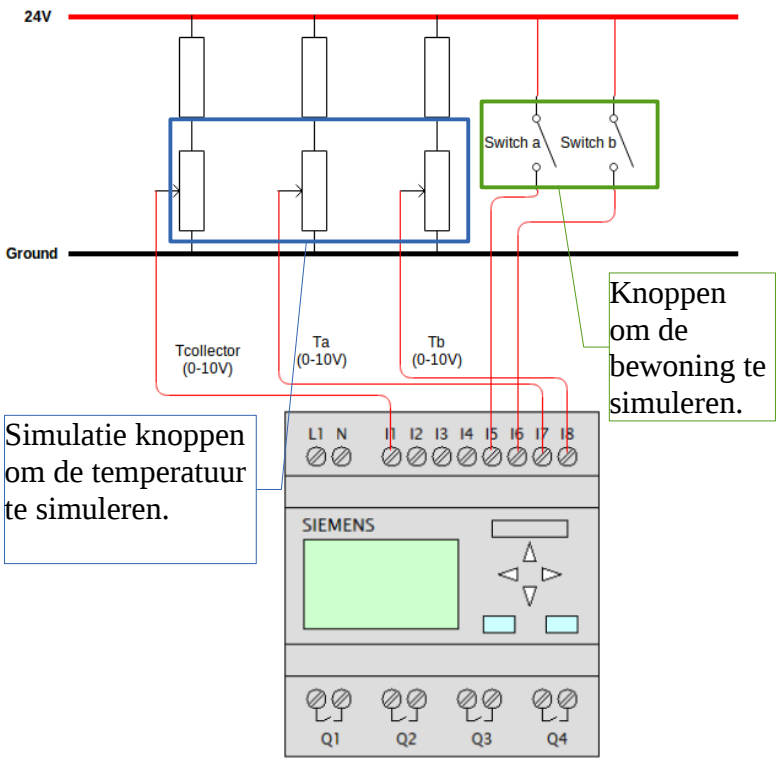
4.3 Sprints

In dit hoofdstuk wordt elke sprint kort behandeld.

4.3.1 Sprint 1: Voorbereiding en Analyse

Tijdens deze eerste sprint is begonnen om het eisenpakket voor dit project samen te stellen. Deze eisen zijn nodig om het plan van aanpak te kunnen schrijven. Ook moet het project ergens aan getoetst kunnen worden om te bepalen of het resultaat voldoet aan alle wensen en eisen.

In deze vroege fase is ook begonnen aan het plannen van het project. Hierbij is alleen de planning voor deze en de volgende sprint gemaakt en de globale planning van het gehele project. Ook is tijdens deze fase een scrumbord opgezet waarop met post-its de taken van de sprint worden weergegeven. Dit bord zorgt ervoor dat de productowner een snel overzicht heeft van het project en kan zien hoe ver het project al gevorderd is.



Figuur 3: Testopstelling

Na het plannen is begonnen met het opbouwen van de testopstelling zoals te zien in Figuur 3. Dit was nodig omdat tijdens de volgende sprint de testopstelling gebruikt wordt om de functionaliteit te testen. Deze testopstelling bestaat uit een aantal potmeters en switches die gekoppeld zijn aan de ingangen van de PLC. Deze PLC is gekoppeld aan de Raspberry PI of de PC van de programmeur om de code te testen en in te laden. Met deze testopstelling kunnen de ingangswaarden (temperatuur en bewoning van de huizen) van de PLC getest worden. In een later stadium kunnen de potmeters verwisseld worden met temperatuursensoren om een reëler beeld te krijgen.

Tijdens deze sprint is ook onderzoek gedaan naar de Raspberry PI. Dit onderzoek moest in een vroeg stadium van het project gebeuren omdat dan nog eventueel wijzigingen zouden kunnen worden doorgevoerd. Uiteindelijk heeft het onderzoek niet geleid tot wijzigingen binnen het project. Wel is tijdens dit onderzoek het communicatieprotocol van de PLC nader onderzocht.

Nadat het onderzoek was uitgevoerd is het plan van aanpak opgesteld. Hierin is globaal het project vastgelegd, met daarbij de eisen waaraan het product moet voldoen en de planning van het project.

Tijdens deze sprint zijn de volgende producten opgeleverd:

1. Plan van Aanpak(zie bijlage A)
2. Requirements documentatie(zie bijlage A)
3. Onderzoek naar de Raspberry PI(zie hoofdstuk 6.1 Onderzoekresultaten)
4. Onderzoek naar de interface(zie hoofdstuk 6.1 Onderzoekresultaten)
5. Onderzoek naar de RRDtool(zie hoofdstuk 6.1 Onderzoekresultaten)

4.3.2 Sprint 2: Proces

Tijdens deze sprint is gewerkt aan de software voor het regelen van het systeem. Dit is met de ontwikkelomgeving van Siemens gedaan. Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Ontwerp van het systeem(zie hoofdstuk 6.2.1)
2. Werkende software van het regelsysteem(zie hoofdstuk 6.2.3 en hoofdstuk 7.1)

4.3.3 Sprint 3: Datacollectie

Tijdens deze sprint is gewerkt aan de software voor de interface met het regelsysteem. Dit is gedaan in de programmeertaal C(zie hoofdstuk 9 voor de keuze motivatie). Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Ontwerp van de interface(zie hoofdstuk 6.2.4 en 6.2.5)
2. Werkende software voor de interface met het regelsysteem(zie hoofdstuk 7.2)

4.3.4 Sprint 4: Webserver

Tijdens deze sprint is gewerkt aan de software voor de webserver. Dit is gedaan in de programmeertaal Python. Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Ontwerp van de webserver(zie hoofdstuk 7.3)
2. Een werkende webserver

4.3.5 Sprint 5: Datavisualisatie

Tijdens deze sprint is gewerkt aan het visualiseren van de opgeslagen gegevens. Hierbij is gebruik gemaakt van HTML5, Javascript en jQuery. Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Webpagina met grafieken(zie hoofdstuk 6.2.6 en 7.4)
2. Extra webpagina met een scadasysteem(zie hoofdstuk 6.2.6 en 7.4)

4.3.6 Sprint 6: Tests

Tijdens deze sprint is het volledige systeem getest en zo nodig bijgewerkt. Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Testdocumentatie
2. Totaal softwarepakket met daarin alle bij eerder genoemde sprints opgeleverde producten

3. Handleiding

4.3.7 Sprint 7: Uitbreidingen

Tijdens deze sprint zijn nog extra uitbreidingen aan het systeem toegevoegd. Dit zijn onder andere een mobiele applicatie en extra uitbreidingen op de webpagina's. Aan het eind van deze sprint zijn de volgende producten opgeleverd:

1. Mobiele applicatie(zie 6.2.7 en 7.5)
2. Uitbreidingen op de webpagina's opgeleverd in sprint 5

4.3.8 Sprint 8: Scriptie

Tijdens deze sprint wordt gewerkt aan de scriptie en de presentatie. Aan het eind van deze sprint worden de volgende producten opgeleverd:

1. Scriptie
2. Presentatie

5 Tools

In dit hoofdstuk worden de belangrijkste tools beschreven die gebruikt zijn tijdens dit project. In de volgende paragrafen wordt per fase beschreven welke tools daarbij zijn gebruikt.

5.1 Onderzoektools

In dit hoofdstuk worden de tools beschreven die gebruikt zijn bij het uitgevoerde onderzoek.

5.1.1 LibreOffice Calc

Tijdens het onderzoek is veel gebruik gemaakt van LibreOffice Calc om rekenmodellen te maken. Calc biedt veel ondersteuning voor het verwerken van veel gegevens en om snel rekenmodellen op te stellen.

5.1.2 RRDtool (Zie 6.1.6.1)

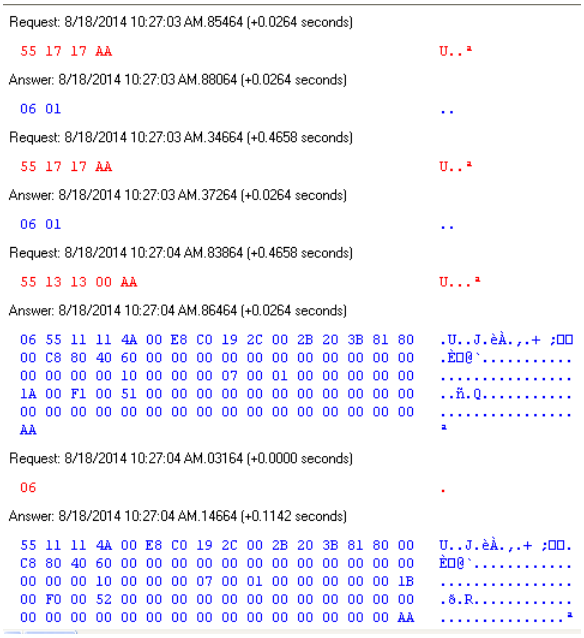
Tijdens het onderzoek naar de sensoren is de RRDtool gebruikt om de gegevens van de sensoren te plotten en zo snel de afwijking tussen twee sensoren te kunnen zien.

5.1.3 Siemens Logosoft

De testprogramma's voor de sensoren zijn met Siemens Logosoft gemaakt en in de LOGO! ingeladen.

5.1.4 Comport logger

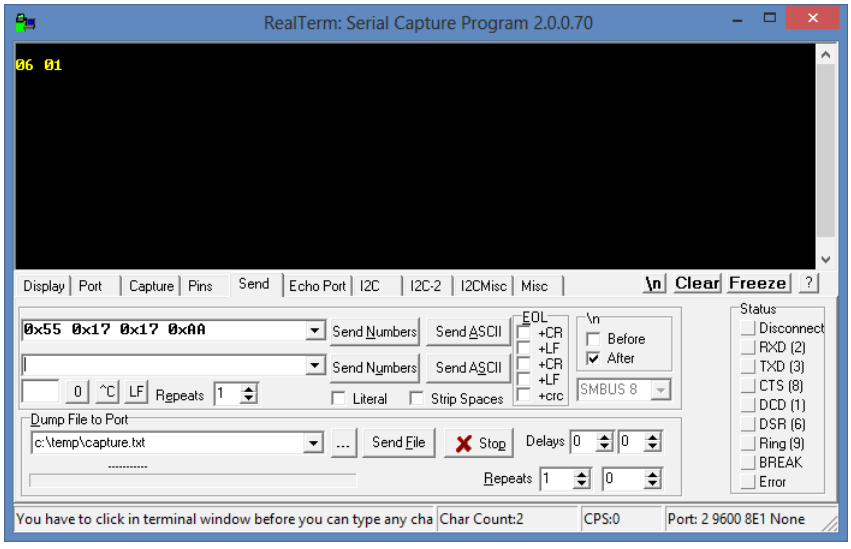
Om de interface met de PLC te loggen is gebruik gemaakt van een Comport logger. Deze logt de communicatie tussen de officiële software van Siemens en de PLC. De output hiervan is te zien in Figuur 4.



Figuur 4: Output comport logger

5.1.5 RealTerm

Met Realterm kunnen commando's naar een comport verzonden worden. Hiermee is getest of de interface die gelogd is ook echt werkt. In Figuur 5 is de applicatie te zien.



Figuur 5: Het programma RealTerm

5.1.6 Jd-gui

Naast het loggen van de communicatie tussen de Siemens Logosoft[2] software en de LOGO! is ook de Siemens Logosoft software gedecompileerd met Jd-gui. Hierdoor kan de code bekeken worden en kan daar de interface uit gehaald worden.

2 http://w3.siemens.com/mcms/programmable-logic-controller/en/logic-module-logo/logo-software/Pages/Default.aspx

5.2 Ontwerptools

In dit hoofdstuk worden de tools beschreven die gebruikt zijn bij het maken van de ontwerpen.

5.2.1 Logosoft

Het ontwerpen van de code voor de PLC is gebeurd in Logosoft. Dit is de grafische applicatie van Siemens waarmee de PLC geprogrammeerd kan worden.

5.2.2 Draw.io

Op draw.io zijn veel van de diagrammen gemaakt die in dit document zijn te vinden en die gebruikt zijn bij het ontwerpen van de applicaties.

5.3 Ontwikkeltools

In dit hoofdstuk worden de tools beschreven die gebruikt zijn om de code te ontwikkelen.

5.3.1 Logosoft

Logosoft is naast het ontwerpen ook tegelijk de applicatie waarmee de software voor de LOGO! ontwikkeld wordt. Deze software kan dan met een seriële verbinding in de LOGO! worden geladen.

5.3.2 Eclipse

In Eclipse is de C code voor de interface geprogrammeerd. Deze code wordt daarna via SSH naar de Raspberry PI geschreven en daar gecompileerd. De code werkt zowel op de Raspberry PI als het ontwikkelstation.

5.3.3 Sublime Text Editor

Sublime Text Editor is gebruikt om de server en webpagina's te ontwikkelen.

6 Resultaten

In dit hoofdstuk worden de resultaten van het project beschreven. Deze resultaten worden opgedeeld in drie onderdelen.

1. **Onderzoeksresultaten**

   Deze paragraaf beschrijft de resultaten van de onderzoeken die zijn uitgevoerd. Hieronder valt het onderzoek naar de interface met de Siemens LOGO! en het onderzoek naar het lineariseren van non-lineaire sensoren.

2. **Ontwerpresultaten**

   Deze paragraaf beschrijft het ontwerp van het totale systeem en de onderdelen van dit systeem. Hieronder vallen alle gemaakte modellen en diagrammen.

3. **Ontwikkelresultaten**

   Deze paragraaf beschrijft alle producten die zijn opgeleverd. Dit zijn alle softwareonderdelen en de testopstelling.

6.1 Onderzoeksresultaten

Het bedrijf kwam met het voorstel om voor het opslaan en opvragen van de data een Raspberry PI te gebruiken. Hiervoor moest wel duidelijk zijn of de Raspberry PI wel geschikt is voor deze toepassing. Daarom is hier onderzoek naar gedaan. Bij dit onderzoek is gekeken naar de interfacemogelijkheden van de Raspberry PI en van de Siemens LOGO!. Naast dit onderzoek werd het later tijdens het uitvoeren van de opdracht noodzakelijk om een ander onderzoek uit te voeren naar het lineariseren van non-lineaire sensoren.

Aan het begin van het project is een onderzoeksvraag opgesteld:

Is de Raspberry PI geschikt als data-recorder voor een Siemens LOGO! PLC?

Deze vraag is opgedeeld in de volgende deelvragen:

- Wat zijn de mogelijkheden van de interface vanuit de Raspberry PI?
- Wat zijn de mogelijkheden van de interface vanuit de Siemens LOGO!?
- Wat is het communicatieprotocol van de Siemens LOGO!?
- Wat voor informatie verstuurt de Siemens LOGO!?
- Wat is het data-format waarmee de Siemens LOGO! de data verstuurt?
- Welke mogelijke alternatieven zijn er voor de RRD(Round Robin Database)?

In de volgende paragrafen zullen deze vragen in het kort worden beantwoord. Deze vragen worden in Bijlage C1 – Vooronderzoek volledig behandeld.

Daarnaast is het tijdens het project nodig gebleken om nog een onderzoek uit te voeren naar het lineariseren van non-lineaire sensoren (Bijlage C2 – Onderzoek non-lineaire sensoren). Dit zal ook kort worden behandeld.

6.1.1 Wat zijn de mogelijkheden van de interface vanuit de Raspberry PI?

Een belangrijk aspect dat al snel zou bepalen of de Raspberry PI als datacollector geschikt is, zijn de interfacemogelijkheden. Zonder communicatie is het onmogelijk om de gegevens te verzamelen en te publiceren.

De Raspberry PI heeft verschillende mogelijkheden om met andere apparaten te communiceren. Om met hardware te communiceren zijn er de volgende interfaces:

- 26 GPIO pinnen

  Dit zijn 26 input/output pinnen op de Raspberry PI. Tussen deze pinnen zitten een aantal pinnen die de volgende interfaces kunnen leveren: UART, i2c bus en SPI bus.

- i2s audio

  Dit is een andere aansluiting waarmee de Raspberry PI zou kunnen communiceren.

Daarnaast heeft de PI één of twee usb poort(en)(afhankelijk van de uitvoering). Via deze usb poort kan gecommuniceerd worden met elk ander apparaat dat via een usb-kabel is te verbinden, en ondersteund wordt door Linux Raspbian.

Bij de Raspberry PI model B zit er ook een Ethernet/LAN-aansluiting op. Hierdoor is het mogelijk om de PI met het internet te laten verbinden.[3]

6.1.2 Wat zijn de mogelijkheden van de interface vanuit de Siemens LOGO!?

De Siemens LOGO! 0BA6(Gebruikte versie bij de opdracht) heeft als enige communicatiemogelijkheid zijn seriële poort. Daarnaast zijn er hardware uitbreidingsmodules waarmee de communicatiemogelijkheden vergroot worden.

Voor de communicatie zijn twee uitbreidingsmodules beschikbaar[4]:

- CM AS-Interface-Slave

  Deze uitbreidingsmodule is een interface tussen de Siemens LOGO! en een AS-Interface systeem. Hiermee wordt het dus mogelijk gemaakt om via een AS-Interface te communiceren.

- CM KNX/EIB

  Deze uitbreidingsmodule zorgt ervoor dat de Siemens LOGO! kan communiceren met systemen die gebruik maken van EIB. Dit gaat doormiddel van het uitwisselen van KNX berichten.

Naast het aanschaffen van een uitbreidingsmodule is er ook de mogelijkheid om in plaats van de Siemens LOGO! 0BA6 de Siemens LOGO! 0BA7 te gebruiken. Hierbij komt de extra mogelijkheid vrij om via de ethernetkabel te communiceren.

6.1.3 Wat is het communicatieprotocol van de Siemens LOGO!?

Het communicatieprotocol van de Siemens LOGO! is zeer uitgebreid. Het is daarom niet nuttig om al deze informatie hier te vermelden. Er zal hier kort worden ingegaan op de commando's die aan de Siemens LOGO! gegeven kunnen worden. Voor meer informatie wordt verwezen naar Bijlage C1 – Vooronderzoek. Uit het onderzoek zijn de volgende commando's naar voren gekomen:

1. Start/Stop

   Met dit commando kan de LOGO! gestart en gestopt worden.

2. Get status

   Hiermee kan de status van de LOGO! worden opgevraagd.

3 http://www.raspberrypi.org/help/faqs/
4 http://w3.siemens.com/mcms/programmable-logic-controller/en/logic-module-logo/modular-communication-variants/pages/default.aspx

3. Get data

   Hiermee kan alle data die de LOGO! heeft, worden opgevraagd.

4. Get firmware

   Hiermee kan de versie van de firmware worden opgevraagd.

5. Set tijd/Get tijd

   Hiermee kan de tijd worden ingesteld of opgevraagd.

6. Set byte/byte-array

   Hiermee kan een byte of byte-array in de LOGO! worden opgeslagen. Dit is een onveilige methode die niet door Siemens wordt ondersteund.

7. Get byte/byte-array

   Hiermee kan een byte of byte-array uit de LOGO! worden gelezen. Dit is een onveilige methode die niet door Siemens wordt ondersteund.

6.1.4 Wat voor informatie verstuurt de Siemens LOGO!?

Het is belangrijk om te weten welke gegevens de Siemens LOGO! verstuurt, daarbij gaat het vooral om de data die de LOGO! verstuurt bij het ontvangen van het get data commando.

Data Reply															
Start	Cmd	Cmd	Length		Checksum		Block length	Values length	actual values Length	B1-B8	B9-B16	B17-B24	B25-B32	B33-B40	B41-B48
B49-B56	B57-B64	B65-B72	B73-B80	B81-B88	B89-B96	B97-B104	B105-B112	B113-B120	B121-B128	B129-B136	B137-B144	B145-B152	B153-B160	B161-B168	B169-B176
B177-B184	B185-B192	B193-B200	I1-I8	I9-I16	I17-I24	TD	Q1-Q8	Q9-Q16	M1-M8	M9-M16	M17-M24	M25-M27	S1-S8	C1-C4	AI1
AI1	AI2	AI2	AI3	AI3	AI4	AI4	AI5	AI5	AI6	AI6	AI7	AI7	AI8	AI8	AQ1
AQ1	AQ2	AQ2	AM1	AM1	AM2	AM2	AM3	AM3	AM4	AM4	AM5	AM5	AM6	AM6	Stop

Uit het bovenstaande dataframe dat verkregen wordt door het uitvoeren van het get data commando kan de volgende informatie worden gehaald:

- Outputs van alle digitale blokken (B1-B200, blauw in bovenstaand dataframe)

  Een PLC programma wordt opgebouwd uit blokken, dit zijn alle digitale waarden van de outputs van deze blokken.

- 24 Digitale inputs (I1-I24, lichtgroen in bovenstaand dataframe)
- Inputs TD module (TD, lichtgroen in bovenstaand dataframe)
- 16 Digitale outputs(Q1-Q16, lichtgroen in bovenstaand dataframe)
- 27 Digitale flags(M1-M27, groen in bovenstaand dataframe)
- 8 Shift register bits(S1-S8, donkergroen in bovenstaand dataframe)
- 4 Toets inputs(C1-C4, donkergroen in bovenstaand dataframe)
- 8 Analoge inputs(AI1-AI8, lichtrood in bovenstaand dataframe)
- 2 Analoge outputs(AQ1-AQ2, lichtrood in bovenstaand dataframe)
- 6 Analoge flags(AM1-AM6, rood in bovenstaand dataframe)

Daarnaast kan de LOGO! 0BA6 ook nog de volgende data versturen:

1. LOGO! status → programma aan/uit
2. Hardware ID
3. Firmware versie
4. Tijd

6.1.5 Wat is het data-format waarmee de Siemens LOGO! zijn data verstuurt?

Deze informatie is tijdens het uitvoeren van de onderzoeksvraag in hoofdstuk 6.1.3 naar voren gekomen. Uit dit onderzoek is gebleken dat de Siemens LOGO! Gebruik maakt van drie soorten data typen:

1. bit

   De bit wordt gebruikt voor alle digitale waarden.

2. 2 bytes

   Voor alle analoge waarden wordt gebruik gemaakt van 2 bytes. Hierbij wordt de LSB-byte als eerste verstuurd.

3. 4 bytes

   Voor alle tijdsaanduidingen wordt gebruik gemaakt van 4 bytes. Hierbij wordt de LSB-byte als eerste verstuurd.

6.1.6 Welke mogelijke alternatieven zijn er voor de RRD(Round Robin Database)?

Voordat naar alternatieven gezocht kan worden moet eerst duidelijk worden gemaakt wat de RRD-tool is. Daarna kan op basis van de specificaties van de RRD-tool een alternatief gezocht worden.

6.1.6.1 RRD-tool

De RRD-tool heeft als doel om de data van een bepaalde tijdsspanne op te slaan. Deze data wordt opgeslagen in een round robin database, die niet van grootte verandert. Dit komt doordat na een bepaalde tijd de oudste entry wordt overschreven door de nieuwste entry.

Specificaties/eigenschappen RRD-tool:

- RRD-tool is een TSD(Time series database), dit betekent dat de RRD-tool is geoptimaliseerd om data voor een bepaalde tijdsspanne op te slaan.
- De RRD-tool levert extra tools waarmee het mogelijk is om de RRD-data om te zetten naar grafieken.
- De RRD-tool ondersteunt meerdere programmeertalen: Perl, Python, Ruby, Tcl, PHP, Lua, Java en C.

De RRD-tool heeft een vast datagebruik, dat wil zeggen dat de grootte van de database niet verandert. Dit komt doordat de RRD-tool aan het eind van zijn database weer verder gaat door het overschrijven van de eerste waardes.

6.1.6.2 Alternatieven voor de RRD-tool

Op basis van de specificaties die in het vorige hoofdstuk zijn aangegeven kunnen een aantal alternatieven gezocht worden voor de RRD-tool.

Database	**TSD-optimalisatie**	**Grafiek ondesteuning**	**Vast datagebruik**	**Gratis**
Geras	X	X	–	–
InfluxDB	X	X	–	X
KairosDB	X	–	–	X
KDB+	X	X	?	–
OpenTSDB	X	X	–	X
TempoDB	X	X	–	–
RRD-tool	X	X	X	X

Deze lijst van alternatieven is opgesteld door te zoeken naar databases die TSD-optimalisatie(Time series database optimalisatie) hebben, aangezien dit een key functie is die er toe heeft geleid om de RRD-tool te kiezen.

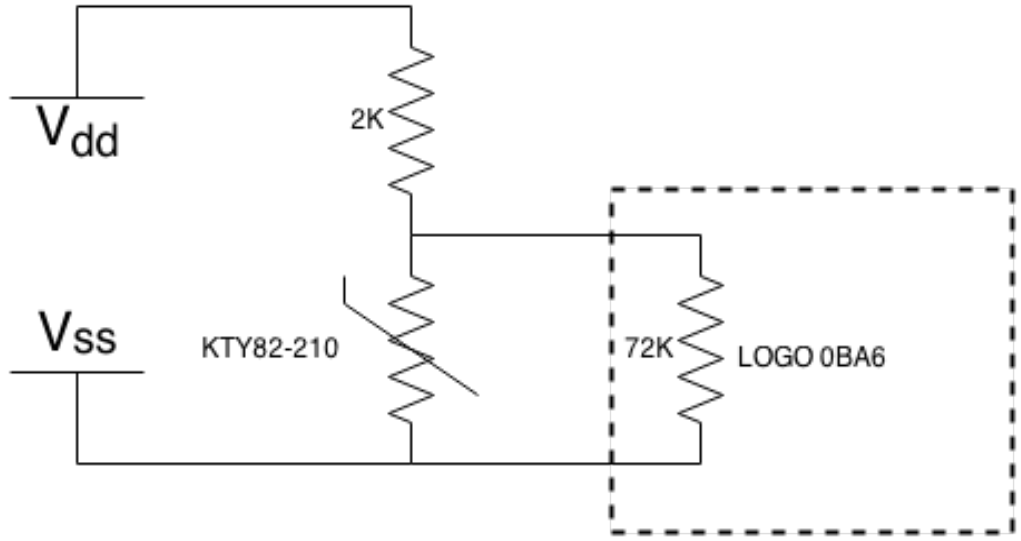
Aan de hand van deze lijst is ook te zien dat eigenlijk geen van alle databases uit zichzelf een vast datagebruik ondersteunt. Dit is geen zeer belangrijk punt aangezien dat gemakkelijk toe te passen is. Grafiekondersteuning is daarentegen een belangrijk punt bij de keuze voor een alternatief, aangezien alle data ook getoond moet worden. Daarnaast moet de database gratis te gebruiken zijn, omdat er geen budget is om een betaalde database te gebruiken. Uit deze lijst blijkt dat OpenTSDB en InfluxDB de beste alternatieven zijn.

6.1.7 Conclusie onderzoek Raspberry PI

Uit dit onderzoek is gebleken dat de Raspberry PI veel verschillende communicatiemogelijkheden heeft, waar tegenover staat dat de Siemens LOGO! 0BA6 maar één communicatiemogelijkheid heeft: de seriële poort. De Raspberry PI kan ook communiceren met deze seriële poort via zijn usbpoort. Het protocol dat daarbij gebruikt wordt is dankzij dit onderzoek bekend geworden. Daardoor is het mogelijk geworden voor de Raspberry PI om met de Siemens LOGO! te communiceren. De informatie die daarbij uitgewisseld wordt is voldoende om een proces te monitoren en in een database op te slaan. Uit het onderzoek naar alternatieven voor de database is gebleken dat er bijna geen goed bekende databases zijn die de RRD-tool zouden kunnen vervangen. De databases die wel als vervanging in aanmerking komen, hebben een groter geheugengebruik dan de RRD-tool wat dan ook het negatieve punt is voor de vervangende databases.

6.1.8 Resultaten onderzoek non-lineaire sensoren

Tijdens het project is besloten om over te schakelen van een lineaire naar een non-lineaire sensor, omdat de lineaire sensor niet de hoogste temperatuur van de zonnecollector kan tonen. Daarom moest worden onderzocht hoe deze sensor gelineariseerd kan worden. Deze linearisatie is nodig omdat de PLC niet goed met non-lineaire sensors om kan gaan. Uit dit onderzoek is naar voren gekomen dat de te gebruiken non-lineaire sensor bijna lineair is met een kleine curve. Uit de documentatie, die de producent heeft geleverd, bleek dat door toepassing van een shuntweerstand en een serieweerstand deze non-lineaire sensor gelineariseerd kan worden voor een bepaald bereik. De schakeling van deze weerstanden en temperatuur sensor is te zien in onderstaand figuur.



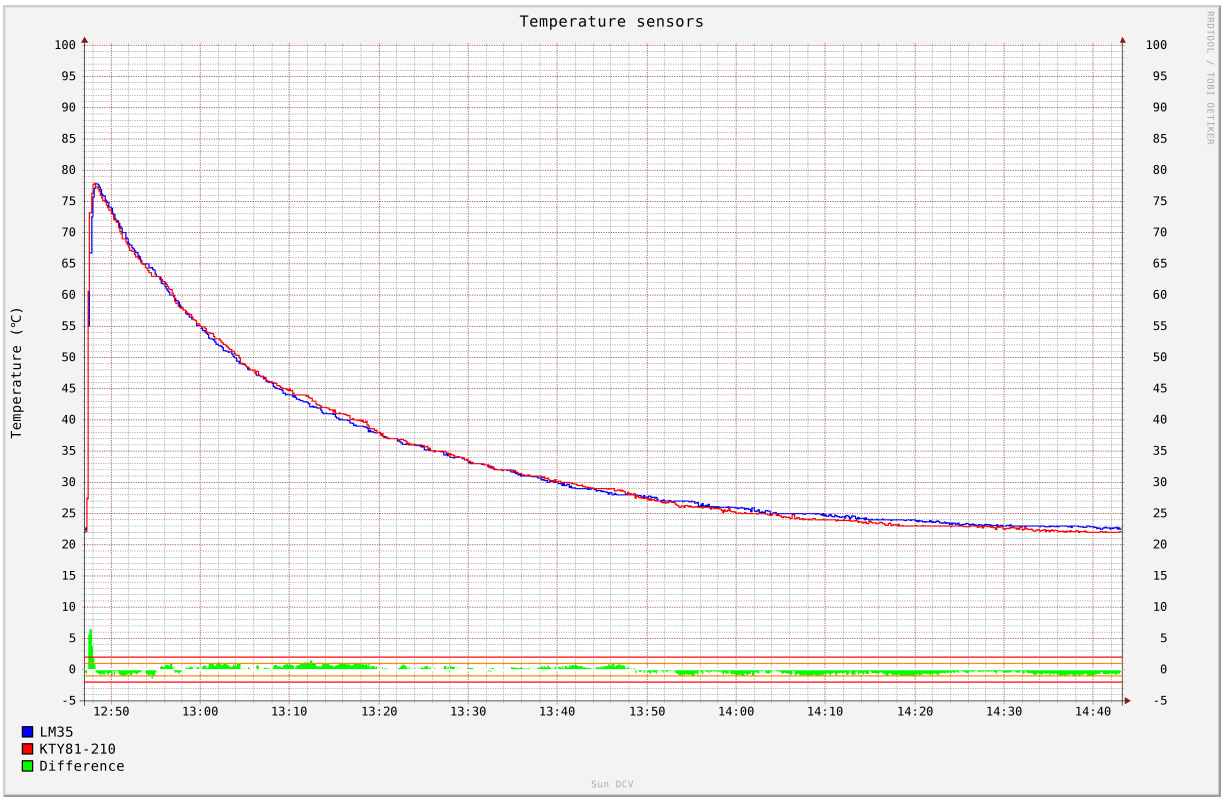
Figuur 6: De non-lineaire sensor met de serieweerstand en de shuntweerstand aangesloten.

Daarnaast is gebleken dat de nauwkeurigheid van de linearisatie afhankelijk is van twee factoren: het aantal mV/℃ en de stroomsterkte door de sensor. Deze beide waarden worden beïnvloed door de shuntweerstand.

De nauwkeurigheid wordt beïnvloed door het aantal mV/℃ omdat de Siemens LOGO! een meetnauwkeurigheid heeft van 10mV/℃. Dit heeft tot gevolg dat bij een lager aantal mV/℃ niet alle graden in de temperatuur range getoond kunnen worden.

De nauwkeurigheid wordt beïnvloed door de stroomsterkte omdat de temperatuursensor bij een hogere stroom dan 1 mA zichzelf gaat opwarmen, wat invloed heeft op de gemeten temperatuur.

Met de uiteindelijk gekozen weerstand is een nauwkeurigheid van ongeveer 1 ℃ behaald. Deze nauwkeurigheid kan niet hoger worden omdat de minimale fout altijd één meetunit is. Het resultaat is te zien in Grafiek 1.



Grafiek 1: Afkoelcurve in water met twee sensoren.
Blauw: referentie sensor (ligt tegen rood aan)
Rood: non-lineaire sensor (onder test)
Groen: verschil non-lineaire sensor en referentie sensor

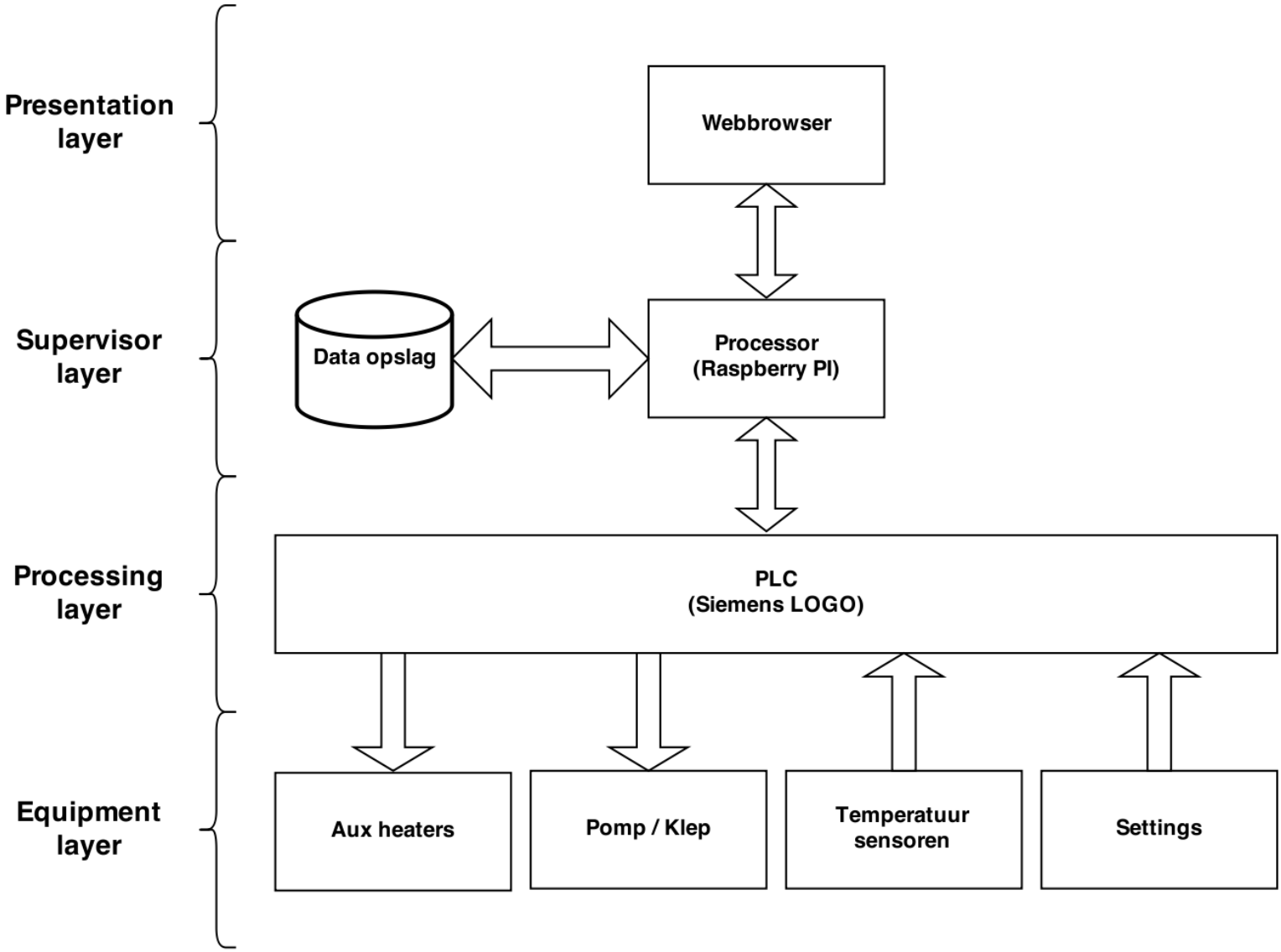
Om tot deze resultaten te komen is een rekenmodel opgesteld. In dit rekenmodel kunnen de gemeten waarden van de non-lineaire sensor worden ingevuld. Daarna kunnen deze waarden door het tunen van één van de ingestelde waarden dichter bij de referentietemperatuur worden gebracht. Voor extra informatie over het rekenmodel wordt verwezen naar Bijlage C2 – Onderzoek non-lineaire sensoren.

6.2 Ontwerpresultaten

Tijdens het project zijn ontwerpen gemaakt van het systeem en de onderliggende softwareonderdelen. Deze ontwerpen zijn vooraf aan het ontwikkelen van elk softwareonderdeel tot stand gekomen.

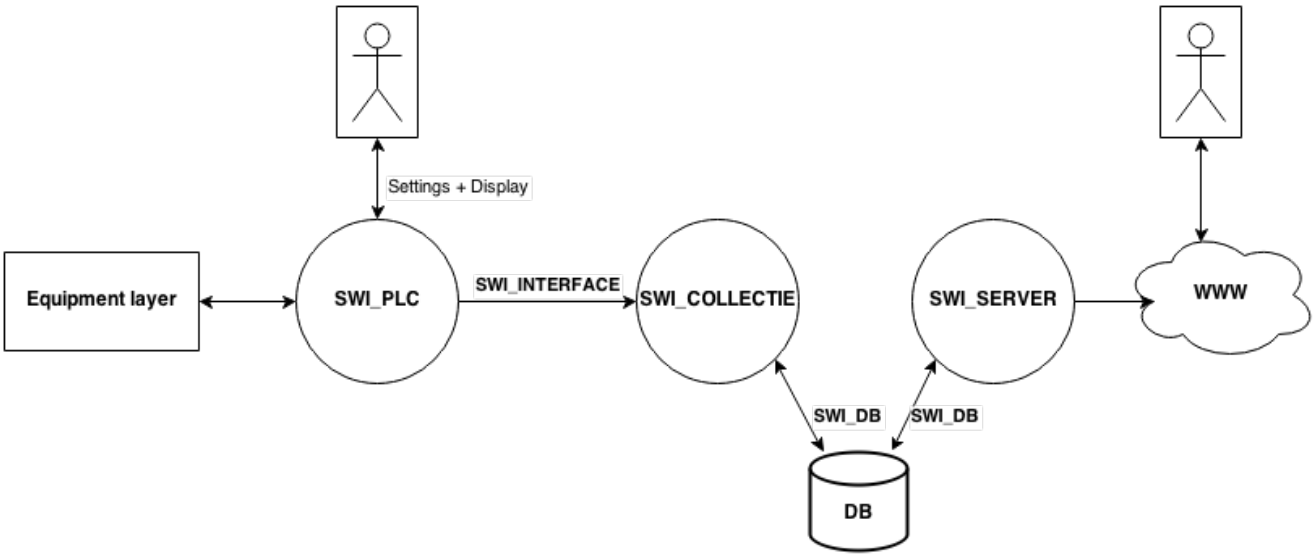
6.2.1 Systeem ontwerp

In Figuur 7 is het lagenmodel van het systeem te zien. Hierin is de opbouw van het totale systeem duidelijk te zien. Er is voor een lagenmodel gekozen omdat dit gedeeltelijk door de hardware wordt afgedwongen. Elke laag uit het lagenmodel bestaat uit een laag in de hardware. De Equipment layer beschrijft de hardware die aan de PLC is gekoppeld, zoals sensoren en actuatoren. De Processing layer is de PLC zelf, deze communiceert met de equipment layer en de Supervisor layer. In de Supervisor layer zitten twee hardware onderdelen: de Raspberry PI en de Data opslag. Voor de Presentation layer wordt gebruik gemaakt van een webbrowser. De pijlen geven de communicatie binnen het systeem weer.



Figuur 7: Lagenmodel van het systeem

In Figuur 8 is te zien hoe de communicatie tussen de verschillende softwareitems loopt en hoe de gebruikers met de softwareitems kunnen communiceren.



Figuur 8: Interactie tussen Softwareitems(SWI)

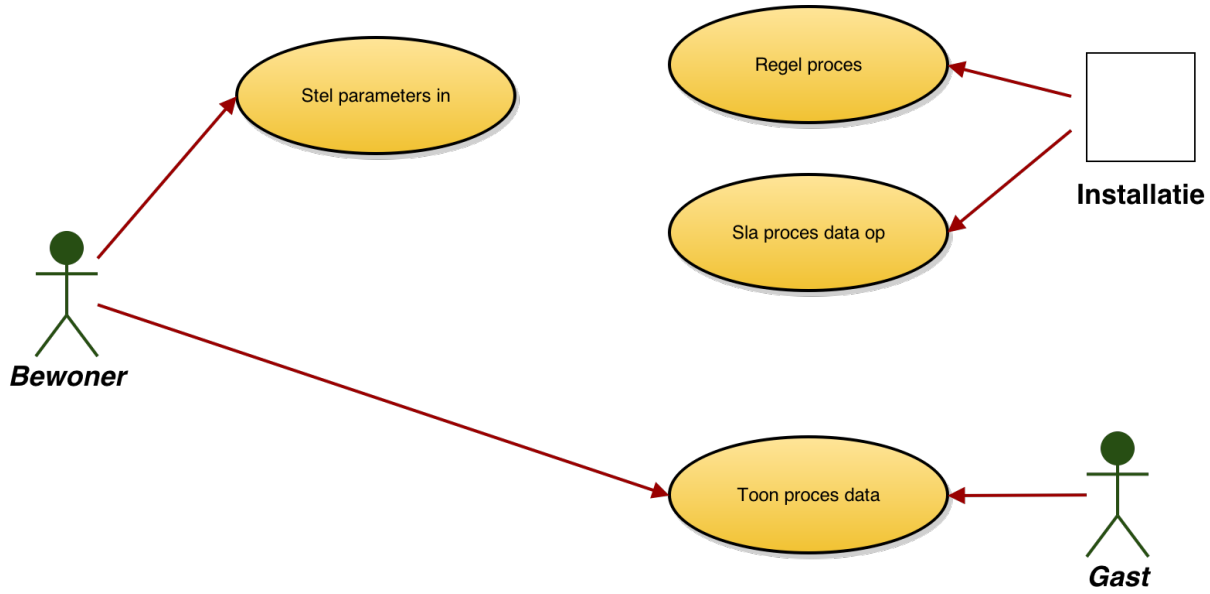
Toelichting:

1. SWI_PLC: De software die de installatie regelt.
2. SWI_INTERFACE: De interface tussen de Raspberry PI en de PLC.
3. SWI_COLLECTIE: De software die de data verzamelt en in de database stopt.
4. SWI_DB: De interface met de database.
5. SWI_SERVER: De server en alle bijbehorende componenten.

De eigenaar en bewoners van de woningen kunnen op de locatie zelf de instellingen wijzigen en de huidige gegevens bekijken op een display. Dit wordt mogelijk gemaakt door de SWI_PLC.

Daarnaast kunnen de gegevens van het proces worden bekeken via de website. Deze website zal communiceren met de server en de laatste gegevens opvragen. Omdat de SWI_SERVER de gegevens van de database niet kan aanpassen is de website vrij toegankelijk gemaakt voor iedereen die de gegevens wil bekijken.

6.2.2 Use cases



Figuur 9: Use case diagram totale systeem

Bij het vaststellen van de eisen voor de software is ook een use case diagram gemaakt. In dit use case diagram zijn de acties getoond die een actor kan uitvoeren. Een actor kan een persoon, maar ook een systeem zijn. In dit use case diagram zijn drie actoren: de bewoner, het te regelen systeem en een gebruiker van de website. De bewoner kan naast bewoner ook gebruiker van de website zijn.In dit use case diagram worden de acties beschreven die de bewoner en website bezoekers kunnen uitvoeren.

Door de actie “Stel parameters in” kan de bewoner de gewenste temperatuur verhogen of verlagen, ook kan de bewoner aangeven of een huis wel of niet verwarmd moet worden. De parameters die hier uit komen worden gebruikt bij het regelen van het proces. De installatie zal het proces regelen, en de data van dit proces zal opgeslagen worden in de database.

Door de actie “Toon proces data” kan de bewoner op een lokaal display de huidige gegevens van het proces bekijken. De huidige gegevens van het proces bestaan ondere andere uit de temperaturen van de boilers en de zonnecollector. De bewoner kan het proces bekijken via de website. Daarnaast is deze website vrij toegankelijk en kan elke gast de website benaderen.

6.2.3 Ontwerp proces

Voor het ontwerpen van het proces is gebruik gemaakt van de geleverde ontwikkelomgeving van Siemens. In deze software is het mogelijk om met functieblokken het proces te ontwerpen en via simulatie te testen. Het ontwerp wordt vervolgens gecompileerd naar werkende code en geladen in de PLC. In deze software is het mogelijk om op twee manieren te ontwerpen: met functieblokken of ladderdiagrammen.

6.2.4 Ontwerp interface

De interface die gebruikt wordt om met de PLC te communiceren is gebaseerd op de communicatie tussen de software van Siemens en de PLC. Door deze communicatie te analyseren zijn de commando's die gebruikt worden om de PLC aan te sturen bekend geworden. De commando's die voor deze opdracht van toepassing zijn, zijn in het interfaceproces verwerkt. De interface die hieruit is ontstaan levert de volgende services:

1. Open/Close connection

Met deze service kan de verbinding met de PLC worden geopend en gesloten.

2. Start/Stop programma

   Met deze service kan het programma op de PLC gestart en gestopt worden.

3. Get status

   Met deze service kan de status van het programma opgevraagd worden: Running/Stopped.

4. Set/Get tijd

   Met deze service kan de tijd van de PLC ingesteld en opgevraagd worden.

5. Get datapackage

   Met deze service kan alle huidige informatie van het programma opgevraagd worden.

6.2.5 Ontwerp database

In tegenstelling tot de object georiënteerde databases, is de Round Robin Database een time series database[5]. Dit houdt in dat de database geoptimaliseerd is om gegevens bij een bepaald tijdstip op te slaan. Het ontwerp van de database ziet er dan ook iets anders uit dan bij andere databases. De RRD wordt opgebouwd uit twee onderdelen: DS en RRA.

Een DS is een variabele die opgeslagen moet worden. Hierbij kan gedacht worden aan bijvoorbeeld de temperatuur van een boiler. Een RRA is een archief waarin deze waarden worden opgeslagen. De structuur van de database is te zien in de tabellen 1 & 2.

DS			
Naam:	**Type**	**Max. tijdafwijking:**	**Waarde:**
Temperatuur collector	Input	120 seconden	0-200 °C
Temperatuur boiler A	Input	120 seconden	0-200 °C
Temperatuur boiler B	Input	120 seconden	0-200 °C
Temperatuur buiten	Input	120 seconden	-20-100 °C
Bewoning huis A	Input	120 seconden	0-1
Bewoning huis B	Input	120 seconden	0-1
Electrische Verwarming huis A	Output	120 seconden	0-1
Electrische Verwarming huis B	Output	120 seconden	0-1
Driewegklep	Output	120 seconden	0-1
Pomp	Output	120 seconden	0-1
Ingestelde drempelwaarde huis A	Input	120 seconden	0-200 °C
Ingestelde drempelwaarde huis B	Input	120 seconden	0-200 °C
Waterflow rate	Input	120 seconden	0-10 l/min

Tabel 1: De DS van de database.

RRA			
Type:	**Max. aantal onbekend:**	**Stap grootte:**	**Aantal:**
Gemiddelde	50.00%	60 s	1051898
Gemiddelde	70.00%	1 h	17532
Gemiddelde	70.00%	1 d	731

Tabel 2: De RRA van de database.

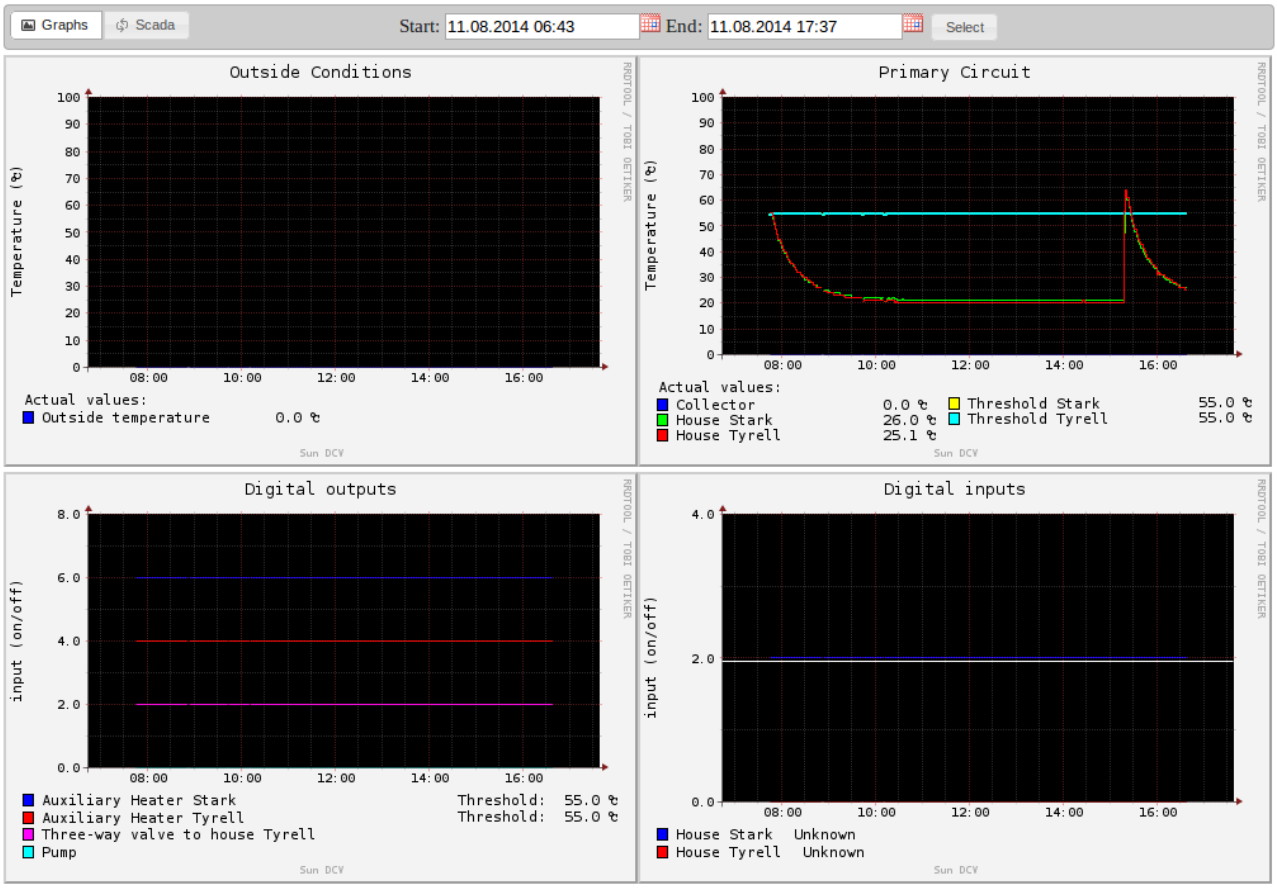
De DS(meerv.) van de database worden elke minuut geüpdatet. Als er 120 seconden geen update is geweest zal de waarde op onbekend worden gezet. Voor elke waarde wordt aangegeven wat het minimum en maximum van die waarde mag zijn. Waarden buiten dit minimum en maximum worden ook op onbekend gezet.

Deze DS(meerv.) worden opgeslagen in drie verschillende RRA's. Dit is gekozen omdat bij het genereren van een grafiek alle waarden in het geselecteerde bereik uit de database worden gehaald. Bij een groot geselecteerd bereik kost dit veel tijd. Dit probleem wordt verholpen doordat een andere RRA met een kleinere dichtheid wordt gekozen bij een groter geselecteerd bereik. Elke RRA baseert zijn waardes op de RRA met een kleinere stapgrootte dan zichzelf. De RRA zal zijn waarde op onbekend zetten als het maximaal onbekende aantal waardes te groot wordt. Elke RRA slaat voor

5 Oetiker, T. (30-09-2014). *rrdtool*. Geraadpleegd op http://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html

twee jaar aan waardes op.

6.2.6 Ontwerp server/site



Figuur 10: Hoofdpagina van de site.

Voor het ontwerp van de site is uitgegaan van een aangeleverd concept door het bedrijf. Dit heeft geleid tot de hoofdpagina zoals te zien is in Figuur 10. Hierin kan met twee invoervelden de inhoud van de grafieken worden aangepast. Deze aanpassing is ook mogelijk gemaakt door een overlay over de grafiek waarmee een bepaald gebied geselecteerd kan worden waarop wordt ingezoomd. Via de bovenste knop kan van pagina gewisseld worden. Op de hoofdpagina van de site zijn vier grafieken zichtbaar waarin de status van de installatie is te zien, hieronder worden de grafieken beschreven.

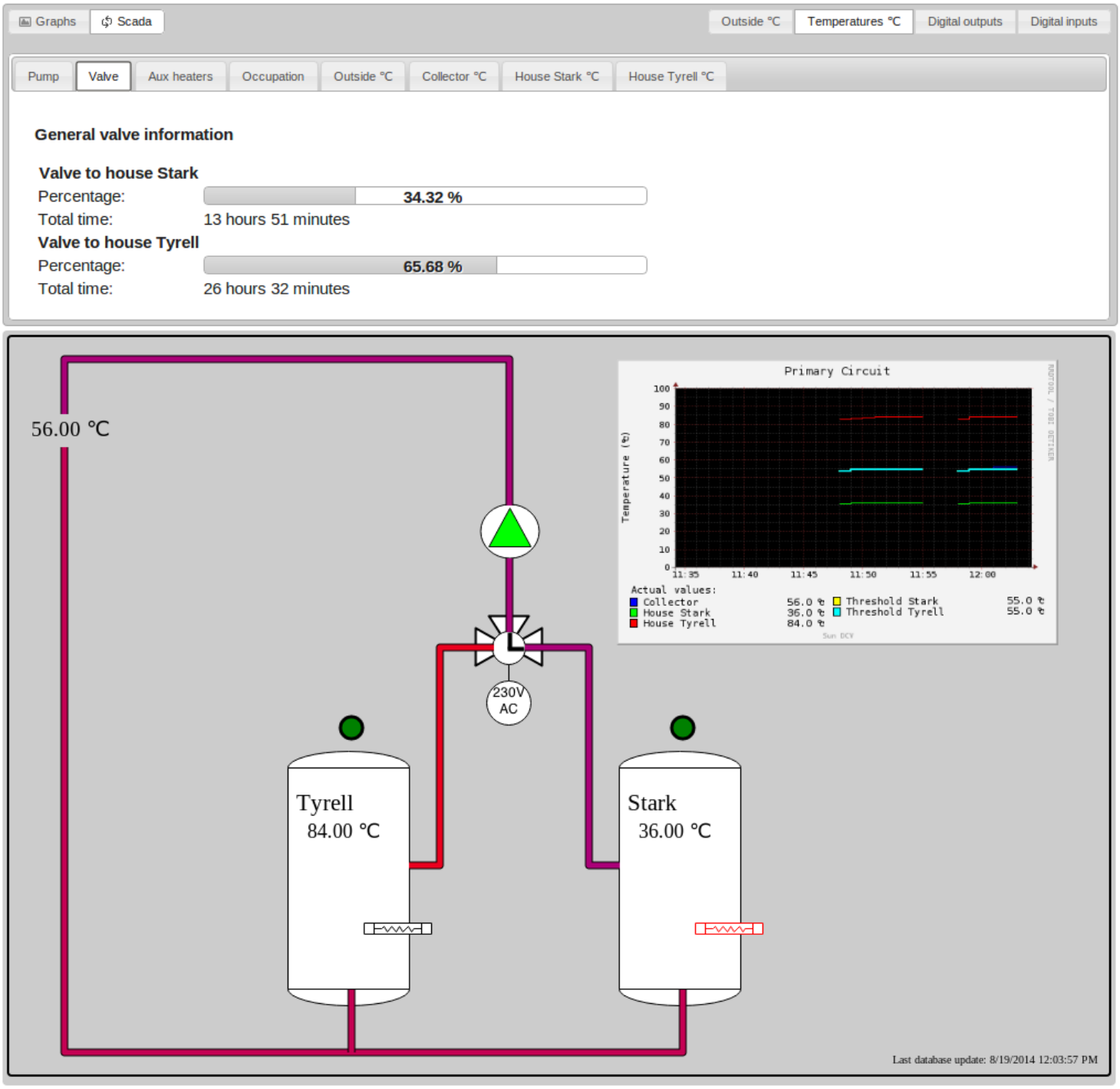
1. In de eerste grafiek wordt de buitentemperatuur getoond. In de situatie dat er geen buitentemperatuur sensor is aangesloten blijft deze grafiek leeg.

2. In de tweede grafiek zijn de temperaturen van de collector en de boilers te zien. Daarnaast zijn de gewenste temperaturen(thresholds) aangegeven.

3. In de derde grafiek staat de status van de digitale outputs, zoals de heaters/pomp en klep. Aan de hand van deze grafiek kan bekeken worden of de heaters ook gebruikt worden.

4. In de laatste grafiek is aangegeven of de huizen bewoond of onbewoond zijn. Om te voorkomen dat beide lijnen over elkaar lopen krijgen bepaalde lijnen een offset, waardoor hun nullijn hoger is(2.0).

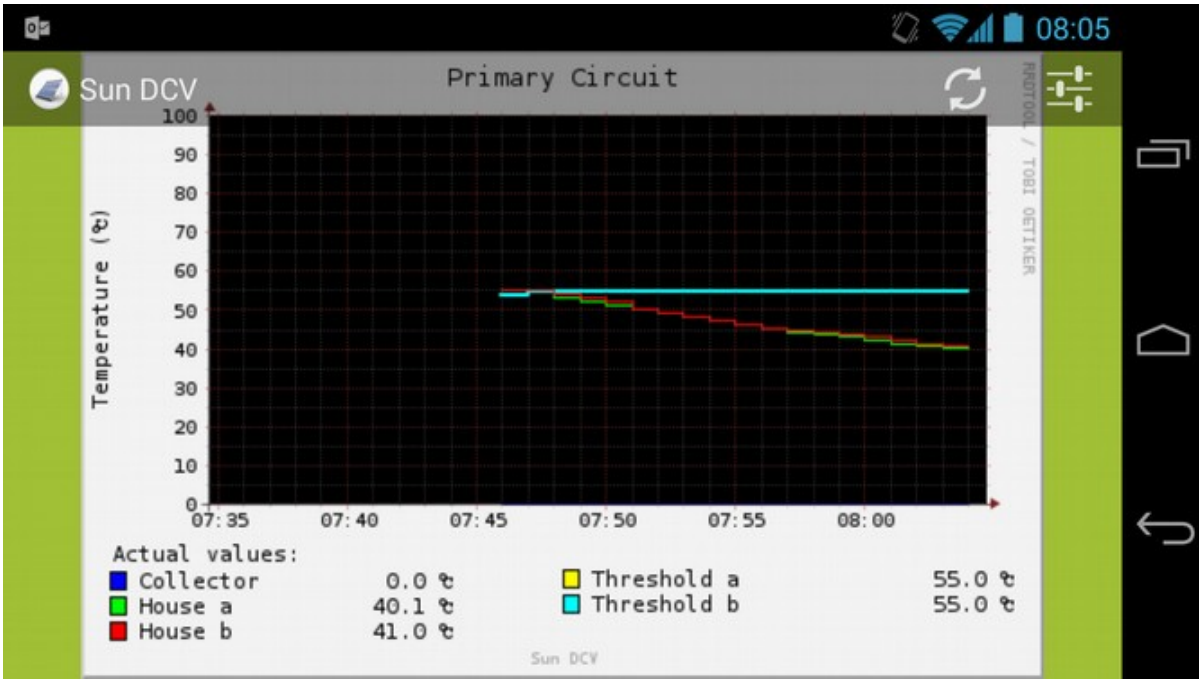
Op de Scada pagina zijn de huidige waarden van het proces grafisch weergegeven. Naast deze gegevens is via tabbladen de algemene informatie van het systeem beschikbaar. Hier staat van elke temperatuursensor de minimum, gemiddelde en maximum temperatuur. Van alle digitale ingangen en uitgangen staat hoelang ze aan of uit zijn geweest. Via vier knoppen is het mogelijk om het type van de grafiek te wijzigen. De Scada pagina is te zien in Figuur 11.



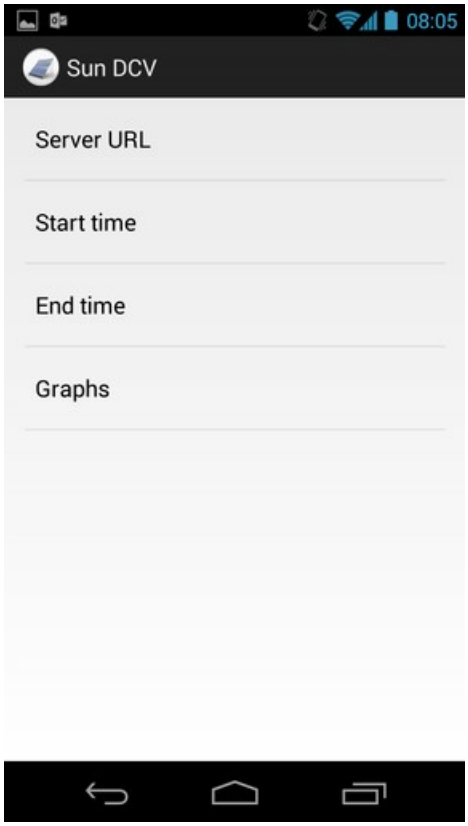
Figuur 11: Scada pagina van de site, met de huidige gegevens.

6.2.7 Ontwerp mobiele applicatie

Naast de website is er ook voor gekozen om een mobiele applicatie voor Android toestellen te maken. In deze applicatie zijn de grafieken van het systeem te bekijken. Zie Figuur 12.



Figuur 12: Ontwerp van de mobiele applicatie.



Naast het bekijken van de grafieken is er een aantal instellingen die via het instellingenmenu kunnen worden ingesteld. Hiermee kan onder andere de URL van de server worden aangepast, de starttijd en eindtijd van de grafieken worden ingesteld en kan gekozen worden welke grafieken getoond worden.

Figuur 13: Settings screen.

6.3 Ontwikkelresultaten

In dit hoofdstuk worden de producten beschreven die voor dit project zijn gemaakt.

6.3.1 Processoftware

De processoftware bestaat uit een blokkendiagram dat gemaakt is in de geleverde software van Siemens. Met dit blokkendiagram wordt het proces van de PLC beschreven. Dit is te zien in Figuur 14. In dit figuur wordt de input van de gebruiker op de PLC afgehandeld.

In dit diagram zijn vier knoppen aanwezig(C1, C2, C3 en C4), één pulse timer(T008), drie optelblokken(C006, C022 en C024), twee mathematische blokken(SF019 en SF023) en één outputblok(Q5).

De PLC gaat door dit programma van links naar rechts, waarbij de waarde van elk blok wordt gecontroleerd of berekend. Hierbij gaat het vooral om de volgende type blokken:

Contact blok

 Het contact blok zal als de gegeven waarde hoog is de uitgang ook hoog maken.

NOT-contact blok

 Het not-contact blok zal als de gegeven waarde hoog is de uitgang laag maken.

Coil blok

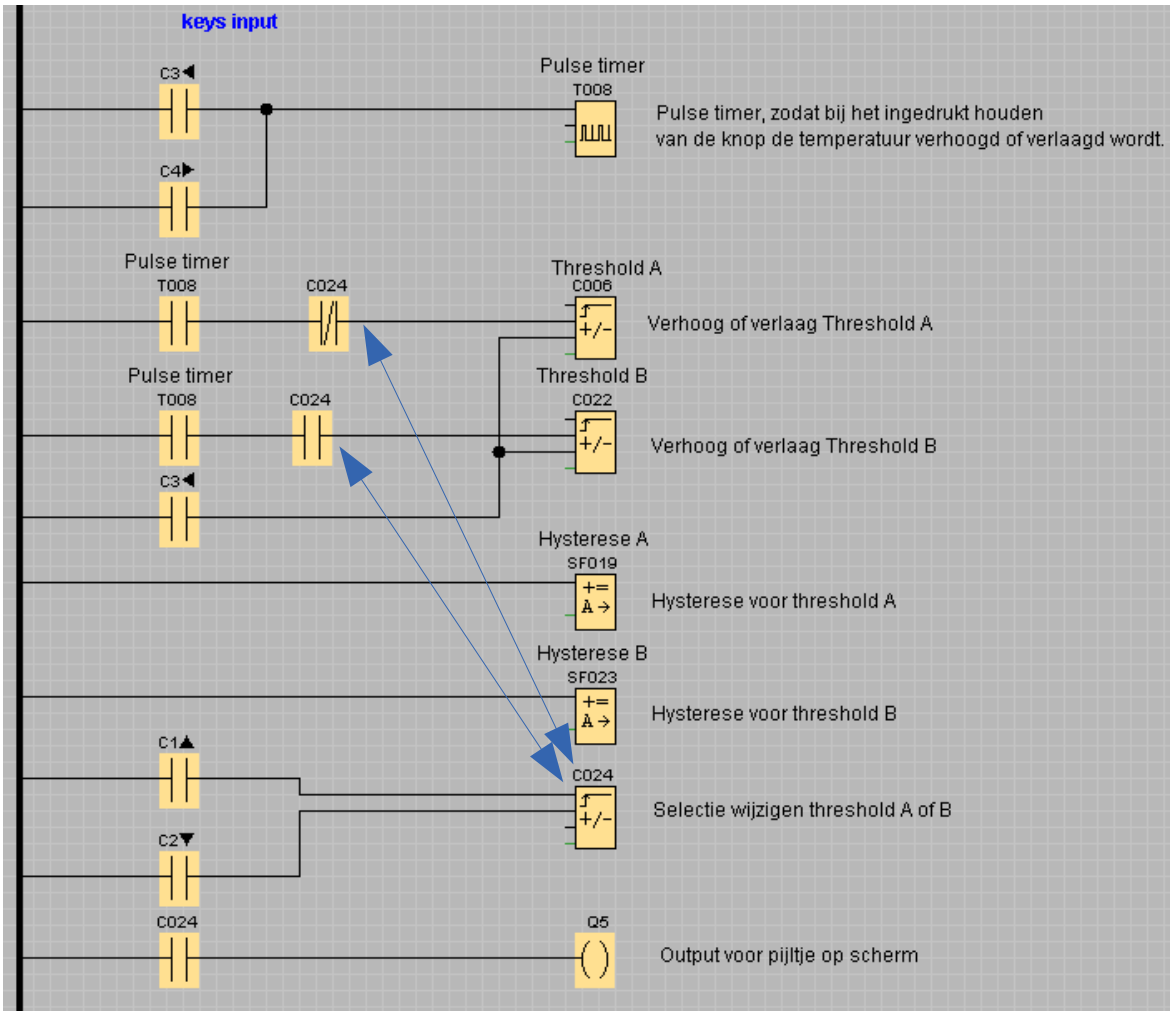
 Het coil blok is een output of flag van de PLC.

Als deze blokken onder elkaar worden gezet zullen ze parallel gecontroleerd worden, anders worden ze serieel gecontroleerd. Dit is gelijk aan een OR-poort(parallel) of een AND-poort(serieel).

In het eerste deel van het blokschema zal de pulse timer(T008) gaan lopen als knop drie(C3) of knop vier(C4) wordt ingedrukt. In het daaronder staande gedeelte wordt de waarde van deze pulse timer gecontroleerd met een contact blok. Aan de hand van deze waarde en de waarde van de selectie berekening(C024) zal één van de Thresholds(C006 en C022) verhoogd of verlaagd(Als C3 wordt ingedrukt) worden. De selectie berekening wordt iets lager in het programma uitgevoerd(Zie blauwe pijlen). Deze selectie wordt berekend aan de hand van de toetsen C1 en C2(C1 maakt de output laag en C2 maakt de output hoog).

De hysterese op de thresholds(SF019 en SF023) moet altijd toegepast worden daarom zitten deze blokken direct op de hoofdbaan aangesloten.

De laatste regel in het diagram toont het omzetten van de selectie waarde(C024) naar een output(Q5). Dit wordt gedaan omdat het scherm van de PLC de rekenblokken niet kan tonen maar wel de outputs.



Figuur 14: Afhandelen van keys input

6.3.2 Interface communicatie software

De interface communicatie software levert de communicatie met de PLC en de database op. Deze software vraagt de laatste gegevens van de PLC en slaat deze met een bepaald tijdsinterval op in de database. Daarnaast verzorgt deze interface alle andere communicatie met de PLC. Hieronder vallen alle commando's die in de ontwerpfase zijn vastgesteld. De tijd tussen de PLC en de processing unit moet synchroon lopen, deze wordt daarom bij het opstarten van de processing unit gesynchroniseerd. Deze synchronisatie wordt daarnaast periodiek uitgevoerd omdat de PLC geen nauwkeurige klok heeft.

6.3.3 Site

De site bestaat uit twee hoofdonderdelen: de startpagina en de scadapagina.

6.3.3.1 Startpagina

Op de startpagina staan vier grafieken met daarin standaard de gegevens van de laatste vier uur. Deze periode kan met twee invoervelden worden aangepast. Deze invoervelden kunnen worden ingevuld op drie manieren:

1. Handmatig

   Beide invoervelden hebben de mogelijkheid om handmatig ingevuld te worden. Als de velden leeg worden gelaten zullen ze de standaardwaarde van het veld aannemen.

2. Kalendericoon naast het invoerveld.

   Door op het kalendericoon te klikken zal een kalender verschijnen waarop een datum geselecteerd kan worden. Deze datum wordt ingevoerd in het tijdsveld.

3. Overlay over elke grafiek.

   Elke grafiek heeft een overlay waarop met de muis geklikt kan worden. Door met de muis over de grafiek te slepen kan een bepaald gedeelte van de grafiek geselecteerd worden zodat daarop kan worden ingezoomd. Met een dubbelklik op de grafiek zal worden uitgezoomd met twee keer de huidige tijdschaal met de coördinaten van de muis als middelpunt.

Omdat de database gebruik maakt van AT-STYLE(tijdsnotatie) wordt deze hier ook gebruikt.

Met de knop boven aan de pagina kan worden veranderd van pagina.

6.3.3.2 Scadapagina

Op de Scadapagina zijn de huidige gegevens van het proces grafisch weergegeven. Deze gegevens worden periodiek ververst. Daarnaast wordt hier een verkleinde grafiek getoond van de laatste 30 minuten. Het type grafiek kan met de knoppen boven de grafiek worden geselecteerd. Naast deze informatie zijn ook de algemene gegevens van het systeem via tabbladen beschikbaar. Deze informatie bestaat voor temperaturen uit minimum, gemiddelde en maximum temperatuur. Voor de andere waardes bestaat deze uit de tijd die de betreffende IO in een bepaalde stand heeft gestaan. Deze gegevens worden één keer per dag ververst omdat dit veel tijd vereist van de database.

6.3.4 Mobiele applicatie

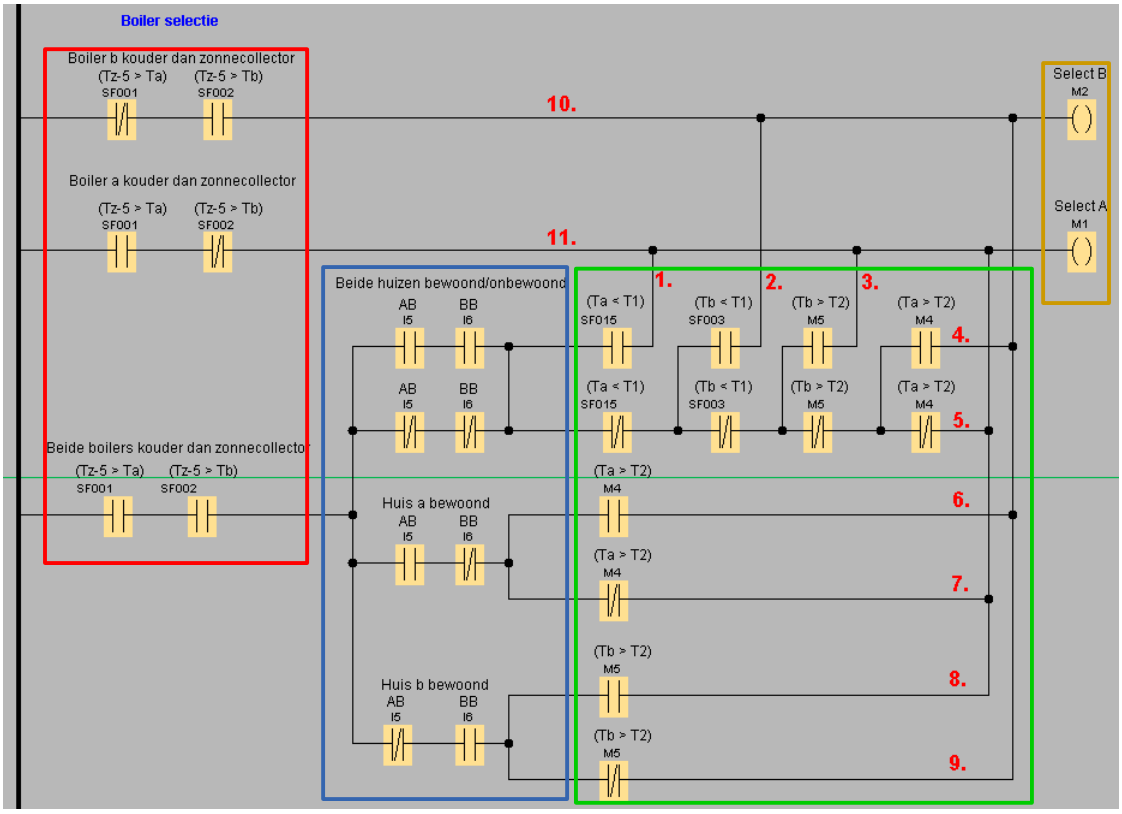
De mobiele applicatie is een uitbreiding op de site. Hiermee kan de gebruiker van een Androidtoestel de grafieken die op de site getoond worden ook via de applicatie bekijken.

7 Implementatie

In dit hoofdstuk wordt de implementatie van elk softwareonderdeel beschreven.

7.1 Procesregeling

De procesregeling is de kern van het project en is ontwikkeld met de door Siemens aangeleverde software. Binnen deze software is gebruik gemaakt van een ladderdiagram. In een ladderdiagram wordt met AND en OR-schakelingen het programma ontworpen. Naast deze AND en OR-schakelingen zijn er ook speciale blokken zoals timers en berekeningsblokken. In onderstaande figuur is een deel van de procesregeling te zien:



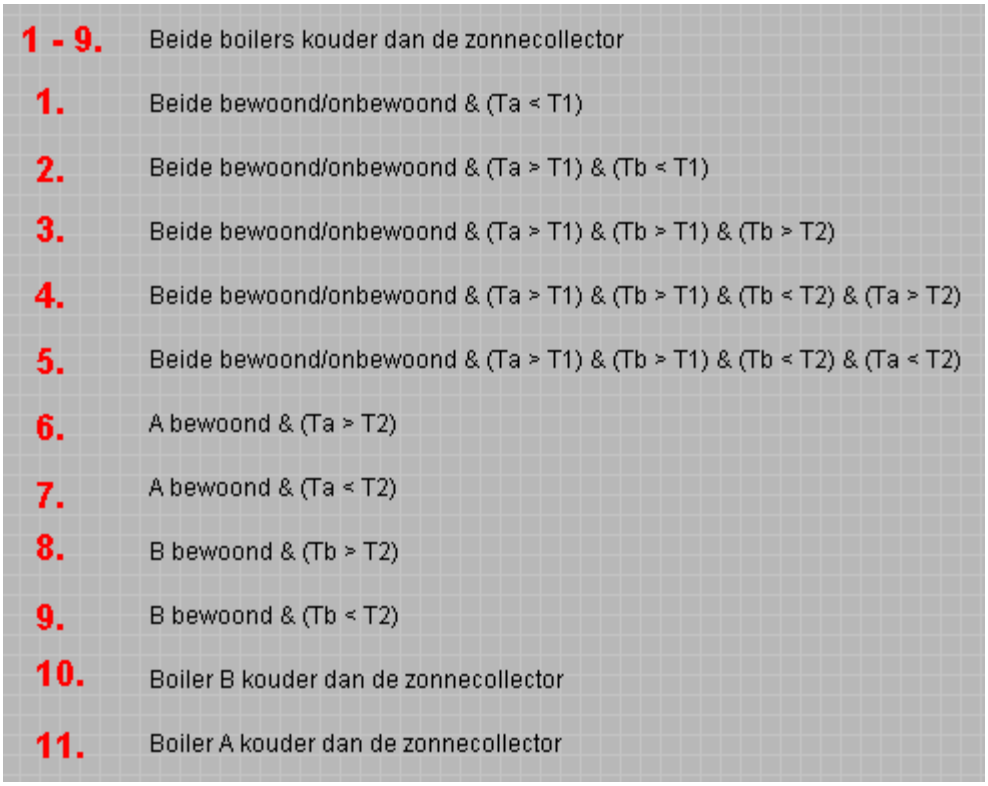
Figuur 15: Selectie voor de richting van de driewegklep.
De aangegeven (rode) getallen zijn nader uitgelegd in Figuur 16.

In deze afbeelding wordt gebruik gemaakt van drie verschillende blokken: contact, “NOT” contact en een coil. Door meerdere contacten en “NOT” contacten in serie te zetten ontstaat een logische AND-poort. Door meerdere contacten en “NOT” contacten parallel te zetten ontstaat een logische OR-poort. De uitkomst hiervan kan dan geleid worden naar een coil. Deze coil is meestal een uitgang of een flag. De contactblokken in dit schema zijn de uitkomsten van allerlei speciale blokken die berekeningen hebben uitgevoerd op de temperaturen van de boilers.(Zie voor meer uitleg 6.3.1 Processoftware) In het getoonde ladderdiagram worden de volgende stappen uitgevoerd:

1. Controleren of de boilers kouder zijn dan de zonnecollector. Dit is te zien in het rode vak.
2. Controleren van de bewoning van de huizen. Dit is te zien in het blauwe vak.
3. Vergelijken van de temperatuur met de ingestelde temperaturen. Dit is te zien in het groene vak.

4. Aan de hand van bovenstaande stappen wordt de goede flag positief gemaakt. Dit is te zien aan de lijnen die richting het gele vak gaan.

De rode nummers in de afbeelding geven de verschillende paden weer die genomen kunnen worden om tot de selectie van een boiler te komen. De vergelijkingen die bij deze paden horen zijn in onderstaande figuur afgebeeld.



Figuur 16: Paden die tot de keuze van de boilers leiden.
T1: Ingestelde threshold van de boiler
T2: Maximale temperatuur van de boiler
Ta: Temperatuur van boiler a
Tb: Temperatuur van boiler b

7.2 Interface

7.2.1 Maken en verbreken verbinding

De verbinding verloopt via een seriële verbinding. Deze verbindingen worden op een Linux besturingssysteem tty-poorten genoemd. Voor het maken en verbreken van de verbinding wordt gebruik gemaakt van de termios library van Linux. Binnen Linux kan een seriële poort op dezelfde manier als een bestand geopend worden, dit geeft een zogenoemde filedescriptor. Met de termios library kunnen bepaalde settings aan de filedescriptor worden gegeven.

Bij het verbreken van de verbinding wordt de seriële poort weer teruggezet naar de oorspronkelijke instellingen.

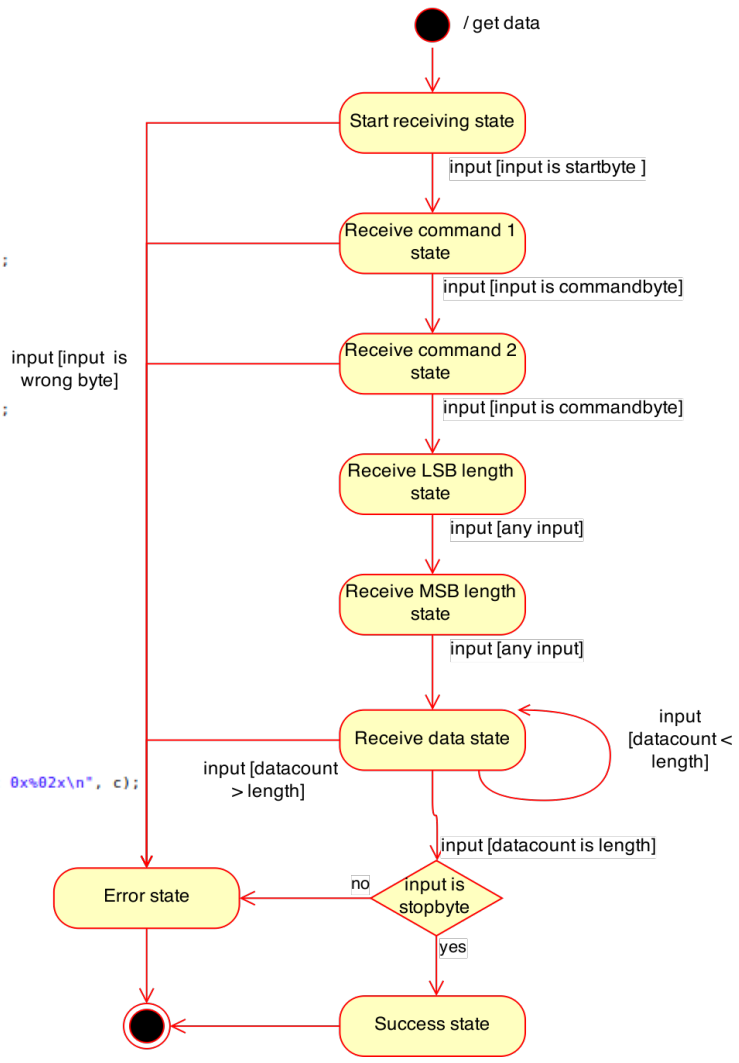
7.2.2 Versturen commando's

De interface bestaat uit twee basisfuncties die door alle andere functies gebruikt worden om met de PLC te communiceren: send en receive.

Voor het ontvangen van een datapakket wordt gebruik gemaakt van een statemachine. Hierbij wordt de state aangepast nadat een bepaalde byte is ontvangen.

In deze code worden eerst de startbyte en de commandobytes ontvangen. Dit zijn de eerste drie states. Daarna wordt twee keer een byte met de LSB en MSB van de datalengte ontvangen. Deze datalengte is nodig voor de laatste state. In de laatste state wordt het datapakket opgehaald. Hierbij wordt nog een controle uitgevoerd of de laatste byte een stopbyte is.

```
switch (state) {
case start_state: // Startbyte
    if (c == START_BYTE) {
        state = first_command_state;
        break;
    }
    log_error("Data: not the right start bit. Expected 0x55 got 0x%02x.\n", c);
    state = error_state;
    break;
case first_command_state: // Commandbyte
    if (c == DATA_COMMAND_BYTE) {
        state = second_command_state;
        break;
    }
    log_error("Data: not the right command bit. Expected 0x11 got 0x%02x.\n", c);
    state = error_state;
    break;
case second_command_state: // Commandbyte
    if (c == DATA_COMMAND_BYTE) {
        state = first_length_state;
        break;
    }
    log_error("Data: not the right command bit. Expected 0x11 got 0x%02x.\n", c);
    state = error_state;
    break;
case first_length_state: // LSB of length
    number_of_bytes = c;
    state = second_length_state;
    break;
case second_length_state: // MSB of length
    number_of_bytes += (c << 8);
    index = 0;
    state = data_state;
    break;
case data_state:
    if (index == number_of_bytes) {
        if (c == STOP_BYTE) {
            send_command(connection, stopdatatransparams, 5);
            receive(connection, &c, 1);
            state = success_state;
            break;
        }
        log_error("Data: not the right data format. Expected stop byte 0xAA. got 0x%02x\n", c);
        send_command(connection, stopdatatransparams, 5);
        receive(connection, &c, 1);
        state = error_state;
    }
    if (index < data_array_length) {
        data[index++] = c;
    } else {
        log_error("Data: The given data array is too small.\n");
        state = error_state;
    }
    break;
case success_state:
    return_status = number_of_bytes;
    break;
case error_state:
    return_status = error;
    break;
```



Code 1: State machine voor het ontvangen van een datapakket

7.2.3 Datapackage object.

Voor het opslaan van de gegevens wordt gebruik gemaakt van een datapackage struct, in deze struct zijn alle waarden opgeslagen die de PLC verstuurt bij het opvragen van de data. Voor deze struct zijn een aantal functies gemaakt die het gebruik vergemakkelijken. Omdat elke minuut van het aantal opgevraagde datapackages het gemiddelde wordt genomen zijn drie functies gemaakt:

1. Fill datapackage

   Deze functie kan gebruikt worden om een datapackage te vullen met de raw data die is opgevraagd.

2. Add datapackages

   Deze functie maakt het mogelijk om twee datapackages bij elkaar op te tellen.

3. Divide datapackage

   Deze functie deelt de waardes van een datapackage door het aantal datapackages dat in totaal hierin is opgeteld. Zo kan de gemiddelde waarde worden bepaald.

7.2.4 Database interface

Voor de connectie met de database wordt gebruik gemaakt van de RRDtool library. Aan de kant van de interface wordt alleen gebruik gemaakt van de insertfunctie. In de code is een wrapperfunctie gemaakt die uit een datapackage de gewenste waardes haalt en via de insertfunctie in de database opslaat. Een voorbeeld van de insertfunctie is hieronder weergeven.

```
char rrd_insert_package(char* database, struct data_package *package) {
    char buffer[256];
    snprintf(buffer, 256, "N:%d:%d:%d:%d:%d:%d:%d:%d:%d:%d:%d:%d:%d",
            package->ACTUAL_VALUES.TZ,
            package->ACTUAL_VALUES.TA,
            package->ACTUAL_VALUES.TB,
            package->ACTUAL_VALUES.TEXT,
            (package->DIS.I1_I8 >> 4) & 1, //BewoningA
            (package->DIS.I1_I8 >> 5) & 1, //BewoningB
            (package->DOS.Q1_Q8 >> 2) & 1, //HeaterA
            (package->DOS.Q1_Q8 >> 3) & 1, //HeaterB
            (package->DOS.Q1_Q8 >> 1) & 1, //Valve
            (package->DOS.Q1_Q8 & 1), //Pump
            package->ACTUAL_VALUES.THRESHOLD_A,
            package->ACTUAL_VALUES.THRESHOLD_B,
            package->ACTUAL_VALUES.FLOW_RATE);

    char *updateparams[] = { "update", database, buffer };

    optind = opterr = 0;
    rrd_clear_error();
    int status = rrd_update(3, updateparams);
    if (rrd_test_error()) {
        log_error("%s\n", rrd_get_error());
        log_error("Update status: %d.\n", status);
        return ERROR;
    }
    debug_message("Update status: %d.\n", status);
    return OK;
}
```

Code 2: Insert functie voor een datapackage

In de code worden de waarden van een package met behulp van snprintf in een bepaald format naar een buffer geschreven. Deze buffer wordt met een rrd_update toegevoegd aan de database.

RRDtool vereist dat de programmeur voor het toevoegen van waarden aan de database bepaalde variabelen reset. Nadat dit is gedaan kan met behulp van de functie rrd_update een insert op de database uitgevoerd worden.

7.2.5 Logging

Voor het loggen van gegevens zijn drie functies gemaakt:

1. Log string

   De standaard logfunctie. Deze functie stuurt de string naar stdout. Hier is een functie van gemaakt, zodat deze met een 'define' uitgeschakeld kan worden.

2. Log error

   De error logfunctie. Deze functie stuurt de string naar stderr. Hier is ook een functie van gemaakt, zodat deze met een 'define' uitgeschakeld kan worden.

3. Debug

   De debugfunctie. Deze functie stuurt de string naar stdout. Deze functie is een copy van Log string. Dit is gedaan zodat onderscheid gemaakt kan worden tussen debug strings en andere strings. Hierdoor is het mogelijk om verschillende soorten strings in of uit te schakelen.

Alle drie de functies voegen het tijdstip toe zodat later gezien kan worden wanneer een bepaalde string gelogd is.

7.2.6 Foutafhandeling

Binnen de software worden fouten op verschillende manieren afgehandeld:

1. Logging

   Voor veel fouten geldt dat deze geen grote invloed hebben op het programma. Hierbij kan gedacht worden aan een verkeerd ontvangen byte. In deze gevallen wordt de fout gelogd.

2. Buffer legen

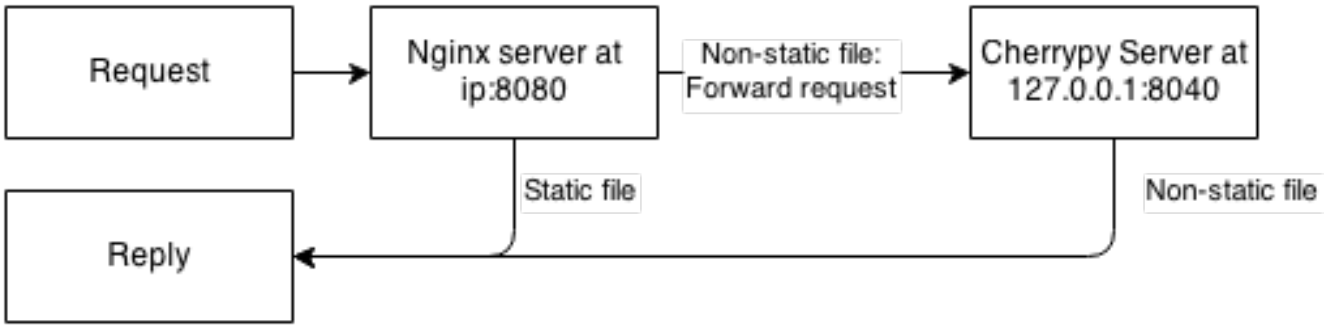
   Als een byte niet is ontvangen of de verkeerde byte is ontvangen, zal de software de buffer legen voor het volgende commando. Hierdoor heeft deze fout alleen invloed op het huidige commando.

3. Herstarten

   Sommige fouten hebben een grotere invloed op het programma, waardoor het programma niet meer goed zou kunnen functioneren. In dit geval wordt het programma afgesloten. Het besturingssysteem krijgt dan de opdracht om het programma weer op te starten.

7.3 Server

De server is met het webframework CherryPy gerealiseerd. Daarnaast worden de static files door een Nginx server geserveerd. Dit wordt ook wel reverse proxy genoemd. Er is hier voor Nginx gekozen omdat dit wordt aanbevolen en ondersteund(CherryPy Team, 2014). Hierbij wordt van de volgende constructie gebruik gemaakt:



Figuur 17: Server opstelling

7.3.1 Grafiek generatie

Voor het genereren van grafieken levert de RRDtool module twee functies:

1. graph
2. graphv

Bij graphv worden gedetailleerde gegevens over de grafiek geleverd. Daarom wordt deze functie gebruikt in plaats van de standaard graph functie. Deze gedetailleerde informatie is nodig om op de site extra functionaliteit te kunnen leveren bij de grafiek, zoals een overlay over de grafiek.

Aan deze functie wordt een lijst met argumenten meegegeven waarmee de grafiek gemaakt wordt. Bij deze argumenten zitten waarden zoals grafieklocatie, starttijd, eindtijd en titel. Een voorbeeld van een argumentenlijst is in onderstaande afbeeldingen te zien.

```
graph_location + name,
'--start', start,
'--end', end,
'--lower-limit', '0',
'--upper-limit', '100',
'--title', 'Primary Circuit',
'--vertical-label', 'Temperature (<span>&#8451;</span>)'
```

Code 3: Eerste deel argumentenlijst

```
'DEF:TCollector=' + database + ':TCollector:AVERAGE',
'DEF:TA=' + database + ':TA:AVERAGE',
'DEF:TB=' + database + ':TB:AVERAGE',
'DEF:ThresholdA=' + database + ':ThresholdA:AVERAGE',
'DEF:ThresholdB=' + database + ':ThresholdB:AVERAGE',
```

Code 4: Tweede deel argumentenlijst

In het eerste gedeelte van de argumentenlijst worden de algemene instellingen vastgesteld. Dit zijn bijvoorbeeld start, einde en titel.

In het tweede gedeelte van de argumentenlijst worden de gegevens vanuit de database in een variabele gestopt. Hierbij wordt uit de gegeven database de waarde met de bijbehorende consolidatie gehaald.

In het derde gedeelte van de argumentenlijst wordt de grafiek gevuld en de legenda daarbij gemaakt. Deze argumenten moeten op de volgorde van de legenda worden toegevoegd. Dit gedeelte wordt samengesteld uit 3 commando's:

1. COMMENT

   Een comment komt in de legenda te staan. Comments worden vaak gebruikt voor de opmaak van de legenda.

2. LINE

   Dit is een lijn op de grafiek. Omdat de tekst van dit commando in de legenda komt te staan moeten deze commando's tussen de andere legenda commando's komen te staan.

3. GPRINT

   Met GPRINT kan een waarde in de legenda worden geprint.

```
'COMMENT:Actual values\\:\\l',
'LINE1:TCollector#0000FF:' + 'Collector'.ljust(21),
'GPRINT:TCollector:LAST:%3.1lf <span>&#8451;</span>',

'COMMENT:\\u',

'LINE2:ThresholdA#FFFF00:Threshold ' + kwargs.get('house_a_name', 'A').ljust(15) + '       \\r',

'COMMENT:\\u',

'GPRINT:ThresholdA:LAST:%3.1lf <span>&#8451;</span>\\r',

'LINE1:TA#00FF00:House ' + kwargs.get('house_a_name', 'A').ljust(15),
'GPRINT:TA:LAST:%3.1lf <span>&#8451;</span>\\l',

'COMMENT:\\u',

'LINE2:ThresholdB#00FFFF:Threshold ' + kwargs.get('house_b_name', 'B').ljust(15) + '       \\r',

'COMMENT:\\u',

'GPRINT:ThresholdB:LAST:%3.1lf <span>&#8451;</span>\\r',

'LINE1:TB#FF0000:House ' + kwargs.get('house_b_name', 'B').ljust(15),
'GPRINT:TB:LAST:%3.1lf <span>&#8451;</span>\\l',
```

Code 5: Derde deel argumentenlijst

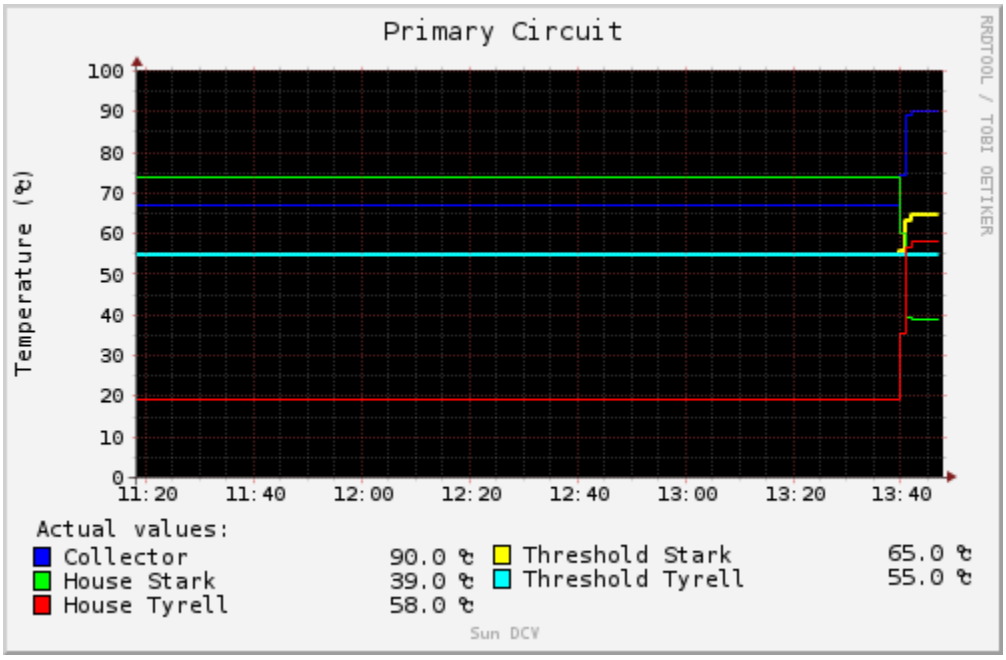
```
_GLOBAL_GRAPH_ARGS = [
    '--height', '325',
    '--width', '500',
    '--full-size-mode',
    '--color', 'CANVAS#000000',
    '--watermark', 'Sun DCV',
    '--pango-markup',
    '--lazy'
]
```

Code 6: Globale argumenten

Naast deze delen van de argumentenlijst is gekozen om elke grafiek een aantal vaste waardes te geven. Deze waardes worden bij elke argumentenlijst aan het einde toegevoegd.

Deze algemene argumenten hebben vooral invloed op het uiterlijk van de grafiek. Hierbij behoren de grootte van de totale afbeelding, de kleur van de grafiek en het watermerk dat aan elke grafiek wordt toegevoegd.

Door deze argumenten aan de graphv functie te geven zal de onderstaande grafiek ontstaan.



Grafiek 2: Primary Circuit grafiek

7.4 Website

De website is met behulp van HTML5 gemaakt. Daarnaast is de site opgebouwd met het gebruik van de volgende onderdelen:

1. Jinja templating tool.

   Er is voor Jinja gekozen omdat hier een module voor CherryPy voor is. Daarnaast is Jinja gemaakt met als voorbeeld het Django templating systeem(Ronacher, 2008).

2. Javascript/jQuery/jQuery UI.

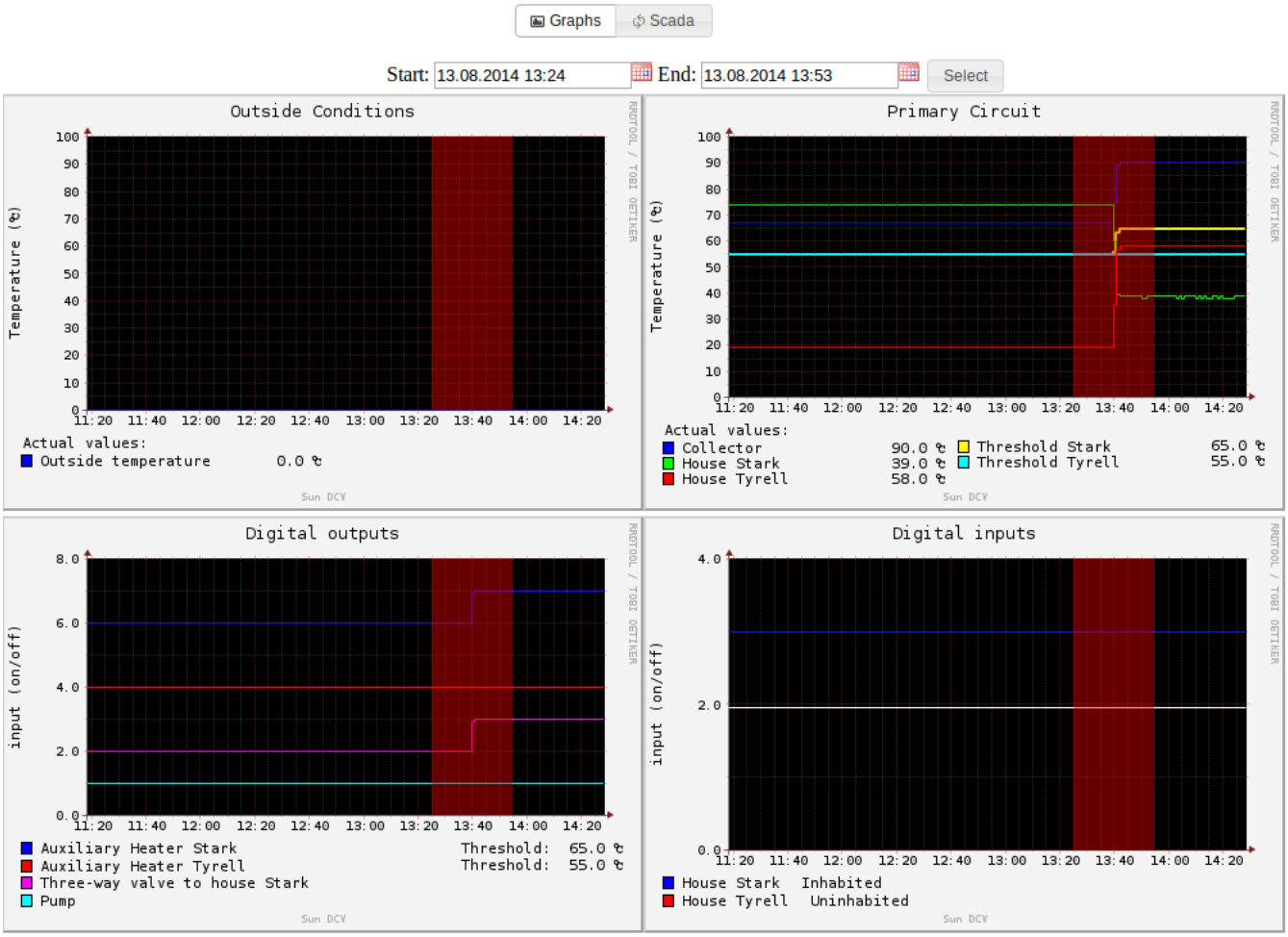
   Er is gekozen voor jQuery en jQuery UI omdat hier al eerder ervaring mee is opgedaan.

7.4.1 Jinja templating tool

Met de Jinja templating tool kan extra informatie worden toegevoegd aan de website. Dit wordt al aan de serverkant van de website gedaan. Aan deze templating tool kan een lijst van key-value pairs worden meegegeven. Deze waarden kunnen via tags aan de website toegevoegd worden.

7.4.2 Javascript/jQuery/jQuery UI

Met behulp van Javascript en jQuery is extra functionaliteit toegevoegd aan de website. Hierbij kan gedacht worden aan een overlay over de grafiek waarmee een deel van de grafiek te selecteren is. Dit is te zien in Figuur 18.



Figuur 18: Grafieken met rode selectie overlay.

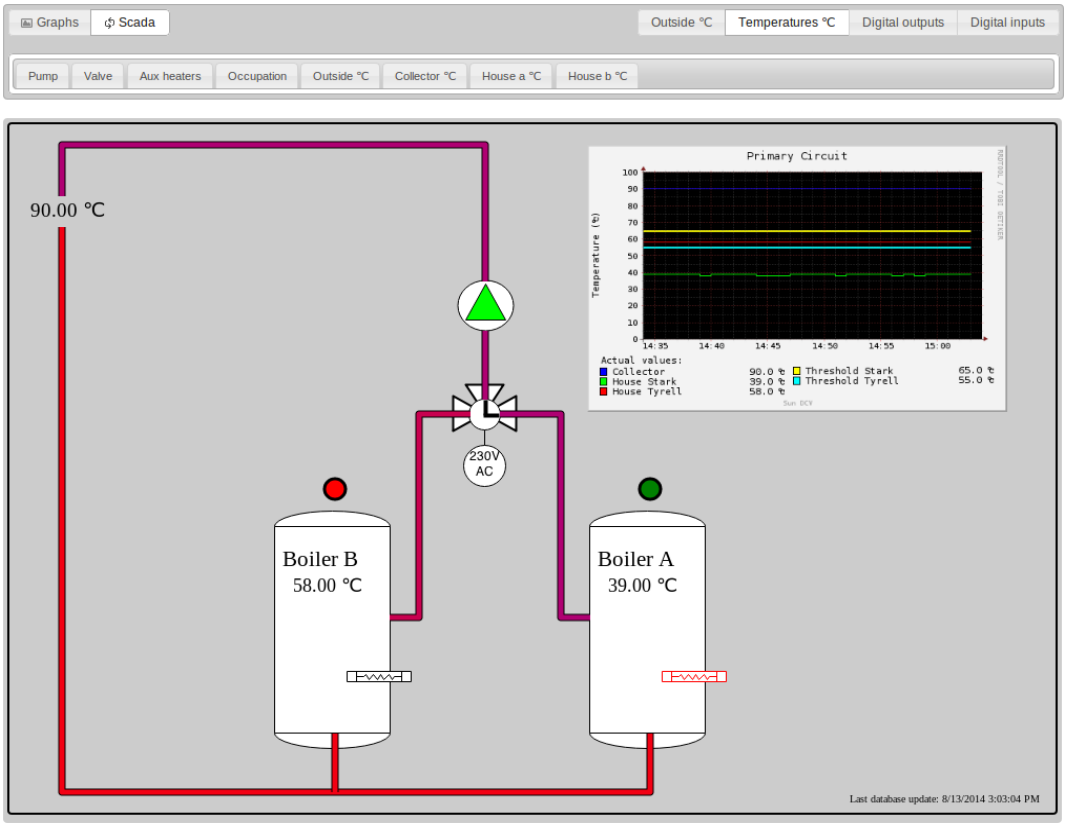
Zoals te zien in bovenstaande figuur wordt het begin en het einde van het selectievak in de start- en eindvelden ingevoerd. Dit wordt uitgevoerd door Javascript en jQuery. Daarnaast wordt ook nog gebruik gemaakt van de jQuery UI om de knoppen een mooier uiterlijk te geven en aan elk invoerveld een datepicker toe te voegen. De functie die gebruikt wordt bij het bepalen van de start- en eindtijd werkt volgens de volgende formules:

$$stepsize = \frac{end - start}{graphwidth}$$
$$start_{new} = start_{old} + stepsize * x_{low}$$
$$end_{new} = end_{old} + stepsize * x_{high}$$

Bij deze formules zijn de start- en eindwaarde in seconden sinds epoch. Hierdoor zijn minder conversiestappen nodig van het geselecteerde vak naar een datum. De uiteindelijke data die in de velden worden ingevoerd moeten voldoen aan de AT-STYLE. Dit is een tijdsnotatiestijl in Linux die gebruikt wordt door het at commando. Er is voor deze stijl gekozen omdat de RRDtool deze stijl gebruikt als tijdsnotatie. Daardoor is het mogelijk om deze data direct aan de database aanvraag te geven, zonder dat nog conversie noodzakelijk is.

7.4.3 Scada canvas

Naast de pagina met de grafieken is ook een Scadapagina ontworpen. Op deze pagina worden net zoals bij een Scadasysteem de huidige waarden van het systeem in een model getoond. Dit gehele model wordt op een HTML5 Canvas getekend. De enige mogelijkheid om op een HTML5 Canvas te tekenen is met Javascript.

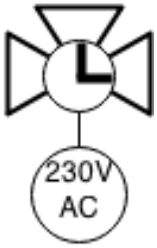
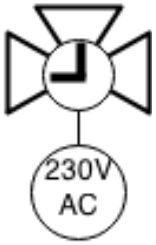
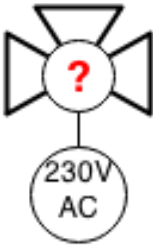
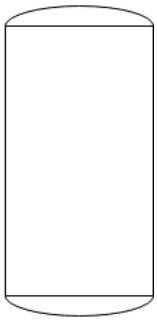


Figuur 19: Scadamodel van het gesimuleerde systeem.

Als de pagina wordt opgevraagd worden alle afbeeldingen voor het Scadamodel ingeladen. Deze afbeeldingen worden in een dictionary gestopt, waaruit dan de afbeelding naar keuze gehaald kan worden.

Dit model vraagt elke minuut nieuwe gegevens van de server. De server reageert daarop met een JSON-dataobject met daarin de nieuwe huidige gegevens.

7.4.3.1 Scadacanvas onderdelen

Onderdeel	Betekenis	Onderdeel	Betekenis	Onderdeel	Betekenis
	Pomp aan.		Pomp uit.		Pomp status onbekend.
	Klep rechts geselecteerd.		Klep links geselecteerd.		Klep richting onbekend.
	Heater aan.		Heater uit/ onbekend.		
	Huis bewoond.		Huis niet bewoond.		Bewoning onbekend.
	Boiler.		Leiding.		Het kleur verloop van de leiding. Boven is 100 ℃ en onder is 0 ℃. Bij een onbekende temperatuur zal de leiding wit zijn.

In deze tabel zijn alle onderdelen binnen het systeem op een overzichtelijke manier weergegeven. Hierin is ook te zien welke verschillende mogelijkheden elk onderdeel heeft.

7.4.3.2 Leidingtemperatuur kleur bepalen

De leiding kleur wordt bepaald aan de hand van vier waarden:

1. Leidingtemperatuur

   Dit is de huidige temperatuur van het water in de leidingen.

2. Switchtemperatuur

   Dit is de temperatuur waarbij het water onder deze temperatuur als koud moet worden beschouwd. Als het water als koud wordt beschouwd krijgt de watertemperatuur een blauwere kleur.

3. Minimumtemperatuur

   De minimumtemperatuur van de leiding. Bij deze temperatuur is de kleur volledig blauw.

4. Maximumtemperatuur

   De maximumtemperatuur van de leiding. Bij deze temperatuur is de kleur volledig rood.

```
if (temp < switch_temp) {
    step_size_blue = 105 / (switch_temp - min_temp);
    step_size_red = 150 / (switch_temp - min_temp);

    blue = Math.round(255 - (step_size_blue * (temp - min_temp)));
    red = Math.round(150 - (step_size_red * (switch_temp - temp)));
} else {
    step_size_blue = 150 / (max_temp - switch_temp);
    step_size_red = 105 / (max_temp - switch_temp);

    blue = Math.round(step_size_blue * (max_temp - temp));
    red = Math.round(255 - (step_size_red * (max_temp - temp)));
}
```

Code 7: Berekening RGB kleuren voor temperatuur.
Deze functie levert de volgende kleurovergang bij de volgende variabelen:

$$Switch\,temp = 20$$
$$Minimum\,temp = 0$$
$$Maximum\,temp = 100$$

Deze kleurovergang wordt gebruikt om de leidingen hun kleur te geven. De zwarte lijn geeft de switch temperatuur aan. Zoals in de afbeelding te zien is, wordt vanaf de switchtemperatuur de kleur snel blauwer.

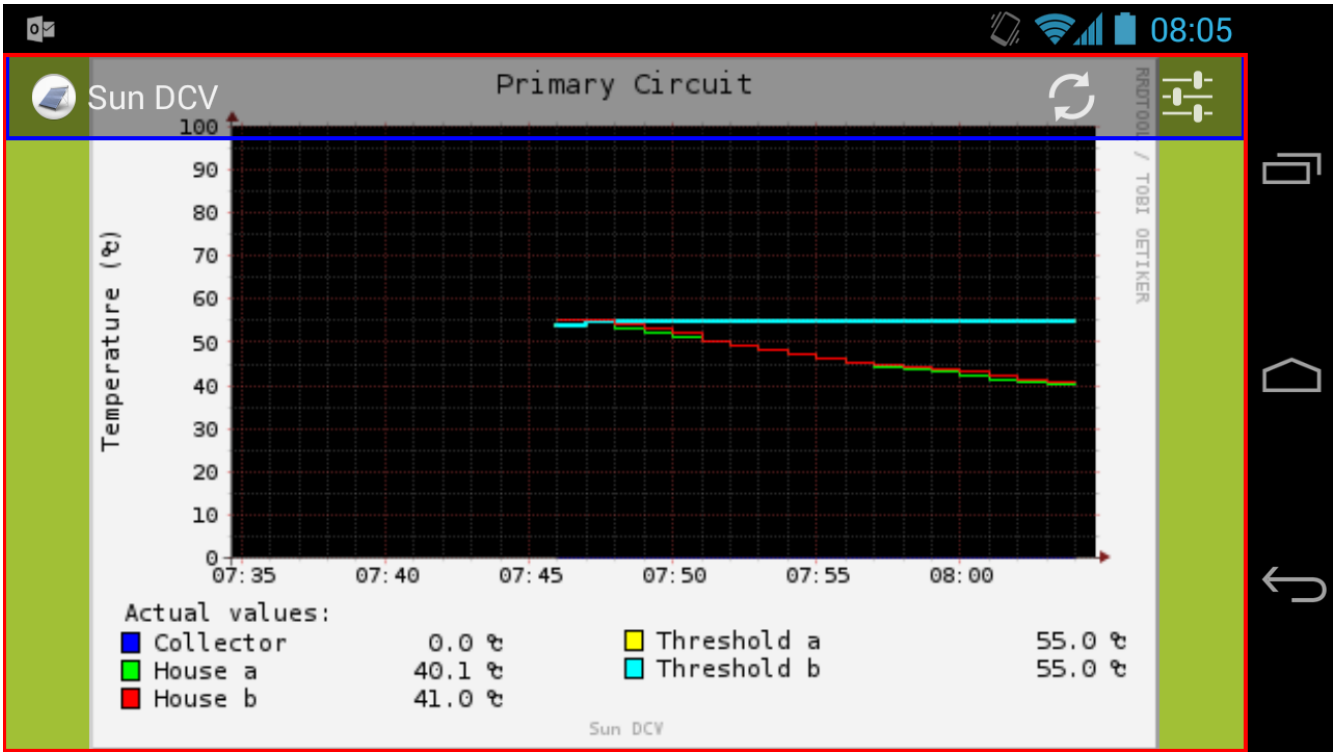
Figuur 20: Kleurovergang

7.5 Mobiele applicatie

De mobiele applicatie is alleen voor Android toestellen gemaakt. De applicatie is opgedeeld in drie onderdelen:

1. Graphs screen
2. Graph load tasks
3. Settings

7.5.1 Grafiek view



Figuur 21: De grafiek view met beide hoofdonderdelen omlijnd.

De grafiek view bestaat uit twee hoofdonderdelen: de grafiek viewer en de actionbar. In Figuur 21 is de grafiek view te zien met de hoofdonderdelen omlijnd. De actionbar(met blauw omlijnd) wordt met een touch op het scherm zichtbaar. Via deze actionbar kunnen de grafieken worden herladen. Daarnaast kan via de actionbar het settingsmenu worden bereikt.

Voor de grafiekviewer is gebruik gemaakt van een viewpager. Met een viewpager kan door het vegen over het scherm van grafiek worden gewisseld.

In onderstaande code is het opvragen van een afbeelding van de viewpager uitgewerkt.

```
@Override
public Object instantiateItem(ViewGroup container, int position) {
    setProgressBarIndeterminateVisibility(true);
    Context context = ViewGraphs.this;
    ImageView imageView = new ImageView(context);
    imageView.setPadding(0, 0, 0, 0);
    imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);

    SharedPreferences sharedPref = PreferenceManager.getDefaultSharedPreferences(activity);
    String serverURL = "http://" + sharedPref.getString(SettingsActivity.KEY_SERVER_URL,"0.0.0.0:8080");
    String startTime = sharedPref.getString(SettingsActivity.KEY_START_TIME, "N-4Hours");
    String endTime = sharedPref.getString(SettingsActivity.KEY_END_TIME, "N");
    GetImageURLTask task = new GetImageURLTask(imageView, this.activity, serverURL);
    task.execute("/refresh_graph?graph_type=" + graphs.get(position) + "&start=" + startTime + "&end=" + endTime);
    ((ViewPager) container).addView(imageView, 0);
    imageView.setOnClickListener(new OnClickListener() {
        @Override
        public void onClick(View v) {
            if (getActionBar().isShowing()) {
                getActionBar().hide();
            } else {
                getActionBar().show();
            }
        }
    });
    return imageView;
}
```

Code 8: Code voor het laden van een afbeelding.

Bij het laden van een grafiek worden de server URL, starttijd en eindtijd uit de settings gehaald. Hierna wordt een asynchrone taak gestart die zal proberen de afbeelding van de server op te halen. Daarnaast wordt aan de afbeelding een listener toegevoegd die bij een onclick de actionbar laat zien.

7.5.2 Grafiek load tasks

7.5.2.1 Get image URL task

Om een grafiek te laden moet eerst de URL van de grafiek bekend zijn. Daarvoor zorgt deze taak. Deze taak wordt in de grafiekviewer aangemaakt met een Weakreference naar de ImageView waar de afbeelding in moet worden getoond. Er is hier gekozen voor een Weakreference zodat tijdens het laden de ImageView ook weer verwijderd kan worden zonder dat dit voor memory leaks zorgt.

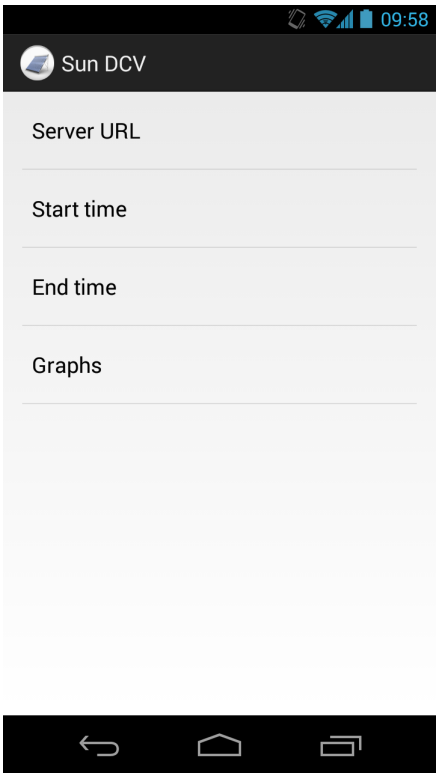
De get image URL task zal de refresh_graph URL van de server aanroepen, hierop zal een jsonobject terug worden gestuurd met daarin onder andere de URL van de grafiek. Als een URL is ontvangen zal de graph download task worden gestart om de grafiek te downloaden. In alle andere gevallen zal een afbeelding getoond worden met het bericht dat geen afbeelding beschikbaar is.

7.5.2.2 Graph download task

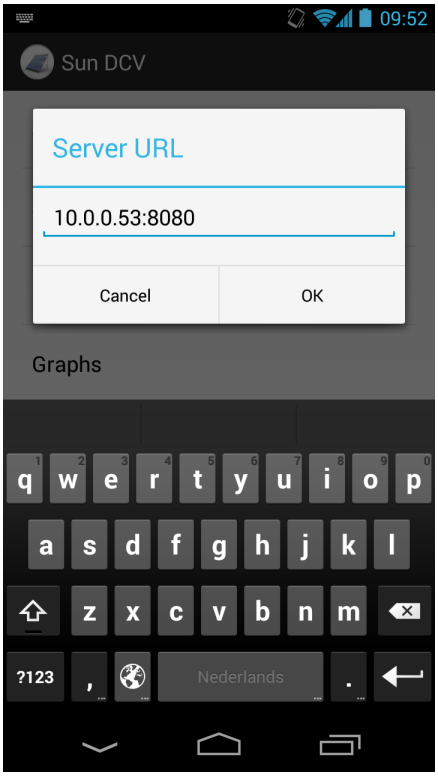
Voor het ophalen van de URL is een httpclient gebruikt. Voor het downloaden is in plaats van een httpclient een inputstream gebruikt. Door het gebruik van een inputstream kan met de BitmapFactory een bitmap gecreëerd worden. Als de afbeelding geladen is zal de afbeelding worden toegevoegd aan de ImageView. Ook zal de ProgressBar stil gezet worden.

7.5.3 Settings

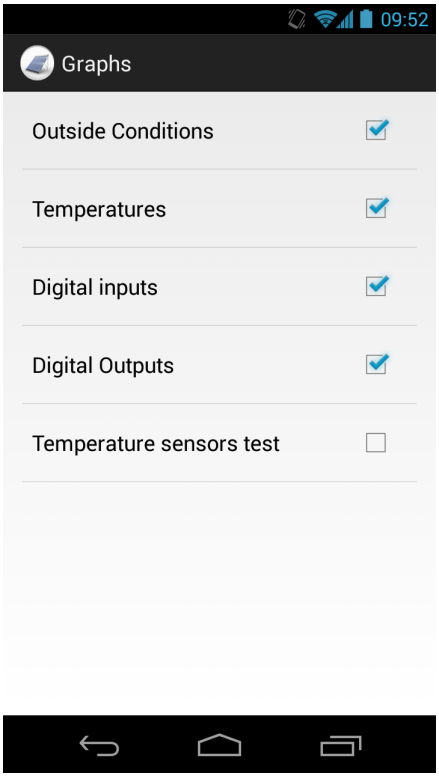
Via de actionbar kunnen de settings bereikt worden. Bij het openen van de settings zal het scherm zoals te zien in Figuur 22 getoond worden. Zoals in de figuur te zien is, zijn 4 settings aan te passen. Drie van deze settings zijn text fields die aangepast kunnen worden zoals te zien is in Figuur 24. De andere setting is een lijst waaruit met checkboxes bepaalde waardes geselecteerd kunnen worden. Deze setting wordt in een apart scherm geopend, dit is te zien in Figuur 23.



Figuur 22: Settings screen



Figuur 24: Settings server URL wijzigen



Figuur 23: Settings grafieken selecteren

Voor de settings wordt gebruik gemaakt van een Activity class en een PreferenceFragment class. De PreferenceFragment wordt gebruikt om de preferences/settings op een nette manier te laten zien. Deze PreferenceFragment moet aan een Activity gekoppeld worden waardoor dus ook een Activity class aanwezig moet zijn. Deze activity laadt alleen de PreferenceFragment in. De PreferenceFragment bouwt zijn lijst van preferences op aan de hand van een XML-file met preferences. Een voorbeeld van een preferencelijst die de bovenstaande preferences genereert is te zien in Figuur 25.

```
<PreferenceScreen>
  <EditTextPreference android:key="serverURL" android:title="@string/server_url" android:defaultValue="0.0.0.0:8080"/>
  <EditTextPreference android:key="startTime" android:title="@string/start_time" android:defaultValue="N-4Hours"/>
  <EditTextPreference android:key="endTime" android:title="@string/end_time" android:defaultValue="N"/>
-<PreferenceScreen android:key="graphs" android:title="@string/graphs" android:persistent="false">
    <CheckBoxPreference android:key="outsideConditions" android:title="@string/graph_oc" android:defaultValue="true"/>
    <CheckBoxPreference android:key="analogInputs" android:title="@string/graph_ai" android:defaultValue="true"/>
    <CheckBoxPreference android:key="digitalInputs" android:title="@string/graph_di" android:defaultValue="true"/>
    <CheckBoxPreference android:key="digitalOutputs" android:title="@string/graph_do" android:defaultValue="true"/>
    <CheckBoxPreference android:key="sensorsGraph" android:title="@string/graph_sensors" android:defaultValue="true"/>
  </PreferenceScreen>
</PreferenceScreen>
```

Figuur 25: Preferences XML file

7.5.3.1 Text field settings

De text fields settings zijn:

1. Server URL

   Via deze settings kan het IP-adres en de port van de server aangepast worden. Deze settings is alleen noodzakelijk voor development omdat de server URL tijdens development niet statisch is.

2. Start time

   De starttijd van de grafieken kan hier gewijzigd worden. De tijd die hier wordt ingevuld moet in AT-STYLE ingevuld worden.

3. End time

   De eindtijd van de grafieken kan hier gewijzigd worden. De tijd die hier wordt ingevuld moet in AT-STYLE ingevuld worden.

7.5.3.2 Checkbox settings

Voor de checkbox setting wordt een extra scherm geopend. In dit nieuwe scherm kunnen dan de gewenste grafieken geselecteerd worden. Na het wijzigen van deze lijst zullen de grafieken op het hoofdscherm herladen worden. Als hier geen grafieken geselecteerd worden zal het hoofdscherm leeg blijven, waarbij de actionbar altijd zichtbaar blijft.

8 Testen & Acceptatie

In dit hoofdstuk worden de testmethoden en acceptatiecriteria van elk software onderdeel beschreven.

8.1 Processoftware

De processoftware is de software die de installatie regelt, deze software staat op de PLC.

8.1.1 Testmethoden

De processoftware staat op een PLC, daardoor is het niet mogelijk om deze software met functionele tests te testen. Dit maakt test driven development voor dit deel van de software erg lastig. Binnen de standaard van het bedrijf is een deel gespecificeerd voor het testen van softwareonderdelen. Hierbij wordt een document opgesteld waarin alle functionele eisen en testen worden beschreven. Hierbij wordt elke eis aan een test gekoppeld zodat duidelijk blijkt dat aan een eis wordt voldaan. De testen uit dit document worden uitgevoerd waarna hiervan een verslag wordt gemaakt.

8.1.2 Acceptatiecriteria

De processoftware wordt op basis van de volgende onderdelen geaccepteerd:

1. Een testrapport waaruit blijkt dat alle testen slagen.
2. Een demo van de functionaliteit op de testopstelling.

8.2 Interface

De software voor de interface is geschreven in C. Omdat er bij dit project is uitgegaan van TTD(test driven development) worden eerst tests geschreven voor elk deel van de functionaliteit. Hiervoor is de testengine CUnit[6] gebruikt, hiervoor is gekozen omdat er al ervaring was met de CUnit testengine.

8.2.1 Testmethoden

Voor de interface software is er gebruik gemaakt van test driven development. Deze tests zijn niet voor elk deel van de software geschreven. Het is belangrijk dat een groot deel van de code met functionele tests wordt getest(code coverage), maar het nut van de test moet groot genoeg zijn om de tijd die erin is gestopt te verantwoorden. Hieronder staat de test voor het controleren of een tijd door de PLC wordt geaccepteerd.

```
void test_check_compatible_time(void) {
    time_t rawtime = time(NULL);
    struct tm datetime = *localtime(&rawtime);
    CU_ASSERT(check_compatible_time(&datetime) == ok);
    datetime.tm_mon = 13;
    CU_ASSERT(check_compatible_time(&datetime) == uncompatible_time);
    datetime.tm_mon = 1;
    datetime.tm_hour = 24;
    CU_ASSERT(check_compatible_time(&datetime) == uncompatible_time);
    datetime.tm_hour = 1;
    datetime.tm_mday = 32;
    CU_ASSERT(check_compatible_time(&datetime) == uncompatible_time);
}
```

De overige functionaliteit wordt op dezelfde manier getest als de processoftware. Dit gebeurt door het opstellen

6 http://cunit.sourceforge.net/doc/index.html

van tests in een document en deze tests uit te voeren.

8.2.2 Acceptatiecriteria

De interfacesoftware wordt op basis van de volgende onderdelen geaccepteerd:

1. Eent testrapport waaruit blijkt dat alle testen slagen.
2. Het uitvoeren van de unittests levert geen fouten.

8.3 Server

De software voor de server is geschreven in Python. Hierbij is er niet gebruik gemaakt van TTD, dit komt omdat de overhead van TTD te groot wordt voor de server, de tests zouden dan namelijk groter zijn dan de server zelf. Daarnaast genereert de server vooral afbeeldingen. Het genereren van deze afbeeldingen kan wel getest worden, maar het testen van de inhoud van de afbeeldingen is niet mogelijk.

8.3.1 Testmethoden

De software voor de server is getest door het opstellen van een testdocument aan de hand van de functionele eisen. Hierbij gaat het bijvoorbeeld om de informatie die in de grafieken getoond wordt.

8.3.2 Acceptatiecriteria

De Serversoftware wordt geaccepteerd als uit een testrapport blijkt dat alle testen slagen.

8.4 Website

De eisen aan de website zijn minimaal, er wordt alleen gespecificeerd welke informatie de website moet tonen.

8.4.1 Testmethoden

De Javascript die bij de website wordt gebruikt, wordt met JSHint[7] gecontroleerd, er zijn voor dit deel geen unittests gemaakt.

8.4.2 Acceptatiecriteria

De website wordt geaccepteerd op het moment dat hij de gewenste informatie toont. Dit wordt bewezen door een demo te geven aan de product owner. Daarnaast moet de software JSHint geen fouten geven.

7 http://www.jshint.com/

9 Motivering keuze methoden en technieken

Voor dit project moesten allerlei keuzes worden gemaakt, zoals de te gebruiken methoden en technieken. Het systeem is opgedeeld in verschillende delen software:

1. De PLC
2. De interface
3. De database
4. De server & website
5. De mobiele applicatie

De keuzes en hun reden worden hieronder per software deel beschreven.

9.1 PLC

9.1.1 Programmeertaal: Ladderdiagram

Voor het programmeren op de PLC is de Siemens Logosoft software gebruikt. Deze software biedt de mogelijkheid tot het programmeren met twee soorten diagrammen: een ladderdiagram of een functieblokkendiagram. Binnen dit project is gekozen voor het programmeren met een ladderdiagram.

Dit is gekozen omdat in het ontwerp van de software veel OR en AND logische schakelingen zitten. Deze schakelingen zijn met een ladderdiagram duidelijker uit te werken dan met een functieblokkendiagram.

Een nadeel van deze keuze is het onderhoud van het programma voor een programmeur die een logisch blokkendiagram gewend is. Daarnaast is gebleken uit gesprekken met andere programmeurs dat het ladderdiagram een oud diagram is, dat steeds minder gebruikt wordt en steeds vaker wordt vervangen door het functieblokkendiagram.

9.2 Interface

9.2.1 Programmeertaal: C

Voor het ontwikkelen van de interface met de PLC is de programmeertaal C gekozen. De programmeertaal C wordt door de gekozen database ondersteund en voor het implementeren van de interface via een seriële verbinding is veel informatie te vinden. Ook gebruikt C minder werkgeheugen dan bijvoorbeeld Java. Dit is een bijkomend voordeel omdat de applicatie op de Raspberry PI moet werken.

Door het kiezen voor C is het een klein programma gebleven die de Raspberry PI niet overbelast.

9.2.2 Blocking read

Voor het wachten op de gegevens van de PLC wordt gebruik gemaakt van een select op een filedescriptor. Dit geeft het effect van een blocking read. Hierdoor geeft de applicatie de CPU vrij zodat deze ook door andere applicaties, zoals de server en scripts, gebruikt kan worden.

9.2.3 Eén byte per read

Er is gekozen voor het lezen van maximaal één byte per keer. Dit is gedaan om meer controle te houden over het ontvangen van de informatie. Een bijkomend voordeel is het sneller kunnen verwerken van de gegevens.

9.2.4 Update interval

Voor het updaten van de waardes in de database is gekozen voor een interval van één minuut. Er is hier eerder een kleiner interval gebruikt van één seconde. Dit interval blijkt geen toegevoegde waarde te hebben vergeleken met de temperatuurveranderingen in het systeem. Een kleiner interval heeft wel forse nadelen zoals extra belasting van de PLC en minder stabiliteit. Daarom is het interval vergroot naar één minuut, dit is volgens de eisen ook de minimum interval voor het updaten. De waarde die wordt opgeslagen is het gemiddelde van alle gemeten waarden in die minuut. Dit zorgt voor een gelijkmatiger grafiek. Door een groter interval te kiezen wordt ook de database ontlast wat een bijkomend voordeel is voor de Raspberry PI.

9.3 Database

9.3.1 Archieven

De gebruikte database, de RRDtool, maakt gebruik van archieven. In deze archieven wordt de data van de PLC opgeslagen. Er is gekozen voor drie archieven met update intervallen van één minuut, één uur en één dag. Hiervoor is gekozen omdat het genereren van grote grafieken veel moeite kost voor de database vanwege de vele datapunten die dan gebruikt worden. Door voor deze grafieken een minder nauwkeurig update interval te gebruiken kan het genereren van de grafieken sneller en met minder gebruik van de CPU.

Het nadeel van het gebruiken van meerdere RRA's is het grotere datagebruik. De tweede en derde RRA voegen in vergelijking met de eerste RRA weinig extra datagebruik toe, terwijl ze het maken van grafieken sneller maken. De voordelen wegen daarom op tegen de nadelen.

9.3.2 Minimale en maximale waarden

Alle velden in de database hebben een minimale en een maximale waarde. Voor deze waarden zijn getallen gekozen waarbij alle waarden die het systeem zou kunnen leveren daar binnen vallen. Deze waarden zijn gekozen omdat dit door de database ontwerper is aangeraden.

Door het aangeven van deze waarden zullen de fout ingevoerde waarden er direct uitgehaald worden en worden omgezet naar onbekend. Hierdoor is een extra controle toegevoegd om de kwaliteit van de grafieken te garanderen.

9.4 Server en website

9.4.1 Programmeertaal: Python

Voor het ontwikkelen van de server is gekozen voor de programmeertaal Python, omdat ik al eerder ervaring opgedaan heb met het programmeren van een webapplicatie in Python. Python is daarnaast ook één van de talen die ondersteund wordt door de gekozen database. Daarnaast kon hierdoor voor sommige delen al bestaande code worden gebruikt.

Door het kiezen voor Python is het makkelijker geworden voor de programmeur om de webserver op te zetten. Dit komt door de vele ondersteuning op het web die er voor deze programmeertaal in combinatie met een webapplicatie is.

9.4.2 Webframework: CherryPy & Nginx

Voor het webframework van de server is gekozen voor CherryPy. Dit is een in Python geschreven webframework die het bouwen van een webserver vergemakkelijkt. De website is met twee pagina's een kleine website te noemen, waardoor CherryPy in aanmerking komt. Ook biedt CherryPy ondersteuning voor templating

wat het genereren van non-statische pagina's eenvoudiger maakt. Een alternatief voor CherryPy zou Django zijn, hiervoor is juist niet gekozen vanwege de specialisatie voor een object gebaseerde database. Er wordt gebruik gemaakt van een time-series database waardoor het voordeel van Django verviel. Deze keuze is gebaseerd op de keuze voor de programmeertaal Python, daarom is niet afgewogen tegen webservers die niet met Python werken.

Tijdens het gebruik van CherryPy viel op dat de server niet goed kon omgaan met static files. Daarom is voor het serveren van static files gekozen voor Nginx. Deze server is gekozen omdat het een zeer kleine server is die snel is op te zetten.

Door te kiezen voor CherryPy was het nodig om een extra server toe te voegen voor de static files. Daarnaast biedt CherryPy ondersteuning voor templating wat het genereren van non-statische pagina's vergemakkelijkt. Door hier voor Nginx te kiezen was dit nadeel geen groot probleem.

9.4.3 Onbeveiligde toegang

Voor de toegang tot de server is voor een onbeveiligde toegang gekozen. De website bevat geen persoonlijke informatie en het is dus geen probleem als andere personen deze informatie te zien krijgen. Daarnaast is het onmogelijk om vanuit de webapplicatie informatie in de database of het systeem te wijzigen.

De keuze voor een onbeveiligde toegang heeft ervoor gezorgd dat de applicatie zonder het invullen van gebruikersnaam en wachtwoord te bereiken is. Dit zorgt ervoor dat de bewoner de informatie sneller kan bekijken. De onbeveiligde toegang heeft geen nadelen doordat het onmogelijk is om de database aan te passen vanuit de webapplicatie.

9.4.4 Javascript en jQuery

Voor het tonen van de gegevens op de website wordt gebruik gemaakt van Javascript en jQuery. Een reden hiervoor is dat Javascript en jQuery op bijna alle apparaten ondersteund worden. Daarnaast zorgt dit ervoor dat de server minder belast wordt doordat Javascript en jQuery op de client-side worden uitgevoerd. De alternatieven zoals Java Applets of C# Silverlight zijn net zoals Javascript afhankelijk van bepaalde onderdelen die door de gebruiker geïnstalleerd moeten worden. Omdat Java Applets al oud zijn en weinig gebruikt worden zijn deze afgevallen. Mijn ervaring met C# Silverlight is veel minder dan met Javascript. Dit heeft dan ook geleid tot de keuze voor Javascript.

De keuze voor Javascript en jQuery heeft ervoor gezorgd dat op de site interactie met de objecten mogelijk is. Een nadeel van deze keuze is het verplicht uitvoeren van Javascript. Op sommige computers wordt Javascript standaard geblokkeerd waardoor de site niet meer naar behoren werkt. Het Scadasysteem wordt bijvoorbeeld geheel gegenereerd door Javascript en jQuery.

9.4.5 Update interval

Voor het updaten van de grafieken en het Scadasysteem is een interval gekozen van één minuut. Een kleiner interval heeft geen zin omdat de database maar één keer per minuut wordt geüpdate. Een groter interval kan niet, omdat er een eis is die een minimum update van één minuut afdwingt. Bij een groter interval wordt het nut van een non-statische afbeelding weggenomen. Voor bepaalde gegevens op het scada systeem is gekozen voor een update interval van één dag. Het berekenen van deze gegevens duurt te lang om dit bij elke aanvraag te doen. Daarnaast veranderen deze gegevens niet veel aangezien het hier om de gemiddelde waarde over een lange periode gaat.

9.5 Mobiele applicatie

9.5.1 Viewpager

Voor het tonen van de grafieken in de applicatie wordt gebruik gemaakt van een viewpager. Hiervan is gebruik gemaakt omdat de grafieken afbeeldingen zijn, die vanaf de server worden geladen. Met de viewpager kan dan met een veeg over het scherm van afbeelding gewisseld worden.

9.5.2 Asynchroon afbeelding laden

De afbeeldingen van de grafieken worden asynchroon geladen. Dit is gekozen zodat de applicatie niet blokkeert bij het laden van een afbeelding. Een nadeel hierbij is dat de afbeelding nog steeds geladen wordt als de gebruiker al verder is gegaan naar de volgende afbeelding.

10 Conclusies en Aanbevelingen

In dit hoofdstuk worden de conclusies en aanbevelingen van dit project beschreven.

10.1 Conclusies

10.1.1 Doelstellingen

10.1.1.1 Warmteverdeling van de zonne-energie

Het hoofddoel van het systeem is om de warmte van de zonnecollector over de boilers te verdelen. Aan alle eisen die bij dit doel zijn gesteld is voldaan(zie 2.8 Eisen). Hieraan is voldaan door de software die op de PLC draait.

10.1.1.2 Warmtevoorziening van de boilers

Naast het verdelen van de zonne-energie moest bij een tekort de elektrische verwarming ingeschakeld worden. Dit is met een extra stuk code in de software op de PLC gerealiseerd.

10.1.1.3 Bedieningsinterface

Via een bedieningsinterface moest het mogelijk zijn om de bewoningsstatus en de gewenste temperatuur per woning in te stellen. De bewoningsstatus kan met een switch worden ingesteld. De gewenste temperatuur kan per woning op de LOGO! module worden ingesteld.

10.1.1.4 Monitoren van de installatie

De procesparameters moesten voor een periode van twee jaar worden opgeslagen. Binnen de beschikbare tijd is dit niet te bewijzen, doordat de afstudeerperiode daar te kort voor is. De gegevens gegenereerd ten tijde van de afstudeerperiode zijn wel bewaard gebleven. Hieruit mag men aannemen dat de software de gegevens voor twee jaar lang kan opslaan.

Daarnaast moest deze data via een webbrowser en op een lokaal display getoond kunnen worden. Dit is beide mogelijk. De gegevens op een lokaal display worden getoond op het display van de LOGO! module zelf. Deze biedt de mogelijkheid om de gegevens op het display te tonen.

10.1.2 Onderzoek

Aan het begin van het project is een vooronderzoek uitgevoerd. Uit dit onderzoek zijn veel belangrijke bevindingen voor het project naar voren gekomen. Hieronder valt bijvoorbeeld het communicatieprotocol met de PLC. De resultaten van dit onderzoek zijn tijdens het hele project gebruikt om de software te ontwerpen en te ontwikkelen.

Is de Raspberry PI geschikt als data-recorder voor een Siemens LOGO! PLC?

1. Wat zijn de interface mogelijkheden van de Raspberry PI?

   De Raspberry PI heeft veel communicatie mogelijkheden waaronder seriële communicatie via een usb poort.

2. Wat zijn de interface mogelijkheden van de Siemens LOGO!?

   De Siemens LOGO! heeft een aantal communicatie mogelijkheden waaronder seriële communicatie. Hierdoor kan de Siemens LOGO! communiceren met de Raspberry PI.

3. Hoe kan worden gecommuniceerd met de Siemens LOGO!?

   Het communicatieprotocol van de Siemens LOGO! ondersteunt het opvragen van data.

4. Wat voor informatie verstuurt de Siemens LOGO!?

   Uit het onderzoek is gebleken dat de Siemens LOGO! alle informatie van het programma kan versturen.

5. Wat is het data-format waarmee de Siemens LOGO! de data verstuurt?

   De Siemens LOGO! heeft drie formaten waarin het de data verstuurt. De resolutie van de gegevens is in deze formaten groot genoeg om ze nuttig te maken om te loggen.

6. Welke alternatieven zijn er voor de RRD(Round Robin Database)?

   Er zijn een aantal alternatieven voor de RRD, maar deze alternatieven hebben geen vast datagebruik wat voor een Raspberry PI met weinig opslagruimte zal zorgen dat de opslag snel vol raakt.

Door de beantwoording van bovenstaande vragen is gebleken dat de Raspberry PI gebruikt kan worden als data-recorder voor de Siemens LOGO! PLC. Daarnaast is dit ook uit het proof of concept gebleken.

10.1.3 Algemene conclusie

De resultaten van het totale project voldoen aan alle eisen die gesteld zijn(zie 2.8 Eisen). Hiermee zijn ook alle doelstellingen behaald. De software zou nu als een volledig systeem gebruikt kunnen worden op de locatie. Er zijn tijdens het project wel nog mogelijkheden ontdekt die tot een beter resultaat zouden kunnen leiden. Uiteindelijk wilde de opdrachtgever als extra functionaliteit ook de PLC aan kunnen sturen. Dit is niet gelukt met de huidige hardware. Dit punt is in hoofdstuk 10.2 Aanbevelingen behandeld.

10.2 Aanbevelingen

10.2.1 Hardware – PLC

Als PLC is tijdens dit project de Siemens LOGO! 0BA6 gebruikt. Het bedrijf had meerdere redenen om voor deze PLC te kiezen:

1. De PLC is zeer robuust.
2. Een PLC is speciaal ontwikkeld om een systeem te regelen.

Tijdens de uitvoering van het project zijn ook enkele nadelen aan deze PLC gevonden:

1. De PLC kan niet via de interface worden aangestuurd, waardoor bijvoorbeeld vanaf de PC de gewenste temperatuur niet kan worden aangepast.
2. De snelheid van de PLC zorgt ervoor dat bij grote programma's zoals bij dit project, het aantal metingen per seconde tamelijk laag komt te liggen. Dit is bij dit project geen probleem omdat het nog binnen de eisen valt. Mochten andere toepassingen bedacht worden dan zal dit zeker als een negatief punt kunnen gelden.

Om de waarden binnen het systeem toch te kunnen wijzigen kan een andere unit gekozen worden. Hierbij kan gedacht worden aan een nieuwere versie van de LOGO! zoals de 0BA7. Daarnaast kan gekeken worden naar custom hardware. Bij custom hardware is er ook nog het voordeel dat vaker per seconde gelogd kan worden. Dit vaker loggen is voor dit systeem niet van groot belang. Bij het gebruik van een ander systeem zou dit een groot verschil kunnen maken. Een bijkomend voordeel van andere hardware is het zelf kunnen bepalen van de interface, hierdoor kan verder geoptimaliseerd worden op de specifieke eisen van het systeem.

10.2.2 Database

Het is gebleken dat de database een groot knelpunt is voor de Raspberry PI. Een aanvraag aan de database vraagt al snel veel van de CPU, waardoor het lang kan duren voordat de site laadt. Er zou daarom gekeken kunnen worden naar het gebruik van een andere database, of naar de optimalisatie van de huidige database. Daarnaast kan worden gekeken naar een krachtiger processing unit, zoals de Beagleboard.

10.2.3 Website

Als van hardware wordt veranderd, kan een deel aan de website worden toegevoegd, waarmee de waarden binnen het systeem gewijzigd kunnen worden. Hierbij moet dan wel gedacht worden aan beveiliging met een wachtwoord zodat niet iedereen het systeem kan wijzigen.

11 Woordenlijst

Afkorting	Betekenis
DS	Data Source / Data Sources
LSB	Least significant byte
MSB	Most significant byte
PLC	Programmable logic controller
RRA	Round Robin Archive
RRD	Round Robin Database
SCADA	Supervisory control and data acquisition
SSH	Secure Shell

AT-STYLE

AT-STYLE is een tijd specificatie die genoemd is naar het Unix commando at(1). Hierbij kan op vrij complexe manier de tijd gespecificeerd worden. Deze specificatie bestaat uit twee delen: de tijd referentie en de tijd offset (Oetiker, n.d.). Bijvoorbeeld: “now-2hours”, hierbij is “now” de tijd referentie en “-2hours” de tijd offset.

12 Bronvermelding

CherryPy Team. (2014). *Deploy – Reverse-proxying.* Geraadpleegd op http://cherrypy.readthedocs.org/en/latest/deploy.html#reverse-proxying

Department of Defence, United States of America. (1995). *J-STD-016 Standard for Software Development and Documentation*

Oetiker, T. (30-09-2014). *rrdtool.* Geraadpleegd op http://oss.oetiker.ch/rrdtool/doc/rrdtool.en.html

Oetiker, T. (n.d.). *RRDtool – rrdfetch AT-STYLE TIME SPECIFICATION*. Geraadpleegd op http://oss.oetiker.ch/rrdtool/doc/rrdfetch.en.html

Over ons. n.d. Geraadpleegd op 22 juli 2014 via http://www.triopsys.nl/over-ons/

Ronacher, A. (2008). Jinja2. Geraadpleegd op http://jinja.pocoo.org/docs/dev/

Bijlagen

Bijlage A – Plan van Aanpak

Bijlage B – Evaluatie

Bijlage C – Onderzoeksdocumenten

Bijlage C1 – Vooronderzoek

Bijlage C2 – Onderzoek non-lineaire sensoren

Bijlage D – Lijsten van objecten

Bijlage D1 – Lijst van figuren

Lijst van figuren

Bijlage D2 – Lijst van grafieken

Lijst van grafieken

Bijlage D3 – Lijst van code

Lijst van code

Bijlage D4 – Lijst van tabellen

Lijst van tabellen

Bijlage D5 – Lijst van flowcharts

Lijst van flowcharts