

2016

Klimaatkast besturing

Auteur:
Danjel L. Keekstra
Datum:
22-01-2016

Titelpagina

Titel: Klimaatkast besturing
Document: Afstudeerverslag
Locatie: Brunelco, Haaksbergen, Nederland
Datum: 22-01-2016
Revisie: 1.0

Organisatie: Saxion University of Applied Sciences
Begeleider: J.S.D. Stokkink

Opdrachtgever: Brunelco Electronic Innovators
Begeleider: N. Voogsgeerd

Uitvoerder: D. L. Keekstra

Voorwoord

Het verslag dat voor u ligt is de afsluiting van een vijf maanden durende afstudeerperiode bij het bedrijf Brunelco. Dit verslag is gemaakt voor mijn studie elektrotechniek aan het Saxion te Enschede.

In deze periode heb ik mijn opgebouwde kennis uit opleiding en voorgaande opleiding kunnen laten zien. Ook heb ik in deze periode nieuwe kennis verschaft. Dit heeft mij zeer geïnspireerd om mij verder te verdiepen in de electronica. Ik heb genoten van deze periode en zal de ervaringen meenemen in mijn verdere carrière.

Graag wil ik een bijzondere dank richten aan Niek Voogsgeerd en Hans Stokkink die mij tijdens mijn afstudeerperiode hebben begeleid. Tevens wil ik het bedrijf Brunelco en hogeschool Saxion bedanken voor het mogelijk maken van mijn afstudeerstage. Verder wil ik mijn moeder, mijn vriendin en de bewoners van L'estate of happiness voor hun steun bedanken gedurende deze periode.

Ik wens u veel leesplezier.

Danjal Lieuwe Keekstra

Haaksbergen, Januari 2016

Samenvatting (Engels)

Most people experienced it, a device that is hard to operate. The user interface often causes the most problems. The user interface can be seen as the direct communication between the device and the operator. In order to have a effective device or system the user interface is of great importance.

In the test department of Brunelco, various designs of printed circuit boards are being tested. These tests are based on requirements set by the clients or the law. For example, one of the requirements might be handling the working temperature. In order to test whether the product is capable of working within the required temperature range, a temperature controlled cabinet is used. Unfortunately, this cabinet is difficult to handle and not very user friendly.

The main objective of my assignment was to create a more user friendly interface. Another goal was to upgrade the performance of the product. This will be done by adding the functions of data logging, temperature sensing, remote monitoring and controlling.

In this assignment the V-model is used, this model ensures efficient communication between the different stakeholders. Furthermore, the model will reflect on every phase of the product to ensure the quality.

The result is an interface for the cabinet that can be controlled over Ethernet. This interface uses a software application that is characterized by a more user friendly graphical interface. To conclude the developed product is a good solution to make the user interface more convenient to use.

However, the product can still be improved. In order to connect with the new interface, you still have to type in an address manually in order to run the application. An 'auto search' function might improve this and could make the application more user friendly. Furthermore, the interface does not accept all temperature sensors. A solution might be to replace the sensor circuitry in order to let the application accept more types of temperature sensors.

Inhoud

Titelpagina -----	2
Voorwoord -----	3
Samenvatting (Engels) -----	4
Inhoud -----	5
Afkortingen -----	8
Figuren-----	9
Tabellen -----	10
1 Inleiding-----	11
1.1 Achtergrondinformatie -----	11
1.2 Probleemstelling -----	11
1.2.1 Probleemstellingen-----	11
1.2.2 Doelstellingen -----	11
1.3 Project grenzen -----	12
1.4 Aanpak en methodologie -----	13
1.5 Verslagopbouw -----	14
2 Projectoverzicht-----	15
2.1 Gebruikers vereisten -----	15
2.2 Functionele vereisten -----	15
2.3 Technische eisen -----	17
2.4 Kosten raming-----	18
2.5 Schematisch overzicht-----	18
3 Huidige situatie-----	19
3.1 Klimaatkast besturing-----	19
3.2 RS232 Interface-----	19
4 Concept onderzoek -----	21
4.1 Basisopstelling-----	21
4.2 Concept keuzes -----	22
4.3 Communicatie-----	22
4.4 Grafische gebruikersinterface -----	23
4.5 Processor-----	24
4.6 Ontwikkelingssoftware -----	25
4.7 Concept Keuze -----	26
5 Functioneel ontwerp -----	27

5.1	Overzicht functioneel ontwerp -----	27
5.2	Mechanisch functioneel ontwerp -----	28
5.3	Elektronisch functioneel ontwerp -----	29
5.4	Elektrisch functioneel ontwerp -----	33
5.5	Software functioneel ontwerp -----	34
6	Interface hardware ontwerp -----	36
6.1	Voeding-----	36
6.2	Digitale ingangen-----	38
6.3	Digitale uitgangen-----	38
6.4	Temperatuur meting -----	39
6.5	RS232module -----	41
6.6	Processor module -----	42
7	Interface software ontwerp-----	44
7.1	Basis werking -----	44
7.2	Real-time Besturingssysteem (RTOS)-----	44
7.3	Ethernet communicatie -----	45
7.4	RS232 communicatie -----	46
7.5	Temperatuur processing-----	46
7.6	Gegevens logging -----	48
8	Computersoftware Ontwerp-----	49
8.1	Grafische interface -----	49
8.2	Alarm meldingen -----	50
8.3	Ethernet communicatie -----	50
9	Controle -----	51
9.1	FMEA-----	51
9.2	FAT -----	51
9.3	SAT -----	51
10	Conclusies en aanbevelingen -----	52
10.1	Conclusies-----	52
10.2	Aanbevelingen -----	52
	Referenties-----	53
	Bijlage 1: Planning-----	54
	Bijlage 2: Print ontwerp-----	57
5.	Component kosten calculatie-----	64

Bijlage 3: Protocol beschrijving -----	65
Bijlage 4: Testresultaten -----	73
Bijlage 5: Datablad klimaatkast-----	85
Bijlage 6: Logboek -----	90
Bijlage 7: Interface Software -----	96
Bijlage 8. Aplicatie software -----	133

Afkortingen

AC	Metal-Oxide-Semiconductor Field-Effect Transistor	38
Alternating Current.....		
CE	Negative Temperature Coefficient...	39
Conformité Européenne.....		
CR	PC	52
Carriage Return	Personal Computer	
DC	PCB	13
Direct Current	Printed Circuit Board	
DHCP	PHY	45
Dynamic Host Configuration Protocol	Physical Layer	
EMC	RAM	43
ElectroMagnetic Compatibility.....	Read Acces Memory.....	
FAT	RF	22
Factory Acceptance Test.....	Radio Frequentie	
FMEA	RMII	45
Failure Mode and Effects Analysis ...	Reduced Media-Independent Interface	
IP	RTOS	44
Internet Protocol.....	Real-Time Operating System.....	
LED	SAT	14
Light-Emitting Diode	Site Acceptance Test.....	
LWIP	TVS	37
Licht weigth IP	Transient-Voltage-Suppression	
mA	UART	41, 46
miliAmpere.....	Universal Asynchronous Receiver/Transmitter	
MAC	UDP	52
Media Acces Control	User Datagram Protocol.....	
MOSFET		

Figuren

Figuur 1.4.1: Het V-model	13
Figuur 2.2.1: Blackbox observatie gebruikersinterface.....	16
Figuur 2.5.1: Basis werking	19
Figuur 3.1.1: Instelmodule VT4002	19
Figuur 4.1.1: Basis werking	21
Figuur 4.1.2: Overzicht functionele functies	21
Figuur 5.1.1: Overzicht functionele functies	27
Figuur 5.2.1: Mechanische schets.....	28
Figuur 5.2.1: Blokschema Voedingsmodule.....	29
Figuur 5.3.2: loggingsmodule	29
Figuur 5.3.3: RS232 module	29
Figuur 5.3.4: Ethernet module	30
Figuur 5.3.5: Differentiaal signaal.....	30
Figuur 5.3.6: Debug modulen.....	30
Figuur 5.3.7: Digitale ingangsmodule.....	30
Figuur 5.3.8: Digitale uitgangsmodule	31
Figuur 5.3.9: NTC Module.....	31
Figuur 5.3.10: Processor module	31
Figuur 5.3.11: Totaal overzicht interface	32
Figuur 5.4.1: Elektrisch ontwerp.....	Fout! Bladwijzer niet gedefinieerd.
Figuur 5.5.1: Interface software diagram.....	34
Figuur 5.5.2: PC software diagram	35
Figuur 6.1.1: Schema voeding.....	37
Figuur 6.1.2: PCB ontwerp voeding.....	37
Figuur 6.2.1: Schema digitale ingangen	38
Figuur 6.2.2: Berekening vermogen weerstand	38
Figuur 6.3.1: Schema digitale uitgang.....	39
Figuur 6.4.1: Schema temperatuur inlezing.....	40
Figuur 6.5.1: RS232 niveaus.....	41
Figuur 6.5.2: Schema RS232 conversie	42
Figuur 6.6.1: Schema processorplatform	43
Figuur 7.2.1: Voorbeeld RTOS taken	44
Figuur 7.3.1: Ethernet routine	45
Figuur 7.3.2: LWIP software overzicht.....	45
Figuur 7.4.1: UART Commando routine.....	46
Figuur 7.5.1: Timing MAX6691	46
Figuur 7.5.2: Schema Timer 4	47
Figuur 7.5.3: Vervangingschema weerstandsmeting	47
Figuur 8.1.1: Computer software	49
Figuur 8.2.1: Alarm melding.....	50

Tabellen

Tabel 1.3.1: Mijlpalen van het project	12
Tabel 1.3.2: Grenzen van het project	12
Tabel 2.1.1: Vereisten van Brunelco aan de gebruikersinterface	15
Tabel 2.2.1: Functionele vereisten.....	17
Tabel 2.3.1: Technische vereisten.....	17
Tabel 2.4.1: Kosten raming	18
Tabel 3.2.1: Protocol klimaatkast.....	19
Tabel 4.2.1: Concept keuzes	22
Tabel 4.3.1: Vergelijkingstabel communicatie	23
Tabel 4.4.1: Vergelijkingstabel grafische interface	24
Tabel 4.5.1: Vergelijkingstabel Processor.....	25
Tabel 4.6.1: Vergelijkingstabel Ontwikkelformware	25
Tabel 4.7.1: Concept keuze	26
Tabel 5.1.1: Legenda kleur betekenis	27
Tabel 5.3.1: Legenda elektronisch ontwerp.....	33
Tabel 5.3.2: Aansluitingen interface.....	33
Tabel 5.4.1: Beschrijving in en uitgangen	33
Tabel 6.1.1: Schatting maximum stroomverbruik 5 Volt DC voeding.....	36
Tabel 6.1.2: Keuze voeding	36
Tabel 6.4.1: Keuze NTC meting	39
Tabel 6.6.1: Keuze processor	42
Tabel 6.6.2: Keuze processorplatform	43
Tabel 7.6.1: Geheugen indeling	48
Tabel 7.6.2: Pakket indeling	48

1 Inleiding

1.1 Achtergrondinformatie

Brunelco is een bedrijf gelegen in Haaksbergen. Het bedrijf maakt klant gespecificeerde producten met de focus op elektronica en software. Het is een relatief klein bedrijf met ongeveer 20 medewerkers. Om de producten te testen op gestelde eisen is er een test afdeling. Voor producten wordt een eis gesteld dat ze een minimum en maximumtemperatuur kunnen doorstaan. Om dit te testen wordt een klimaatkast gebruikt. Dit is een kast waarin de temperatuur kan worden geregeld van -40 tot +130 graden. In deze kast kan een werkend product worden geplaatst om zo een temperatuur test te doen. Deze kast bevat besturing die niet gebruiksvriendelijk wordt ervaren. Ook missen er functies voor het opslaan van temperatuur gegevens. Tijdens dit project is het de bedoeling om deze besturing gebruiksvriendelijker te maken en extra functies toe te voegen. De klimaatkast wordt gebruikt door de medewerkers van Brunelco en de opdracht is dus ook uitgevaardigd door Brunelco.

1.2 Probleemstelling

1.2.1 Probleemstellingen

Het project om een nieuwe interface te maken voor de klimaatkast komt voort uit de problemen die zich voordeden bij het instellen van de klimaatkast. De gebruikersinterface werd ervaren als moeilijk instelbaar. Ook waren niet alle gewenste functies aanwezig.

1.2.2 Doelstellingen

De doelstelling van dit project is het maken van een gebruiksvriendelijke interface voor de klimaatkast. Met een gebruiksvriendelijke interface wordt bedoeld dat de klimaatkast in te stellen is door medewerkers van Brunelco met een korte verdieping van maximaal 5 minuten in de documentatie te hebben gedaan. Naast de gebruikersinterface zijn er een aantal subdoelstellingen om de functionaliteit te verhogen:

- Mogelijkheid om extra temperatuur sensoren aan te sluiten.
- Het Loggen van temperaturen.
- Aanbieden van uitgangen die worden geactiveerd door in te stellen temperatuurveranderingen.
- Het kunnen maken van temperatuurstestprofielen.
- Digitale ingangen voor status weergave van statussen van het te meten product.

1.3 Project grenzen

Het project start op 1 september en zal duren tot 27 januari. Op 27 januari zal er een presentatie plaatsvinden op het Saxion. Hierin wordt het resultaat van het project en de weg hier naartoe gepresenteerd. Mocht het project uitlopen is hier mogelijkheid voor. Dit zal betekenen dat de presentatie verplaatst naar de eerstvolgende presentatie datum van het Saxion. Binnen het project zijn een aantal mijlpalen die in Tabel 1.3.1 in een tabel form zijn opgeschreven.

Tabel 1.3.1: Mijlpalen van het project

(Uiterlijke) Datum	Beschrijving
01-09-2015	Start Afstuderen
15-09-2015	Inleveren projectplan
18-11-2015	Inleveren ontwerp verslag
16-12-2015	Inleveren concept verslag
22-01-2016	Inleveren verslag
22-01-2016	Inleveren reflectie verslag
28-01-2016	Inleveren beoordelingsformulier
28-01-2016	Inleveren presentatie
29-01-2016	Einde afstudeerstage

Het project bestaat uit het maken van een userinterface voor een bestaande klimaatkast. In Tabel 1.3.2 is samengevat wat wel en niet binnen de grenzen van het project valt.

Tabel 1.3.2: Grenzen van het project

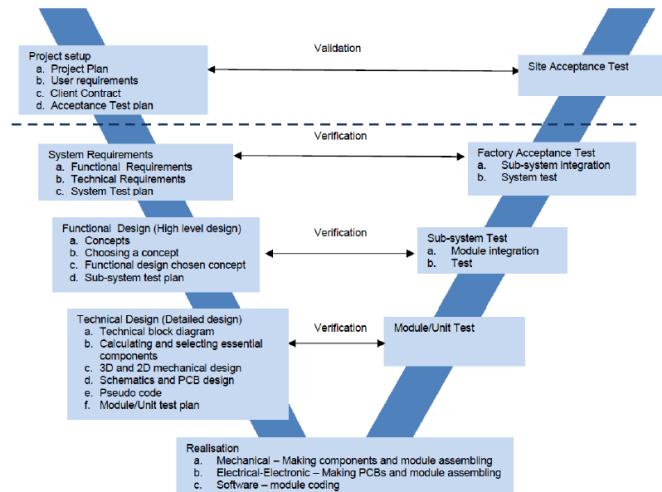
Valt onder het project	Valt niet onder het project
Het maken van een grafische interface die makkelijk te gebruiken is.	Veranderingen aanbrengen in de klimaat kast.
Het maken van een mogelijkheid om temperatuur sensoren aan te sluiten op de interface.	Het ontwerpen van een behuizing voor de interface
Het maken van digitale inputs om signalen te monitoren.	Het zorgen voor een norm kalibratie.
Het maken van software om temperatuur en tijd profielen te kunnen maken en gebruiken	
Het maken van logging van de temperaturen.	
Het maken van een aansturing op afstand.	

Het project zal succesvol zijn als er een gebruiksvriendelijke interface is gerealiseerd. Hierbij zijn de bijbehorende extra functie eisen ook behaald.

1.4 Aanpak en methodologie

Tijdens het project wordt gewerkt met het V-model. Het V-model is een manier van werken waarbij er in fasen wordt gewerkt naar het realiseren van het eindproduct. Na het realiseren zal dan in fasen weer worden gereflecteerd op het gerealiseerde product. In Figuur 1.4.1 is het V-model te zien. Dit model is gekozen om de productiviteit te verbeteren en het zorgen voor een goede communicatie tussen de betrokken personen. Het V-model bestaat uit negen basisactiviteiten. Onder deze basisactiviteiten vallen sub activiteiten waarin de taken beter zijn omschreven. In de lijst hieronder staan de activiteiten voor dit project. Een van de sub-activiteiten zijn reviews. Bij een review kijkt een collega over het gemaakte werk om te controleren op fouten.

1. Projectvoorbereiding
 - a) Gebruikers vereisten opstellen
 - b) Kosten raming maken
 - c) Projectplan schrijven
 - d) Projectplan review
 - e) Projectplan verbeteren
 - f) Project management (Overleg, planning, Rapportage en verslaglegging)
2. Systeem vereisten opstellen
 - a) Functionele vereisten
 - b) Technische vereisten
 - c) Vereisten reviewen
 - d) Test plan opstellen
 - e) Vereisten verbeteren
 - f) Project management (Overleg, planning, Rapportage en verslaglegging)
3. Functioneel ontwerp
 - a) Concept onderzoek
 - b) Concept keuze maken
 - c) Belangrijke componenten selectie
 - d) Ontwerp hardware architectuur
 - e) Ontwerp software architectuur
 - f) Review soft- en hardware architectuur ontwerpen
 - g) Subsysteem testplan schrijven
 - h) Verbetering soft- en hardware architectuur ontwerpen
 - i) Project management (Overleg, planning, Rapportage en verslaglegging)
4. Technisch ontwerp
 - a) Simulatie en onderzoek
 - b) Interface omschrijving uitwerken
 - c) Review interface omschrijving
 - d) Software ontwerp
 - Flowdiagrammen
 - e) Hardware ontwerp
 - Schema
 - PCB



Figuur 1.4.1: Het V-model

- f) FMEA en veiligheidsanalyse uitvoeren
 - g) Wijzigingen toepassen na FMEA
 - h) Review soft- en hardware ontwerpen
 - i) Verbeteren soft- en hardware ontwerpen
 - j) Componentenlijst opstellen
 - k) Module testplan ontwerpen
 - l) Project management (Overleg, planning, Rapportage en verslaglegging)
5. Realisatie
- a) Elektronica assembleren
 - b) Software realisatie
 - c) Mechanisch assembleren
 - d) Project management (Overleg, planning, Rapportage en verslaglegging)
6. Module test
- a) Module werkingstest
 - b) Project management (Overleg(intern/extern), Rapportage)
7. Subsystemen test
- a) Module integratie
 - b) Project management (Overleg(intern/extern), Rapportage)
8. Interne acceptatie test (FAT)
- a) Sub systeem integratie
 - b) Systeem test
 - c) Pre-compliance testen (CE, EMC en temperatuur)
 - d) Project management (Overleg, planning, Rapportage en verslaglegging)
9. Externe acceptatie test (SAT)
- a) Praktijk test
 - b) Project management (Overleg(intern/extern), Rapportage)

1.5 Verslagopbouw

Dit verslag zal beginnen met de vereisten die werden gesteld aan het project. Hierna zal informatie worden gegeven over de bestaande situatie. Er wordt dan gekeken na de mogelijke oplossingen. Deze oplossingen worden vergeleken en hier wordt een conclusie mee gevormd. Hieruit zal voor een concept worden gekozen om de interface te ontwerpen. Tijdens het ontwerpen wordt er onderscheid worden gemaakt tussen interface hardware, interface software en computersoftware. Deze onderwerpen zullen per onderdeel worden besproken. Hierna volgt een beschrijving van de controles die op het project zijn gedaan. Als laatste zal er een conclusie aan het project worden gehangen met daarbij horende aanbevelingen. Hierop volgen bijlagen ter ondersteuning van het document.

2 Projectoverzicht

In dit hoofdstuk wordt bekeken welke eisen er aan het product worden gesteld. Deze eisen komen van de klant en worden aangevuld door analyse en onderzoek. Hierbij komen zowel gebruikers, functionele en technische eisen aan bod.

2.1 Gebruikers vereisten

Het doel van dit project is het maken van de gebruiksvriendelijke interface voor de klimaatkast. Met een gebruiksvriendelijke interface wordt bedoeld dat de klimaatkast in te stellen door medewerkers van Brunelco met een korte verdieping in de documentatie te hebben gedaan. Naast de gebruikersinterface zijn er een aantal subdoelen om de functionaliteit te verhogen:

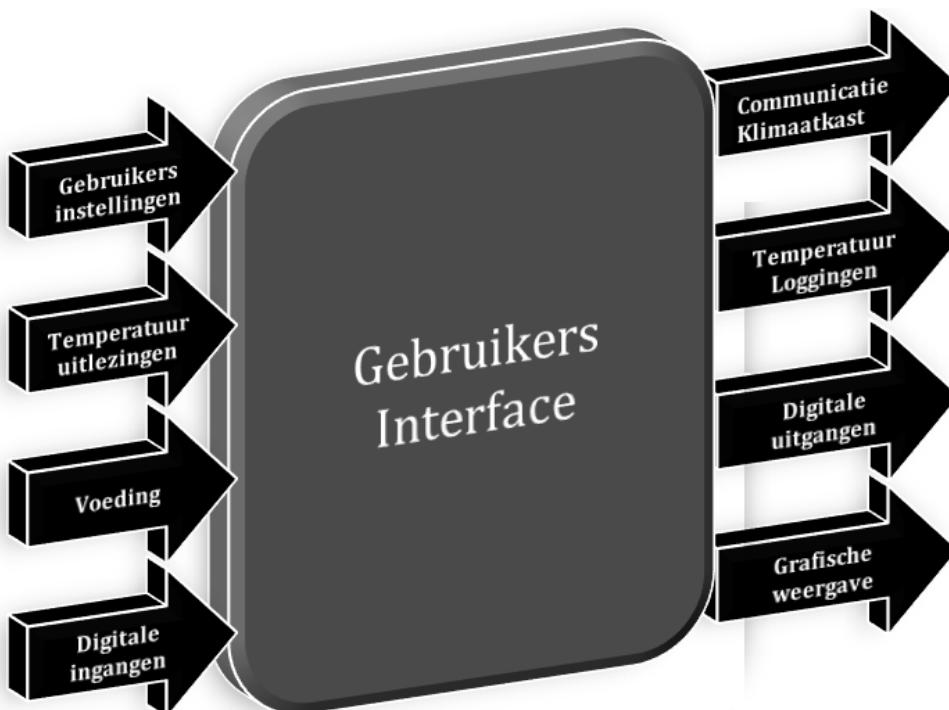
- Mogelijkheid om extra temperatuur sensoren aan te sluiten
- Loggen van temperaturen
- Het gebruikmaken digitale uitgangen als noodstop bij overschrijding van grenswaarden
- Het maken van temperatuurtestprofielen
- Digitale inputs voor terugkoppeling van het product.

Tabel 2.1.1: Vereisten van Brunelco aan de gebruikersinterface

Nummer	Vereisten
GV0101	Gebruiksvriendelijke interface
GV0102	Externe temperatuur sensoren
GV0103	Loggen van temperatuur data
GV0104	Digitale uitgangen
GV0105	Digitale ingangen
GV0106	Mogelijkheid om temperatuur profielen te gebruiken

2.2 Functionele vereisten

Om een gebruikersinterface te creëren zijn een aantal functionele eisen benodigd. Deze functionele eisen zijn door analyse van de vereisten van de klant en voor onderzoek bepaald. Hierbij is gebruikgemaakt van het back box principe. Daarbij wordt het systeem als een onbekende doos bekeken en worden de invoer en uitvoer geanalyseerd. Dit is te zien in Figuur 2.2.1:Blackbox observatie gebruikersinterface. Na de black box observatie kunnen de functionele vereisten worden opgesteld. Deze functionele eigenschappen zijn vervolgens in overleg met de klant vast gelegd. Deze zijn te vinden in Tabel 2.2.1: Functionele vereisten.



Figuur 2.2.1:Blackbox observatie gebruikersinterface

De in en uitgangen van de gebruikersinterface bestaan uit de volgende eigenschappen.

- De gebruikers instellingen zijn de instellingen die de gebruiker aan het systeem geeft.
- De temperatuur uitlezing is de temperaturen die het systeem uit de klimaatkast leest.
- De voeding zorgt voor de energie levering aan de gebruikers interface
- De digitale ingangen zorgen voor een aansluiting voor de gebruiker om een één of nul door te geven aan het systeem.
- De communicatie naar de klimaatkast zorgt voor de aansturing en uitlezing van de klimaat kast.
- De temperatuur logging zorgt voor het opslaan en het weder oproepen van de gegevens die worden verzameld tijdens het proces.
- De digitale uitgangen zijn voor de elektrische terugkoppeling naar de gebruiker aan te sluiten apperatuur.
- De grafische weergave is te visuele terugkoppeling van de gebruiker. Hierop wordt de gebruiker geïnformeerd over het proces en krijgt het visuele terugkoppelingen op de acties die zijn gedaan.

Door deze eigenschappen te gebruiken is het nu mogelijk om de functionele vereisten op te stellen. Deze zijn te vinden in Tabel 2.2.1: Functionele vereisten.

Tabel 2.2.1: Functionele vereisten

Nummer	Verreisden	Relatie	Waarde	Eenheid
FV0101	4 aansluitmogelijkheden voor een NTC temperatuur sensor	=	Waar	-
FV0102	4 Monitorbare digitale ingangen	=	Waar	-
FV0103	4 op temperatuur limit geschakelde uitgangen	=	Waar	-
FV0201	Aansluiting voor een net adapter	=	Waar	-
FV0301	Verbinding op afstand met de interface	=	Waar	-
FV0302	Grafische interface Bereikbaar binnen het kantoor van Brunelco.	=	Waar	-
FV0401	Aansturing klimaatkast	=	Waar	-
FV0501	Keuze tussen handmatig en profiel modus	=	Waar	-
FV0502	Configureerbare temperatuur profielen	=	Waar	-
FV0503	Per sensor instelbare limiet waarde	=	Waar	-
FV0504	Alarm melding bij kritieke (instelbare) temperaturen	=	Waar	-
FV0601	Bij stroomuitval moet de loggingsdata bewaard blijven	=	Waar	-
FV0602	Exporteer mogelijkheid voor loggingsdata	=	Waar	-
FV0603	Onder de logging valt per punt: setpoint, sensor temperaturen, klimaatkast temperatuur, status digitale ingangen, statusdigitale uitgangen en een tijdsstempel.	=	Waar	-
FV0701	Grafische weergave van data in grafiek form	=	Waar	-
FV0702	Grafische weergave van actuele status, grenswaarden en tempraturen	=	Waar	-
FV0702	Keuze voor wel of niet weergeven van waardes in grafische weergave	=	Waar	-

Er is hierbij gekozen om de digitale en temperatuur ingangen en digitale uitgangen in viervoud uit te voeren. Deze maat is gekozen om ervoor te zorgen dat er genoeg mogelijkheden zijn en de ruimte die in de klimaatkast beschikbaar is. Ook moet de grafische interface op afstand beschikbaar zijn dit om ervoor te zorgen dat de het gebruiksgemak en daar door de gebruikersinterface goed is.

2.3 Technische eisen

De technische vereisten zijn te vinden in Tabel 2.3.1: Technische vereisten.

Tabel 2.3.1: Technische vereisten

Nummer	Verreisden	Relatie	Waarde	Eenheid
TV0101	Voeding spanning DC	>..<	11 .. 13	V
TV0102	Maximaal AC component	<	200	mV
TV0103	Maximum DC stroom opname	<	1.5	A
TV0104	Zekering op de voeding	=	Waar	-
TV0201	Communicatie bus naar klimaatkast	=	RS232	-
TV0401	Minimale omgevingstemperatuur in werking	>=	0	°C
TV0402	Maximale omgevingstemperatuur in werking	<=	50	°C
TV0403	Maximale operatie luchtvuchtigheid	<	80	%
TV0501	Minimale opslag temperatuur	>	-20	°C

TV0502	Maximale opslag temperatuur	<	90	°C
TV0503	Maximale opslag luchtvochtigheid	<	80	%
TV0601	Protectiegraad behuizing	=	IP20	-
TV0701	Data verversingssnelheid	=	1	Hz
TV0702	Grafische verversingsnelheid (instelbaar)	>..<	1..0.06	Hz
TV0703	Resolutie temperatuur meting	<=	0.1	°C
TV0704	Meetbereik temperatuur sensoren	>..<	-40..130	°C
TV0705	Nauwkeurigheid meting volledige bereik	<=	+/-2	°C
TV0706	Aan te sluiten temperatuursensor NTC	=	10000	Ohm
TV0801	Logische 0 digitale ingangen	>..<	0 .. 0.3	V
TV0802	Logische 1 digitale ingangen	>..<	2.8 .. 30	V
TV0901	Maximale uitgangsrelais spanning AC	<	250	V
TV0902	Maximale uitgangsrelais stroom AC	<	8	A
TV0903	Maximale uitgangsrelais spanning DC	<	30	V
TV0904	Maximale uitgangsrelais stroom DC	<	8	A
TV1001	Voldoet aan de CE eisen voor industrieel gebruik	=	Waar	-

2.4 Kosten raming

Om te kijken of de kosten binnen de grenzen van de klant blijven, is er een kosten raming gemaakt. Deze kostenraming bevat een schatting van de benodigde kosten om het project succesvol te laten verlopen. Hierbij is rekening gehouden met de te gebruiken software, de kosten van de uren van zowel begeleiding als stage, gebouw en gereedschapskosten.

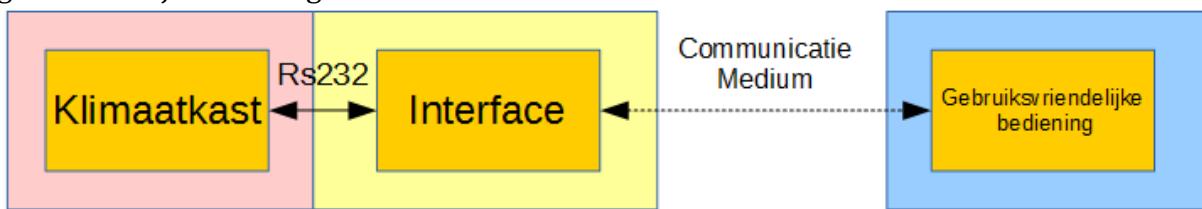
Tabel 2.4.1: Kosten raming

Kosten	Beschrijving	Aantal	Beschrijving calculatie	Kosten
Personeel	Uren stagiaire	0,4 jaar	504 euro stagevergoeding per maand	€2016
	Uren stagebegeleiding	0,4 jaar	100 euro maal 2 uur maal 20 weken	€4000
Machines	Afschrijving laptop	0,4 jaar	700 euro 90% afschrijving in 3 jaar	€84
	Afschrijving display	0,4 jaar	200 euro 90% afschrijving in 3 jaar	€24
	Schatting Afschrijving gereedschap	0,4 jaar		€50
Software licenties	Windows, office	0,4 jaar	200 euro afschrijving 90% 3 jaar	€24
	Altium	0,4 jaar	3200 euro afschrijving 90% 3 jaar	€384
Gebouw	Gebouwkosten 6m2	0,4 jaar	Uitgaande van 150euro per m2 per maand	€360
Materiaal	Geschatte Component kosten			€200
	Geschatte PCB ontwikkeling		Schatting van 2 prototypes	€150
Totale kosten				€ 7.292,00

2.5 Schematisch overzicht

Uit de gemaakte functionele en technische vereisten kan nu een basis werking schema opstellen. In Figuur 2.5.1: Basis werking is te zien dat er 3 componenten zijn. Hiervan is de klimaat kast het bestaande component. De communicatietussen de interface en Klimaatkast wordt gedaan over Rs232. Dit is bepaald door de klimaat kast omdat deze over dit medium communiceert. De interface zal dan communiceren via een te kiezen medium met de eis dat het bereikbaar is binnen de kantoren van Brunelco. Deze

communicatie communiceert dan met een gebruiksvriendelijke bediening waarop de gebruiker zijn instellingen kan doen.



Figuur 2.5.1: Basis werking

3 Huidige situatie

In dit hoofdstuk wordt de huidige functionele en technische werking van de klimaatkast geschetst. Hierbij wordt tevens de werking van het protocol dat over de RS-232 communicatie wordt gevoerd verteld.

3.1 Klimaatkast besturing

De klimaat kast is geproduceerd door de firma Vötsch. De klimaatkast heeft het type nummer VT4002. In de bijlage is de gebruikershandleiding te vinden. De huidige gebruikersinterface bestaat uit een klein display met een aantal knoppen. Deze is te zien in Figuur 3.1.1: Instelmodule VT4002. Om de klimaat kast te starten is het nodig om eerst de hoofdschakelaar om te zetten. Vervolgens zal de machine opstarten en is het mogelijk om het wachtwoord in te toetsen met de knoppen. Als de juiste toetsencombinatie is bevestigd zal de machine gereed zijn om in te stellen. Om vervolgens de temperatuur in te stellen is het nodig om door het menu heen te gaan met de pijl toetsen. Hierna kan de machine gestart worden. Als tijdens het draaien de temperatuur moet worden veranderd kan dit doormiddel van hetzelfde principe als het instellen als de kast nog niet draait. Om de kast vervolgens uit te zetten moet in het menu het start item worden geselecteerd en bevestigt. Vervolgens kan met de knoppen + en - de keuze off worden geselecteerd na de bevestiging hiervan stopt de kast met draaien. Hierna kan de hoofdschakelaar om worden gezet.



Figuur 3.1.1: Instelmodule VT4002

Het kleine display en de vele handelingen die er moeten worden gedaan om het product te gebruiken maakt het een niet gebruiksvriendelijk product.

3.2 RS232 Interface

De klimaatkast heeft een communicatieprotocol over een RS-232 verbinding. Dit protocol bestaat uit een aantal functies te vinden in Tabel 3.2.1: Protocol klimaatkast.

Tabel 3.2.1: Protocol klimaatkast

Start	Adres	Commando	Argument	Eind	Functie
\$	00	I	Geen	CR	Uitlezen van de huidige staat van de klimaatkast. (Temperatuur, Temperatuurstijging, Start)
\$	00	F	Geen	CR	Lees error status.

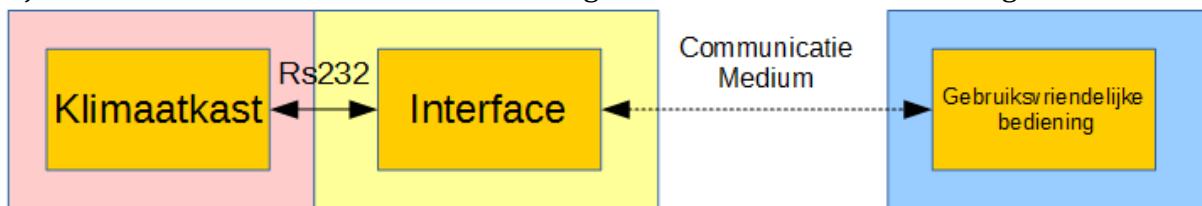
\$	00	Q	Geen	CR	Reset error status.
\$	00	E	Temperatuur Klimaatkast	CR	Het instellen van de temperatuur.
\$	00	U	Temperatuurstijging Klimaatkast	CR	Het instellen van de temperatuurstijging.
\$	00	P	Aan of uit.	CR	Het starten of stoppen van de klimaat kast.

4 Concept onderzoek

Voor het ontwerpen van het product wordt er eerst een onderzoek gedaan. In dit onderzoek wordt gekeken welke oplossingen en opties er zijn binnen de gestelde eisen. Om de afwegingen te vereenvoudigen wordt het systeem opgesplitst in subsystemen. Voor ieder subsystem wordt een onderbouwde keuze gemaakt tussen de mogelijke opties. Bij de keuze worden verschillende parameters gebruikt waaronder implementatie tijd en kosten. Dit om de juiste oplossing voor het project te kiezen.

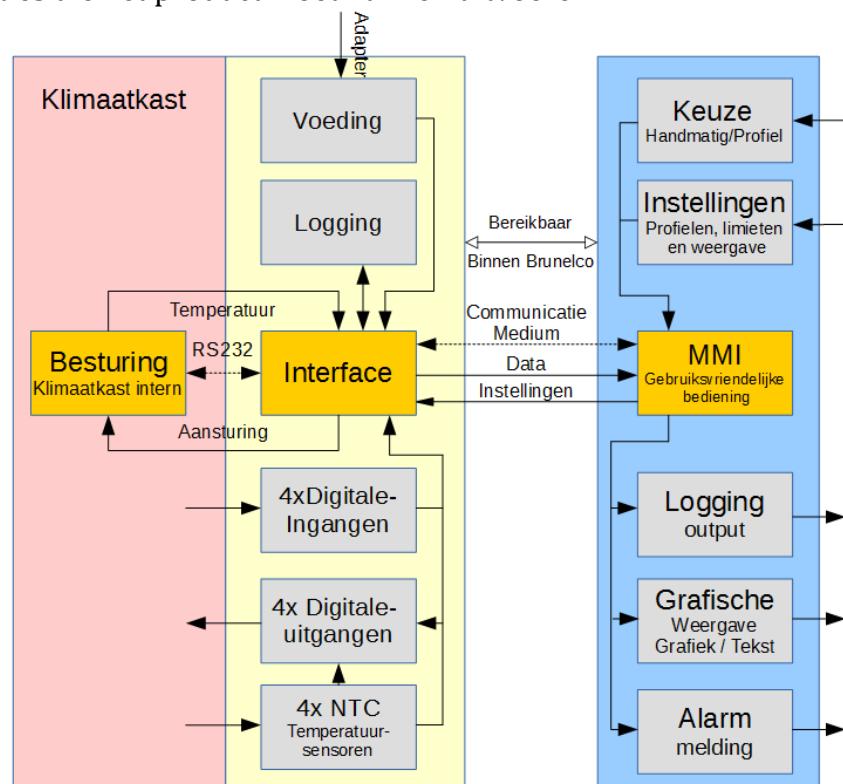
4.1 Basisopstelling

De basisopstelling die door de klant aangeleverd bestaat uit een klimaatkast met RS232 aansluiting. Om de kast aan te sturen is een processor nodig. Deze processor is nodig om de communicatie te verzorgen met de klimaatkast. De processor zal dan nog een communicatie moeten maken naar de gebruiksvriendelijke interface. De gebruiksvriendelijke interface zal volgens de functionele vereisten bereikbaar moeten zijn binnen het kantoor van Brunelco. In Figuur 4.1.1: Basis is de situatie geschetst.



Figuur 4.1.1: Basis werking

Om ook aan de verder gestelde eisen te voldoen moet de interface een aantal extra functies uitvoeren. In Figuur 4.1.2: Overzicht functionele functies het totale overzicht van alle functies die het product moet kunnen uitvoeren.

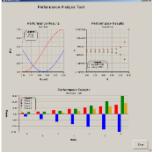
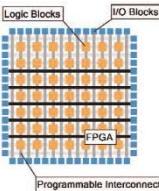


Figuur 4.1.2: Overzicht functionele functies

4.2 Concept keuzes

Uit onderzoek zijn een aantal keuzemogelijkheden gekomen. Deze keuzes worden afgewogen om zo tot een concept ontwerp te komen. In Tabel 4.2.1: Concept keuzes staan de te overwegen keuzes.

Tabel 4.2.1: Concept keuzes

Communicatie	RF Transceivers 	Ethernet 	Wifi 	Bluetooth 
Grafische gebruikers interface	Web interface 	Losse module 	Software 	
Processor	FPGA 	8Bit 	32Bit 	
Ontwikkelings-software	uKeil	Eclipse+ Workbench	RaisonanceRide + ARM SDK	Atmel Studio

Deze keuzes worden in de volgende paragrafen overwogen om zo tot de beste keuze te komen.

4.3 Communicatie

De communicatie tussen de interface en de gebruiksvriendelijke bediening moet zo zijn dat de bediening overal binnen het kantoor van Brunelco bereikbaar is. Dit is te bereiken door de volgende oplossingen.

RF ofwel Radiofrequentie

Hierbij wordt over radiofrequenties data gecommuniceerd. Er zijn verschillende frequenties geschikt om over te communiceren. Om dit te doen zijn 2 modules nodig een aan de kant van de klimaatkast en een aan de kant van de bediening.

Ethernet

Er wordt hierbij over een bestaand netwerk van kabels gecommuniceerd. Hierbij wordt gebruik gemaakt van een bestaand universeel IEEE 802.3 protocol. Dit protocol is binnen Brunelco op iedere gebruikers computer te bereiken. Hierdoor is het niet

noodzakelijk om extra hardware aan de grafische interface kan te gebruiken. Dit zorgt voor een betere universaliteit. Het nadeel van deze communicatiemogelijkheid is dat een kabel noodzakelijk is.

WiFi

Bij wifi wordt ook over het ethernet netwerk gecommuniceerd. Het verschil bij wifi is dat er een draadloze verbinding met het basisstation(router) maakt. Deze verbinding kan worden gemaakt in twee frequentiebanden 2.4GHz en 5,8GHz. Het nadeel van deze communicatie is dat er verstoring kan optreden. De frequenties worden namelijk ook door een tal van andere toepassingen gebruikt.

Bluetooth

Is een draadloze verbinding die zich net als wifi in de 2,4GHz band bevindt. Het verschil tussen wifi en bluetooth is dat bluetooth niet met een basisstation werkt. Bij bluetooth wordt direct gecommuniceerd met communicerende. Dit heeft als voordeel dat er geen extra infrastructuur nodig is. Bluetooth is ontworpen om op kleine afstand te functioneren met een maximum van 100 meter.

Draadloze verbindingen

Bij de draadloze verbindingen is er nog een nadeel namelijk de locatie van de climaatkast. Deze bevind zich in een kelder. Gezien deze locatie kunnen draadloze verbindingen slechter presteren.

Tabel 4.3.1: Vergelijkingstabel communicatie

	Afstand	Implementatietijd	Transmissie snelheid	kosten	universaliteit	Storingsgevoeligheid	uitslag
Telling	20%	30%	10%	10%	10%	20%	
RF Transceivers	++	+	+-	+-	--	-	20
Ethernet	++	-+	++	+	++	+	110
WiFi	+-	-	+	-	++	-	-30
Bluetooth	-	-	+-	+-	--	-	-70

4.4 Grafische gebruikersinterface

De grafische gebruikersinterface is de communicatie tussen de persoon en de aansturing. Hierbij zal de persoon een grafische weergave krijgen van instellingen en waardes. Uit de keuze voor communicatie is ethernet gekomen. Er zijn verschillende mogelijkheden voor een grafische interface in combinatie met ethernet. Door alle voor het project belangrijke parameters te vergelijken kan voor de juiste interface gekozen worden.

Web interface

Hierbij kan een webpagina in een browser worden geopend waarop instellingen zijn te besturen en te zien. Het handige van een webpagina is dat het op de meeste apparaten

mogelijk is op hem te gebruiken. Het nadeel van deze oplossing is dat je gelimiteerd bent in de mogelijkheden.

Touch display module

Hierbij zijn er twee losse modulen die met elkaar communiceren. Bij de module die de gebruiker gebruikt bevindt zich een Touch screen. Hiermee kan de gebruiker de instellingen wijzigen op het Touch screen.

Computersoftware

Bij deze optie is er computerprogramma wat wordt geïnstalleerd op een computer. De gebruiker kan dan in een vertrouwde omgeving werken. Hierbij zijn er ook extra mogelijkheden zoals het geven van een alarm mogelijk.

Tabel 4.4.1: Vergelijkingstabel grafische interface

Mogelijkheden	Implementatie-tijd	Compatibiliteit	Flexibiliteit	Kosten	Ondersteuning	Uitslag
Telling	20%	30%	10%	10%	10%	20%
Web interface	+-	+	+	-	+	-
Losse module	+	-	++	-	--	++
Computer software	+	++	-+	+	+	++
						140

4.5 Processor

De processor zal binnen het project zorgen voor de communicatie tussen grafische interface en de klimaat kast. De eisen aan van de processor zijn gebaseerd op de technische en functionele vereisten. Hiermee wordt ook rekening gehouden met de interfaces en de extra functies die de processor moet uitvoeren.

FPGA

Ofwel field programmable logic array. Hiermee kan door logische componenten een processor worden samengesteld die precies aan de eisen voldoet die gesteld worden. Het nadeel is dat de implementatie tijd hierdoor hoger wordt.

8Bit

Deze microprocessoren hebben een kern die 8bits per keer verwerken. Ze worden vaak ingezet voor kleine taken en hebben beperkte geheugens.

32Bit

Deze microprocessoren zijn op dit moment een van de meest geavanceerde microcontrollers. Tegenwoordig zijn de meeste 32 bit microprocessoren uitgerust met een cortex core. Dit is het instructie gedeelte van de processor. Deze is eenmaal ontwikkeld cortex en wordt door vrijwel alle fabrikanten in hun microprocessoren gebruikt.

Tabel 4.5.1: Vergelijkingstabel Processor

	Mogelijkheden	Implementatietijd	Snelheid	Kosten	Ondersteuning	Uitslag
Telling	20%	30%	10%	20%	20%	
8 Bit Micro Processor	-	++	--	++	+	80
32 Bit Micro Processor	+	+	+	+	++	120
FPGA	-+	--	+	+-	-	-70

4.6 Ontwikkelingssoftware

De ontwikkelingssoftware is nodig om de gekozen processor van firmware te voorzien. Er zijn verschillende ontwikkelingsplatformen waar uit gekozen kan worden.

uKeil

Is een softwarepakket met een eigen toolchain en bibliotheken. uKeil ondersteund een grootte variëteit van microprocessoren. Keil is eigendom van ARM de maker van de cortex processoren.

Eclipse + Workbench

Eclipse is een opensource programma waarin de talen C en C++ kunnen worden geschreven. Doormiddel van een plug-in Workbench kan er software worden gemaakt voor microprocessoren. Deze software wordt op dit moment geïmplementeerd bij Brunelco en kan voor beide partijen een goed leerproces zijn.

RaisonanceRide

Heeft zoals uKeil zijn eigen Toolchain. RaisonanceRide bied een aantal extra functies zoals een geïntegreerde werking met SVN een opslag oplossing speciaal voor software. Hier is voorheen bij Brunelco mee gewerkt ondersteuning is hier dan ook bij aanwezig.

Atmel Studio

Dit is ontwikkelingssoftware gemaakt voor uitsluitend Atmel processoren. Het is gemaakt in een schil van Visual Studio van Microsoft en daardoor erg herkenbaar en uitgewerkt.

Tabel 4.6.1: Vergelijkingstabel Ontwikkelsoftware

	Mogelijkheden	Implementatietijd	Kosten	Ondersteuning	Uitslag
Telling	20%	40%	20%	20%	

uKeil	+	-+	-	-	-20
Eclipse + Workbench	+	++	++	+	160
RaisonanceRide + ARM SDK	+	+	-	++	100
Atmel Studio	-	++	++	-+	100

4.7 Concept Keuze

Tabel 4.7.1: Concept keuze

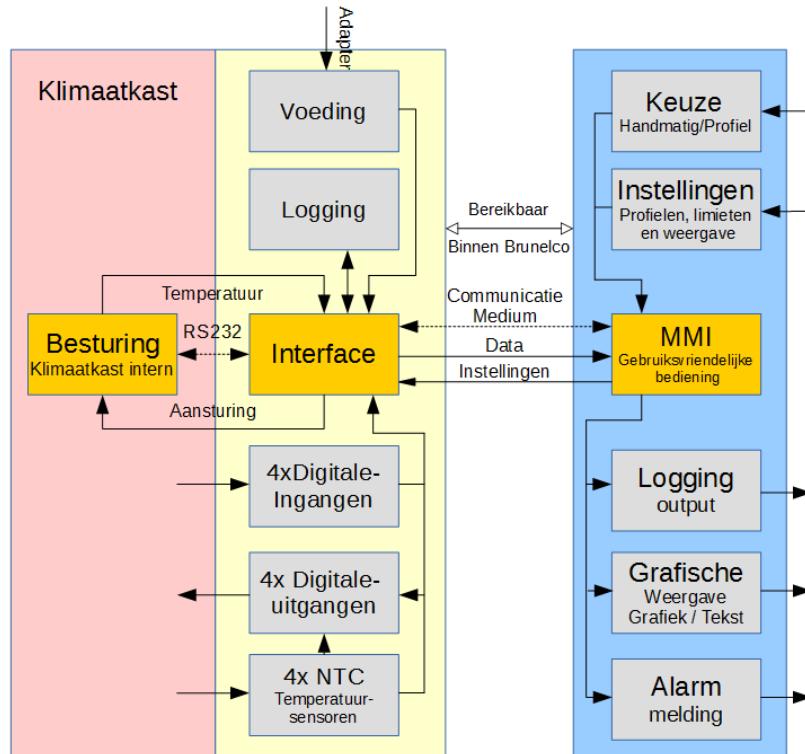
Communicatie	RF Transceivers	Ethernet	Wifi	Bluetooth
Grafische gebruikers interface	Web interface	Losse module	Software	
Processor	FPGA	8Bit	32Bit	
Ontwikkelings-software	uKeil	Eclipse + Workbench	RaisonanceRide + ARM SDK	Atmel Studio

De keuze is gevallen op een concept waarbij een 32bit microprocessor over een ethernet verbinding communiceert naar applicatie op een PC. Deze keuze is door het afwegen van de voordelen en nadelen gemaakt. Hierbij is er ook gekeken naar de infrastructuur die al aanwezig is bij Brunelco. Voor het ontwikkelen van de firmware voor de interface is gekozen voor Eclipse + Workbench. Deze keuze is gebaseerd op de ondersteuning en kosten hiervoor.

5 Functioneel ontwerp

Het functioneel ontwerp is opgedeeld meerdere disciplines. Per discipline is er een functioneel ontwerp.

5.1 Overzicht functioneel ontwerp



Figuur 5.1.1: Overzicht functionele functies

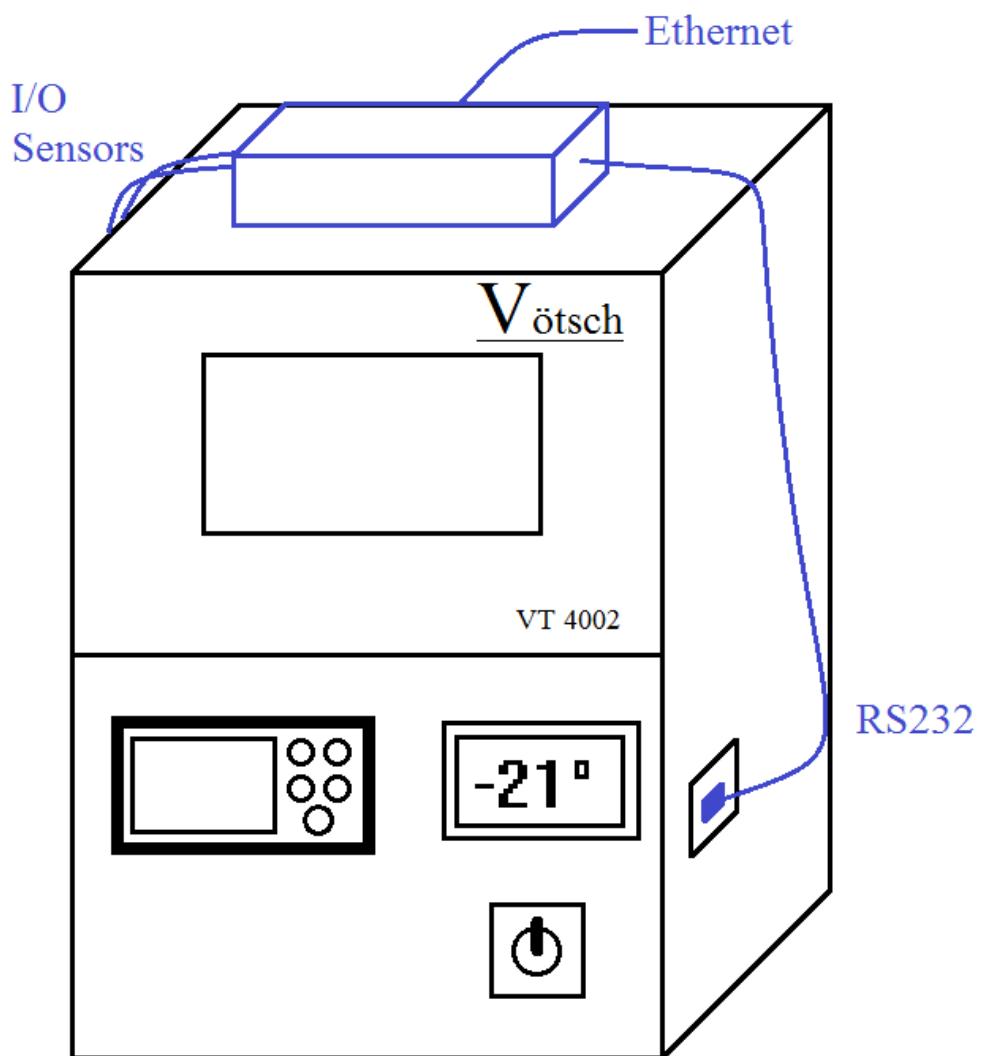
Tabel 5.1.1: Legenda kleur betekenis

Kleur	Beschrijving
Roze	Klimaatkast
Geel	Interface module
Blauw	Gebruikers interface software
Grijs	Functie blokken
Oranje	Hoofd functie blokken

In het ontwerp in Figuur 5.1.1: Overzicht functionele functies wordt onderscheid gemaakt tussen kleuren. In Tabel 5.1.1: Legenda kleur betekenis staat een legenda met bijbehorende kleuren.

5.2 Mechanisch functioneel ontwerp

In dit project worden geen specifieke mechanica gemaakt. Wel wordt er gebruik gemaakt van een kant en klare behuizing afhankelijk van de grootte van de elektronica. Deze behuizing wordt gebruikt om zo te voldoen aan de vereisten van de beschermingsgraad van het product. In Figuur 5.2.1: Mechanische schetsbevindt zich in een schets van de klimaat kast en de interface. Hierbij is de blauwe kleur het product en de zwarte kleur de bestaande klimaat kast.



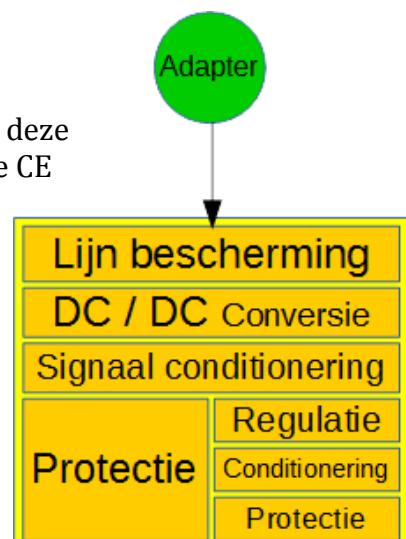
Figuur 5.2.1: Mechanische schets

5.3 Elektronisch functioneel ontwerp

Voedingsmodule

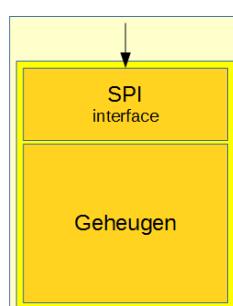
Deze module heeft als functie alle andere modulen voeden. Aan deze module wordt een adapter aangesloten om zo te voldoen aan de CE vereisten. Vervolgens wordt er een lijnbescherming toegepast. Hierbij wordt het signaal tegen kortsluiting en ESD beveiligd. Ook zal er een EMI filter worden gebruikt om emissie tegen te gaan. Hierna volgt de DC naar DC conversie om het signaal om te zetten naar de benodigde spanning. Hierop volgt een signaal conditionering om interferenties met andere componenten te voorkomen. Dit wordt gedaan doormiddel van filtering. Vervolgens zal het signaal worden beveiligd met een zekering. Daaruit volgt de eerste voedingsspanning. Vanaf

de signaal conditionering volgt een regulatie om het signaal te verzwakken naar een lagere spanning. Vervolgens zal het signaal opnieuw worden geconditioneerd. Vervolgens volgt een protectie.



Figuur 5.2.1: Blokschema
Voedingsmodule

Logger



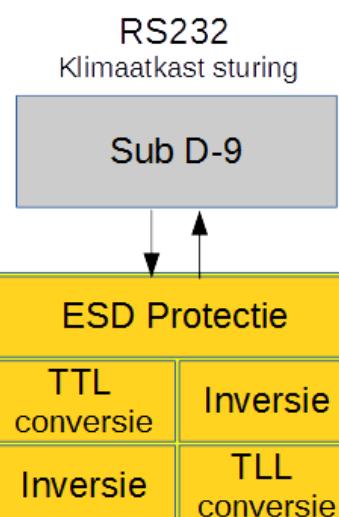
Datalogger

Om ervoor te zorgen dat als de voeding uitvalt of de verbinding met de computersoftware uitvalt is er een logger. Deze moet ervoor zorgen dat data bewaard blijft ook zonder voeding. Als interface om met de processor te communiceren wordt SPI gebruikt. Dit wordt gebruikt om snel serieel te kunnen communiceren. Vanuit de interface wordt gecommuniceerd met het onderliggende geheugen.

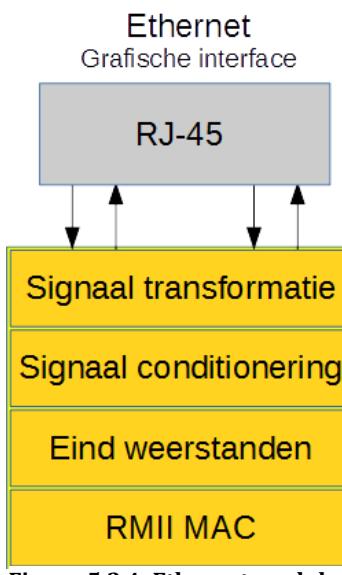
Figuur 5.3.2: loggingsmodule

RS232 aansturing

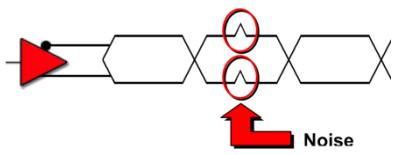
In deze module wordt van een UART input vanaf de processor een RS232 signaal gemaakt. De data in signalen volgen een andere weg als de data uit. Bij data in wordt na de connector eerst ESD protectie toegepast. Dit om statische schokken te weerhouden van het maken van schade aan de elektronica. Het signaal wat binnen komt heeft andere logische niveaus dan vereist zijn. Om deze toch te kunnen gebruiken moet een conversie plaatsvinden naar TTL. De signaal niveaus zijn nu nog omgedraaid. Door inversie wordt dit weer teruggedraaid. Bij data uit wordt als eerste ESD Protectie toegepast vervolgens inversie en vervolgens TTL Conversie.



Figuur 5.3.3: RS232 module



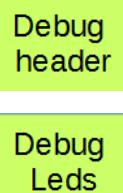
Figuur 5.3.4: Ethernet module



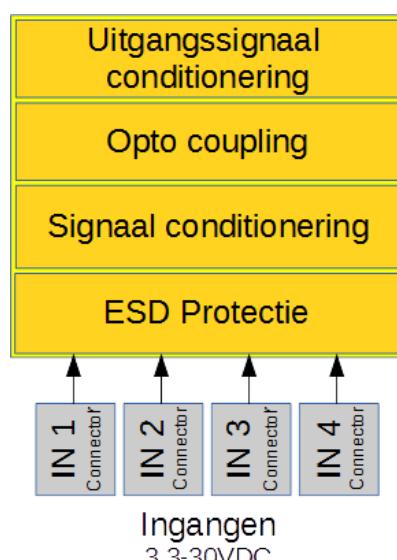
Figuur 5.3.5: Differentiaal

Debug

Om tijdens het schrijven van software feedback vanuit het systeem te krijgen zijn er debug modulen. De debug header heeft een connectie met de processor via uart. De processor kan dan zo worden geprogrammeerd om feedback informatie naar de debug header te sturen. Het doel is dan om met een UART naar USB-converter op een computer de debug informatie te ontvangen. De debug LEDs hebben een doel om snel een visuele feedback te geven. Deze worden doormiddel van een uitgang van de processor aangestuurd.



Figuur 5.3.6:
Debug modulen



de processor.

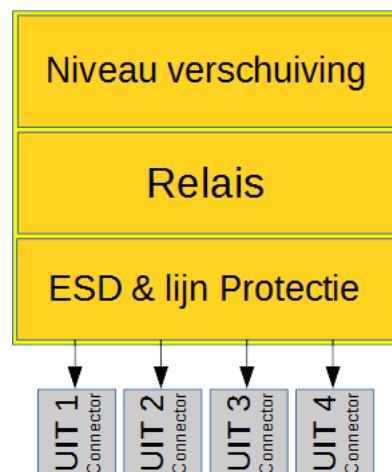
Figuur 5.3.7: Digitale
ingangsmodule

Ethernet

De ethernet module dient voor de communicatie tussen de grafische interface en de interface. De signalen worden over een gedraaide paren kabel met aan de uiteinden een RJ-45 stekker gestuurd. Dit is de standaard die hoort bij ethernet. De signalen die over de paren gaan zijn differentiaal. Dit houdt in dat het signaal over beide aders wordt gevoerd waarbij een signaal geïnfereerd is. Hierdoor zal bij een storing die optreedt het signaal op beide aders worden verstoord. Doordat het signaal in verhouding tot elkaar niet veranderd kan het signaal toch storingsvrij worden getransporteerd. Het differentiële signaal dient weer terug te worden getransformeerd naar een digitaal signaal. Het signaal zal doormiddel van transformatoren eerst galvanisch gescheiden worden met transformatoren. Vervolgens zal het signaal worden geconditioneerd. Hierna zal doormiddel van eindweerstand de juiste belasting voor de transformator worden gegenereerd. Dit om ervoor te zorgen dat het signaal juist belast is. Vervolgens zal het signaal worden opgepikt door de RMII MAC. Dit is een media acces controller die de pakket en adres afhandeling doet.

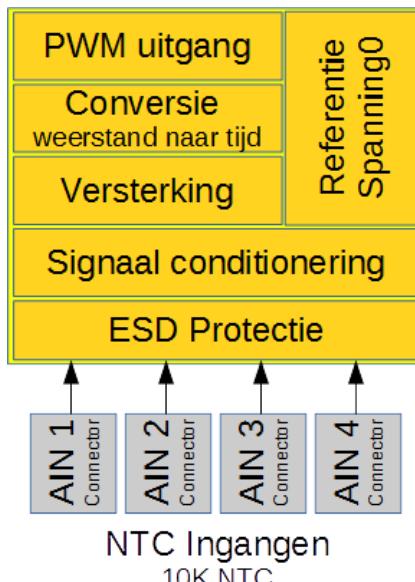
Uitgangen

Deze module bevat vier uitgangen die door het systeem worden geschakeld bij grensoverschrijdende waarden. Als interne ingangen heeft de module vier digitale ingangen. Als eerste worden deze door een niveau verschuiving naar een juiste voedingsspanning gebracht. Vervolgens zullen de signalen op de spoelen van het relais worden gezet. Bij de uitgang van het relais wordt een ESD en een lijn protectie aangebracht. Dit om het circuit tegen kortstondige spanningspieken en interferentie te beschermen.



Uitgangen
250VAC 8A / 30VDC 8A

Figuur 5.3.8: Digitale uitgangsmodule



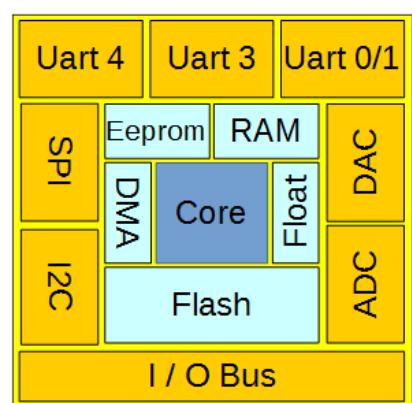
Figuur 5.3.9: NTC Module

NTC ingangen

Deze module dient voor het inlezen van vier NTC-weerstanden om temperatuur te meten. De module heeft vier ingangen voor NTC-weerstanden. Als uitgang heeft de module een 1-wire interface. Na de aansluiting van de NTC-weerstand bevindt zich een protectie blok. In dit blok is een beveiliging tegen ESD. Dit gebeurt om bij een statische ontlading ervoor te zorgen dat er geen elektronica wordt beschadigt. Vervolgens zal het signaal worden geconditioneerd worden om een beter meet bereik te verkrijgen. Vervolgens zal het signaal versterkt worden om het op niveau te brengen. Hierna zal een conversie plaats vinden van weerstand naar tijd. Dit om zo het signaal te kunnen versturen via 1-Wire. Na de signaal conditionering bevindt zich een referentie spanning. Door deze terug te meten door de processor kan het signaal met een hogere precisie worden ingelezen.

De processor

Deze module stuurt alle andere modulen aan en is het spreekwoordelijke hart van de interface. In het midden bevindt zich de instructie set. Hieromheen bevinden zich interne functies. Hieromheen bevinden zich naar buiten communicerende functies. Vanaf de instructie set wordt er verbinding gemaakt met het flash geheugen. Op dit geheugen staan programma-instructies staan die worden uitgevoerd. Het Ram geheugen zal worden gebruikt voor de tijdelijke opslag van variabelen. De eeprom wordt gebruikt voor variabelen die ook na een reset worden vastgehouden. Het DMA-geheugen is ook een ram geheugen. Het speciale aan het



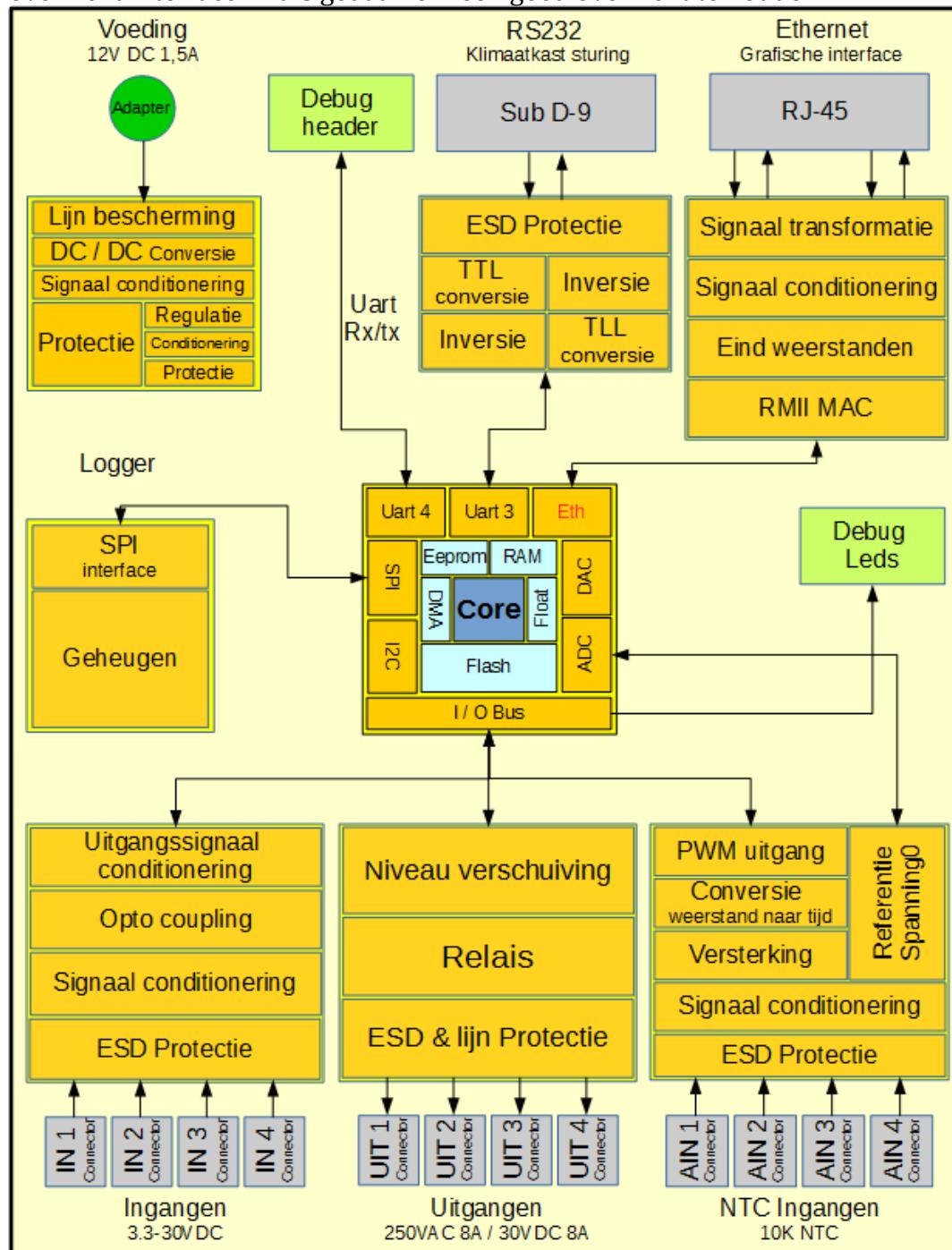
Figuur 5.3.10: Processor module

DMA-geheugen is dat dit door de omliggende functie blokken direct kan worden aangeroepen. Dit wordt gebruikt om niet bij iedere gebeurtenis een instructie te hoeven uitvoeren. Als een functie een bepaalde hoeveelheid heeft ontvangen kan alle data in een keer worden verwerkt. Naast de core bevindt zich als laatste een floating processor. Dit is een processor deel dat speciaal voor het snel verwerken van komma getallen is. De processor bevat UART, SPI en I2C dit zijn

communicatie bussen en worden gebruikt om met andere modulen te communiceren. Om met analoge spanningen te kunnen werken heeft de processor een ADC en een DAC. Hiermee kan de processor conversies doen van digitaal naar analoog en van analoog naar digitaal. Als laatste heeft de processor een I/O bus hier zitten in- en uitgangen aan. Deze zijn voor diverse toepassingen te gebruiken waaronder het aansturen van LEDs, inlezen van digitale ingangen, aansturen van schakel elementen enzovoorts.

De totale interface

Alle elementen zijn hier aan elkaar gekoppeld om tot een geheel te komen. De voeding staat in verbinding met alle modulen. Dit is echter niet te zien in Figuur 5.3.11: Totaal overzicht interface. Dit is gedaan om een goed overzicht te houden.



Figuur 5.3.11: Totaal overzicht interface

Tabel 5.3.1: Legenda elektronisch ontwerp

Kleur	Beschrijving
Donker blauw	Processor kern
Licht blauw	Processor functies
Geel	Functie groep blokken
Oranje	Functie blokken
Groen	Software ontwikkeling functies
Donker groen	Adapter aansluiting
Grijs	In en uitgangen

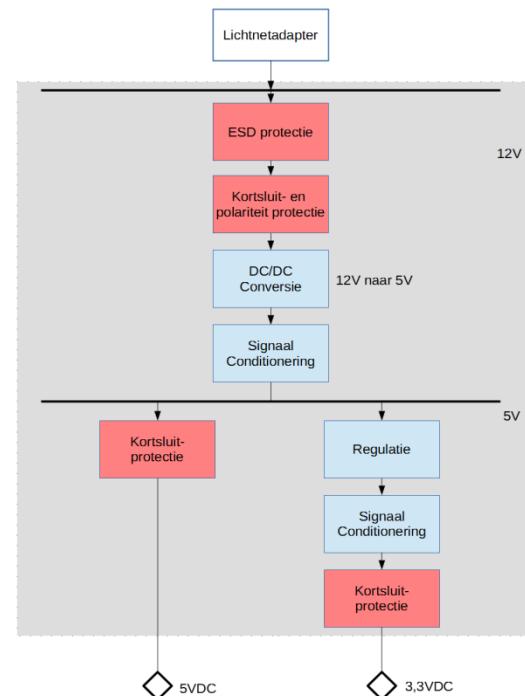
Er zijn een aantal aansluitingen die naar buiten worden gevoerd.

Tabel 5.3.2: Aansluitingen interface

Naam	Beschrijving	Relatie	Waarde	Eenheid
In 1 t/m 4	Digitale ingangen	><	3.3-30	Volt DC
A In 1 t/m 4	10k ohm NTC ingangen	=	Waar	-
Uit 1 t/m 4	Relais contacten 250VAC of 30VDC	<	8	Ampère
Sub D-9	RS232 aansluiting	=	Waar	-
RJ-45	Ethernet aansluiting	=	Waar	-

5.4 Elektrisch functioneel ontwerp

Het elektrisch functioneel ontwerp bestaat uit de voedingstopologie. In het ontwerp is gebruik gemaakt van een dubbel geïsoleerde lichtnetadapter. Dit is gedaan om zo makkelijker aan de CE-eisen te voldoen. Door het gebruik hiervan ben je isoleert van het lichtnet waardoor de eisen hiervoor niet gelden voor de te maken interface.



Figuur 5.4.1: Elektrisch ontwerp

Tabel 5.4.1: Beschrijving in en uitgangen

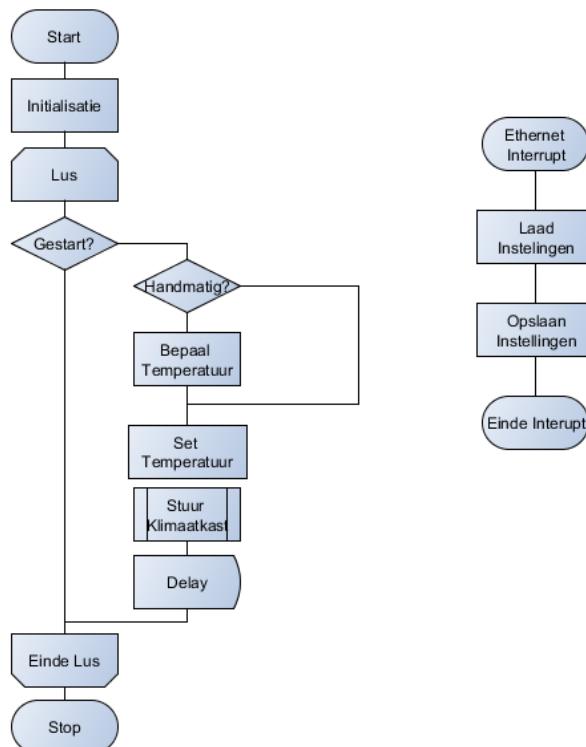
Naam	Beschrijving	Relatie	Waarde	Eenheid
Ingang	Lichtnetadapter, 12 Volt	=<	1,5	Ampère
Uitgang	Voedingsspanning V1, 5V	=<	1	Ampère
Uitgang	Voedingsspanning V2, 3,3V	=<	0,5	Ampère

Het schema in **Fout! Verwijzingsbron niet gevonden.** bestaat uit onderdelen met eerde kleuren. Hierbij zijn rode blokken voor veiligheid, blauwe voor signaal bewerking en wit voor de voedingsadapter.

Er zal een spanning aan worden gebracht van 12 Volt gelijkspanning. Deze spanning wordt hierna beveiligd op kortsluiting, polariteit en elektrostatische lading (ESD). Hierna wordt een conversie gedaan met een geschakelde voeding naar 5 Volt gelijkspanning. Hierna zal het signaal worden afgevlakt naar de juiste waarden. Deze aflatting is er om de elektronica te voorzien van een stabile voeding. Hierna volgt een splitsing naar een kortsluit protectie naar de uitgang van de 5 Volt. De andere splitsing zal naar de regulatie gaan voor de 3.3 Volt. Hierna zal de 3.3 volt worden afgevlakt om stabiele voeding te creeren. Hierna wordt een kortsluitingsbeveiliging gedaan om de veiligheid te verhogen.

5.5 Software functioneel ontwerp

De software bestaat uit 2 delen, een hoog niveau en een laag niveau. Het hoge niveau is de userinterface. Deze software draait als een applicatie op een computer. Het lage niveau is de software in de microprocessor. Hier worden de rechtstreekse instructies gemaakt voor de processors instructie set.



Figuur 5.5.1: Interface software diagram

Initialisatie

In dit blok worden verschillende interface geïnitialiseerd. Er wordt begonnen met het instellen van de in en outputs hierna worden alle protocollen geïnitialiseerd. Vervolgens zal de ethernet module worden geactiveerd. Er wordt ingesteld dat er bij een inkomend bericht over ethernet de interrupt routine wordt aangeroepen.

Hoofdfunctie

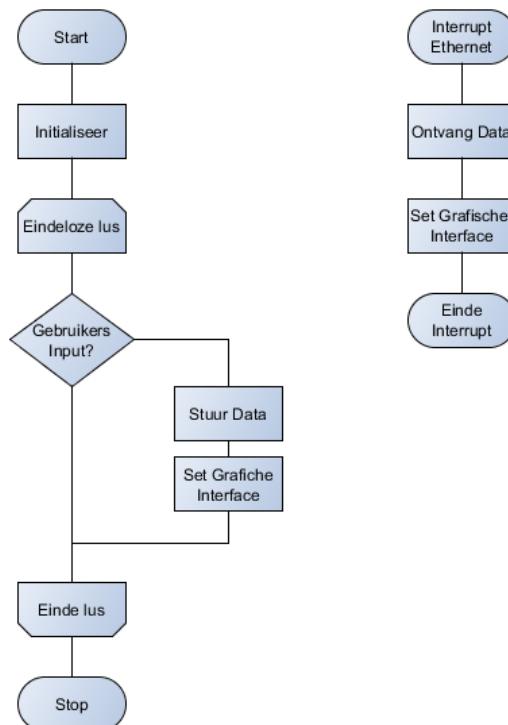
In de hoofdfunctie zal een eindeloze lus lopen. Binnen deze lus wordt er gekeken of er een start instructie is binnen gekomen. Als dit het geval is zal er worden gekeken of de instelling staat op handmatig. Als dit niet het geval is worden de programma instellingen geladen en de temperatuur bepaald. Hierna zal in beide gevallen de temperatuur verstuurd worden over ethernet. Vervolgens zal de temperatuur naar de klimaat kast gestuurd worden. Hierop volgt een delay om na een seconde opnieuw te beginnen. Dit om zo een verversingstijd te hebben van 1 Hertz.

Ethernet interrupt

De interrupt routine wordt aangeroepen als er een Ethernet pakket wordt ontvangen.

Als dit gebeurt wordt het pakket ontvangen en worden de instellingen geladen.

Vervolgens worden de instellingen opgeslagen.



Figuur 5.5.2: PC software diagram

Initialisatie

Deze functie is voor het klaar zetten van het programma. Het begint met het plaatsen van alle grafische componenten en de events die hierbij horen. Vervolgens wordt de ethernet ontvangst interrupt aangezet. Dit zorgt ervoor dat bij het ontvangen van een pakket een event ontstaat.

Hoofdfunctie

Hierin wordt er gekeken of er gebruikers input is. Als dit het geval is wordt de data verstuurd naar de interface via het ethernet. Vervolgens zullen de grafische componenten worden ververst met de nieuwe informatie.

Ethernet interrupt

Op het moment dat er een interrupt ontstaat wordt een pakket ontvangen. Vervolgens zal de data worden weer gegeven op de grafische componenten.

6 Interface hardware ontwerp

In dit hoofdstuk wordt het hardware ontwerp besproken van de gebruikersinterface. Dit is de hardware die zich bevindt tussen de klimaat kast en de gebruikersinterface. Hierbij is het ontwerp opgesplitst in losse onderdelen die samen een geheel vormen. Deze losse onderdelen worden in de volgende paragrafen besproken.

6.1 Voeding

De voeding dient voor het voeden van alle modulen. De voeding krijgt een ingangsspanning die vast gelegd is in de technische specificaties. Het gaat hier om een spanning die zich tussen de 11 en de 13 Volt DC bevindt. Hierop kan zich een AC component bevinden van maximaal 200mV. Uit deze voeding kan maximaal 1.5 Ampère worden ontrokken. Hierbij moet er een zekering zijn toegepast.

Binnen de module zijn er 3 spanningen die worden gebruikt 12, 5 en 3 Volt DC. De 11 tot 13 Volt dc-ingangsspanning valt binnen de normen van het schakelen van een relais en kan zodoende worden gebruikt als 12 Volt DC Schakelspanning. De 3 Volt dc-voeding wordt lokaal op het processor bord gemaakt. Hierdoor is het niet nodig om deze zelf te ontwerpen. Voor de 5 Volt DC is het wel nodig om een voeding te ontwerpen. Hierbij moet er een conversie worden gedaan van de ingangsspanning naar de 5 Volt DC.

Om de voeding te ontwerpen is er eerst een schatting nodig van het verbruik deze is te vinden in Tabel 6.1.1: Schatting maximum stroomverbruik 5 Volt DC voeding.

Tabel 6.1.1: Schatting maximum stroomverbruik 5 Volt DC voeding

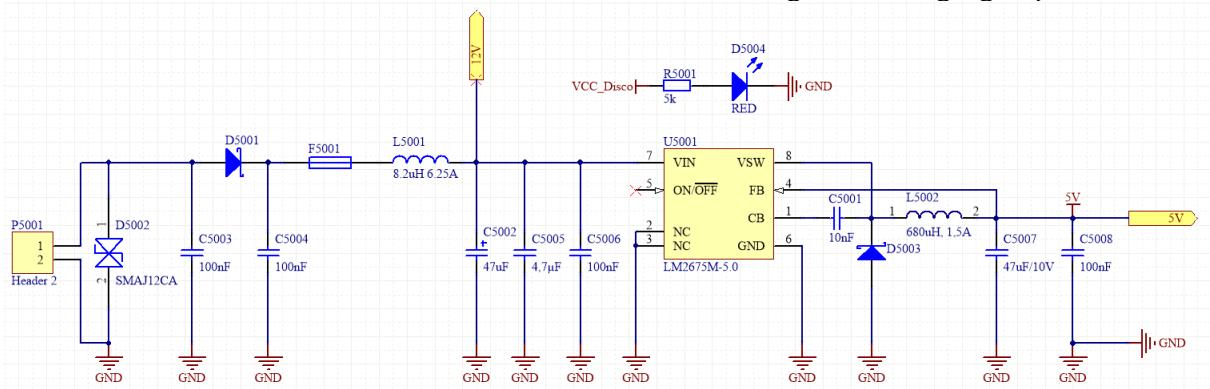
Verbruiker	Geschat maximum stroomverbruik
Processor module	600mA
RS232 module	10mA
Temperatuur meting	3mA
Digitale ingangen	20mA
Digitale uitgangen	10mA (van 5 Volt DC)
Debug LEDs	10mA
Totaal	653mA

Voor de processor module wordt een groot stroom gebruik geschat. Dit is gedaan om geen limitatie aan de module te brengen. Hierbij kan de module bijvoorbeeld worden uitgerust met mogelijkheden die veel stroom verbruiken zoals ethernet, wifi, geheugens en een display. Nadat de schatting is gemaakt kan er een voeding oplossing worden gekozen. In Tabel 6.1.2: Keuze voeding worden verschillende mogelijkheden vergeleken om zo tot een bewuste keuze te komen.

Tabel 6.1.2: Keuze voeding

Mogelijkheid	Eis	LM2675 (IC)	TSR 1-2450 (module)	TL2575 (IC)	LM1086 (Regulator)
Implementatiemoeilijkheid	-	Redelijk	laag	Redelijk	laag
Complexiteit	-	Hoog	Laag	Hoog	Laag
Prijs	-	€ 3,92	€ 4,04	€ 1,23	€2,48
Uitgangsvermogen	1A	1A	1A	1A	1.5A
Efficiëntie	-	90%	88%	77%	40%
Rimpel	-	80mV	50mV	150mV	10mV
Keuze:					

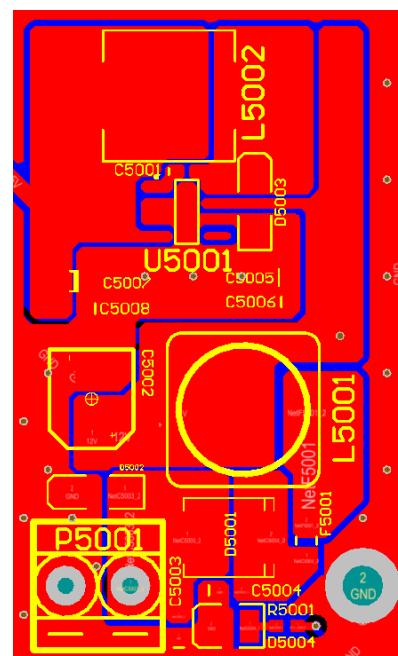
Er is gekozen voor de LM2675 omdat deze een hoge efficiëntie en een redelijk lage uitgangsrimpel heeft. De implementatiemoeilijkheid is redelijk en in vergelijking met een module is deze niet kant en klaar waardoor er ook nog een uitdaging blijft.



Figuur 6.1.1: Schema voeding

Het schema van de voeding is te zien in Figuur 6.1.1: Schema voeding. Hierbij is de signaal volgorde van links naar rechts. Aan de uiterst linkse zijde komt het voedingssignaal binnen op een connector. Hierbij zal onderscheid worden gemaakt in de positieve en de negatieve kant. Vervolgens zal er een TVSdiode over de voeding staan. Deze TVSdiode dient voor overspanning en ESD-protectie. De TVSdiode zal dan ook een lagere impedantie vormen op het moment dat er een hogere spanning wordtaangebracht dan 12V. Dit om de elektronica die in verbinding staat te beschermen. Hierna volgt een schakeling van een diode en twee condensatoren. Deze schakeling dient voor een beveiliging tegen het verkeerd om aansluiten van de voedingsspanning. De diode zal de stroom blokkeren als dit het geval is. De twee condensatoren zijn voor het filteren van hoge frequenties. Wanneer de voedingslijn lang wordt kunnen er signalen worden opgevangen. Deze signalen kunnen dan voor op slingeringen zorgen op de voedingsspanning. Dit wordt afgevangen door de condensatoren. Hierna is een zekering geplaatst om ervoor te zorgen dat bij een te hoge stroom opname zo min mogelijk schade wordt veroorzaakt. Hierna zal een spoel en een serie condensatoren volgen om voor een stabiele voedingsspanning te zorgen en eventuele rimpel spanning af te vangen. De condensatoren dienen ook als buffer voor het schakelen van de voeding. Hierna zal het voedingssignaal als ingang aan worden geboden aan de LM2675. Deze zal het aangeboden signaal schakelen naar de VSW-uitgangspin. Dit is een typische buck conversie schakeling. De LM2675 zal met behulp van de terugkoppeling de duty cycle bepalen. Dit om zo tot de gewenste spanning te komen. Ook zal hij bij kortsluiting en een thermische fout stoppen met schakelen. Dit om schade te voorkomen. Na de buck schakeling zit nog een condensator ter extra afvlakking. Hierna zal het voedingssignaal de verschillende modules voeden. Om een nog betere conditionering te bereiken worden een aantal modules lokaal extra geconditioneerd.

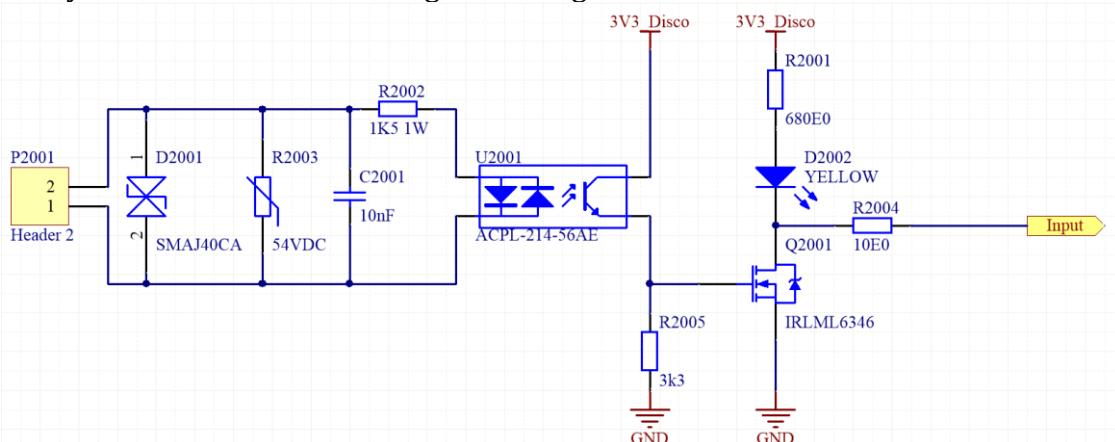
In het pcb-ontwerp in Figuur 6.1.2: PCB ontwerp voeding is te zien dat de diode en de spoel dicht bij de LM2675 zijn geplaatst. Dit is gedaan om ervoor te zorgen dat de stroom paden zo klein mogelijk zijn. Dit om ervoor te zorgen dat de regellussen intact blijven.



Figuur 6.1.2: PCB ontwerp voeding

6.2 Digitale ingangen

De digitale ingangen zijn voor het inlezen van signalen die door de gebruiker worden aangebracht. Deze dienen daarom ook goed beveiligd te zijn om eventuele gebreken aan het systeem door verkeerd aangeboden signalen.



Figuur 6.2.1: Schema digitale ingangen

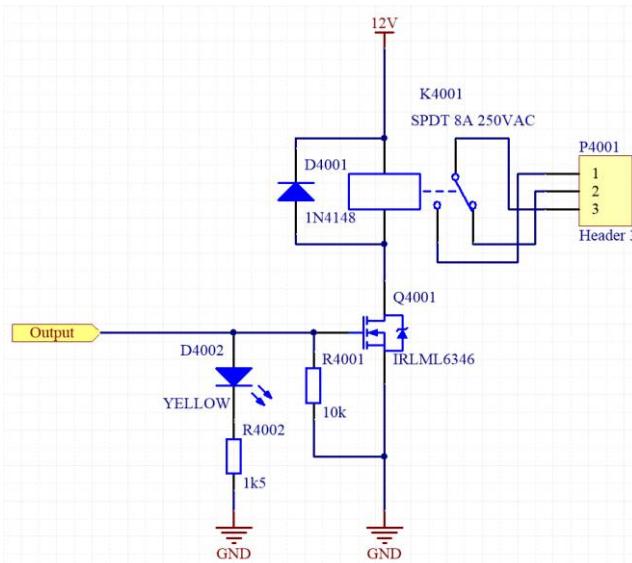
In het schema in Figuur 6.2.1: Schema digitale ingangen is maar een ingang te zien. Er zijn in totaal vier inputs in het ontwerp. De gebruiker heeft de mogelijkheid om een signaal op de connector aan te sluiten. Een logische 0 zal gedetecteerd worden tussen -0.3 en 0.3 volt DC. Een logische 1 tussen 3 en 30 Volt DC. Het signaal mag een potentiaalverschil hebben van maximaal 30 Volt DC. Het signaal zal bij een spanning hierboven worden belast door een varistor en een TVS diode. Om korte spanningspieken snel op te kunnen vangen en het tegen gaan van spanningswisselingen is er kleine capaciteit geplaatst. Vervolgens is er een opto-coupler geplaatst dit om ervoor te zorgen dat er een galvanische scheiding ontstaat tussen de gebruikers ingang en de interface elektronica. Hierdoor is het niet mogelijk om eventuele storingen of spanningsverschillen door verschillende voedingen over te brengen. Het is alleen mogelijk om de ingang defect te maken. De opto-coupler heeft een bi-directionele ingang. Hierdoor is het mogelijk om zowel een negatief als een positief signaal aan te bieden. Door de gekozen weerstand van 1500ohm zal er bij 3 Volt DC 2mA door de LED diode lopen en bij 30 Volt DC 20mA. In Figuur 6.2.2: Berekening vermogen weerstand is te zien dat de dissipatie van de weerstand hoog wordt bij 20mA daarom is er gekozen voor een 1 watt vermogens weerstand. Door de weerstand waarde zal in de gehele range de fototransistor open schakelen. Als dit gebeurt zal dit zorgen voor een logische 1 op het punt boven de uitgangsweerstand. Vervolgens zal hierdoor de MOSFET worden ingeschakeld waarmee een led indicatie wordt geactiveerd. Dit signaal zal vervolgens via een serie weerstand naar een ingang van de processor gaan.

$$P_{Weerstand} = (V_{In} - V_{Opto-coupler}) * I_{Weerstand} = (30V - 1.2V) * 0.02A = 0.576W$$

Figuur 6.2.2: Berekening vermogen weerstand

6.3 Digitale uitgangen

De digitale uitgangen dienen voor het schakelen van een door de gebruiker gedefinieerde bron. Dit schakelen kan met spanningen tot maximaal 250 Volt AC en 30 Volt DC. Ook is de maximumstroom hiervan gelimiteerd tot 8 Ampère. Dit schakelen wordt geactiveerd door de interface bij een door de gebruiker gedefinieerde temperatuur.



Figuur 6.3.1: Schema digitale uitgang

In het schema in Figuur 6.3.1: Schema digitale uitgang is voor een digitale uitgang. In het ontwerp zijn 4 digitale opgenomen. Op de ingang van de uitgangsschakeling is het mogelijk om een logische 1 van 3,3 Volt DC of een logische 0 van 0 Volt te zetten. Bij een logische 0 zal de schakeling in rust zijn en het relais niet geschakeld zijn. Om te voorkomen dat bij een reset van de processor de lijn gaat zweven en de MOSFET toch in geleiding komt is er een weerstand naar ground geplaatst aan de gate. Op het moment dat vanuit de processor een logische 1 wordt gezet zal de schakeling geactiveerd worden. Hierdoor zal de led indicatie aan gaan om de gebruiker feedback te geven. Dit zal tevens de MOSFET inschakelen. Deze zal het relais schakelen. Door het gebruik van een relais wordt er een galvanisch scheiding van het signaal van de gebruiker gecreëerd. Dit relais wordt geschakeld door de 12 Volt DC spanning. Dit om op de 5 Volt DC minder storing en vermogensafname te zorgen. Over het relais is een diode geplaatst om bij het schakelen van de spoel van het relais de opgenomen stroom niet te ontladen op het schakelende element. Dat is in dit geval dus de MOSFET.

6.4 Temperatuur meting

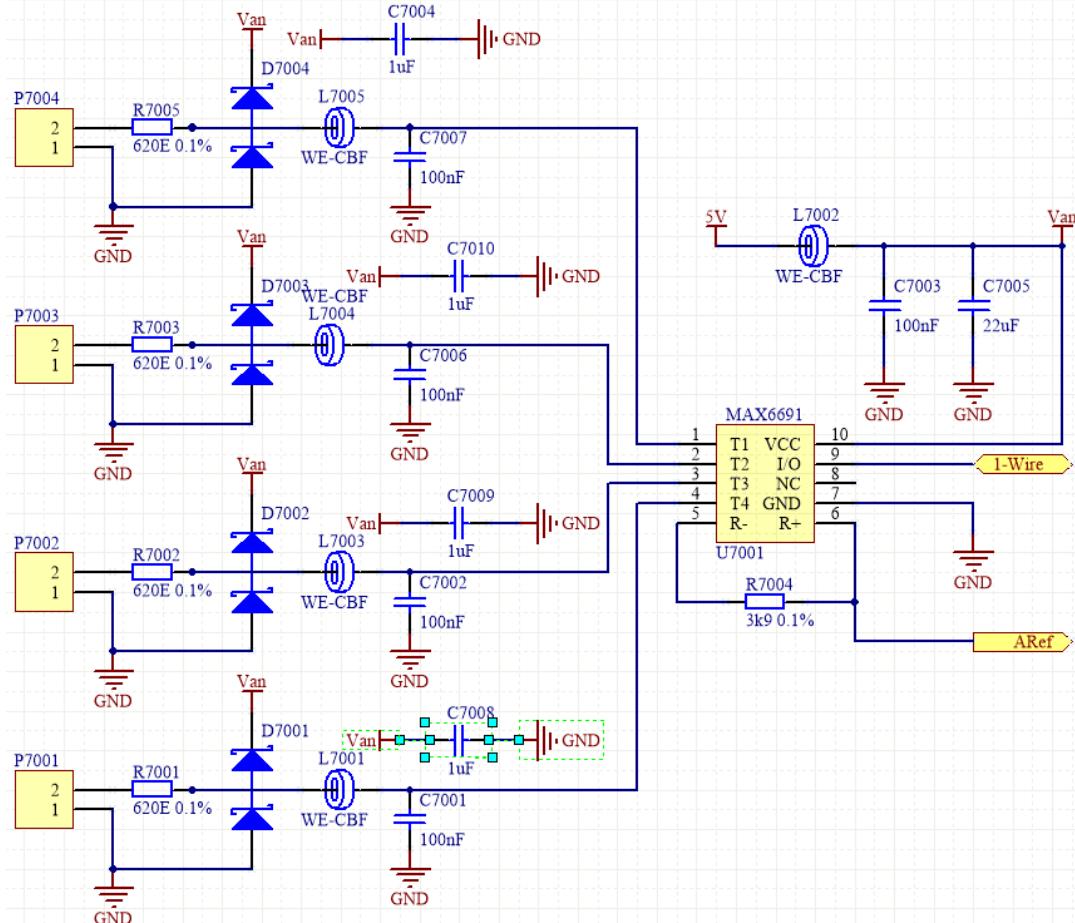
Voor dit project is het nodig om 4 NTC temperatuur sensoren te kunnen meten. Deze temperatuur sensoren worden uitgelezen door de processor. Er zijn een aantal oplossingen om dit te doen. In Tabel 6.4.1: Keuze NTC meting zijn de meetinstrumenten weergegeven met hun belangrijkste eigenschappen.

Tabel 6.4.1: Keuze NTC meting

Mogelijkheid:	Eis	12 bit ADC	MAX6698	MAX6682	LTC2983	MAX6691
Aantal NTC sensoren per oplossing:	-	5	3	1	20	4
Interface	-	Intern	I2C	SPI	SPI	PWM
Temperatuur meetbereik	-40-130°C	-40-130°C	-30-127	0-50°C	-55-300°C	-55-300°C
Implementatiemoeilijkheid	-	laag	laag	laag	Redelijk	Redelijk
Complexiteit		Middel	Laag	Laag	Hoog	Middel
Prijs voor 4 mogelijkheden	-	€ 0	€ 9,98	€ 9,48	€ 27,22	€ 4,36
Resolutie meting	0,1°C	0,415°C	2°C	0,125°C	0,001°C	-
Nauwkeurigheid over	2°C	-	3,5°C	0,375	0,1°C	-

meetbereik					
Keuze					

Door vergelijking is voor de MAX6691 gekozen. Hierbij is gekeken of de oplossing aan de gestelde eisen voldeed. De doorslag hiervoor kwam door prijs en de complexiteit. Dit IC schakelt bij een meting kort een stroombron van 1mA aan op de sensor. Hierdoor wordt er weinig zelf opwarming door de sensor gegenereerd. Vervolgens zet het IC de gemeten weerstand om naar een verhouding van een aan en uit tijd. Deze kan worden ingelezen door de microprocessor. Dit zal verder worden uitgewerkt in de software. Door gebruik te maken van verhoudingen worden eventuele fouten in de timing vermeden.



Figuur 6.4.1: Schema temperatuur inlezing

In Figuur 6.4.1: Schema temperatuur inlezing wordt begonnen met 4 connectoren voor NTC-sensoren. Vervolgens volgt een schakeling met twee dioden en een weerstand om overspanning tegen te gaan. Er is hierbij niet gekozen voor een TVS-diode of varistor omdat deze het signaal te veel kunnen beïnvloeden. Bij deze schakeling zal spanning van onder de -0.8 Volt worden kortgesloten naar ground en een voedingsspanning boven de +0.8 Volt worden kortgesloten naar de voeding. Deze spanningen komen uit de forward spanning van de diode. Om te voorkomen dat de stroom door de diode te hoog wordt, wordt er een serie weerstand geplaatst. Deze serie weerstand wordt mee gemeten in de weerstandsmeting daarom is er voor een hoge precisie gekozen. De extra weerstand zal in de calculatie door de processor worden gecompenseerd.

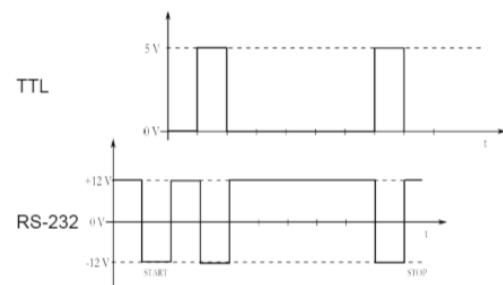
Het kan voor komen dat er een extreme hoge spanning op de ingang komt te staan door toedoen van een gebruiker. De spanning zal ervoor zorgen dat hij de voeding zou gaan

ontlasten. Op het moment dat de spanning zo hoog wordt dat hij meer vermogen levert dan de belasting nodig heeft zal de voedingsspanning verhogen. Om dit kortstondig op te kunnen vangen wordt er een $1\mu F$ capaciteit geplaatst. Deze zal kort een grote belasting vormen. Na de diodes volgt een ferriet kraal om bij een lange kabel hoogfrequente storing tegen te gaan. De condensator doet dit ook voor de laagfrequente storingen. Dit samen maakt een laagdoorlaat filter. De condensator kan niet groter zijn dan $100nF$ omdat dit de meting verstoord. De verstoring treedt op als er een sample wordt genomen van de sensor. Tijdens de sample wordt er een stroom van $1mA$ door de sensor gestuurd. Deze stroom zal de condensator opladen. Als de condensator groter wordt zal de meting niet zijn eindwaarde bereiken. Als uitgang heeft de schakeling een referentie spanning en een onewire communicatie buslijn. Deze worden beide door de processor afgehandeld. Door een goede conditionering worden verstoringen van elders op de PCB tegen gegaan.

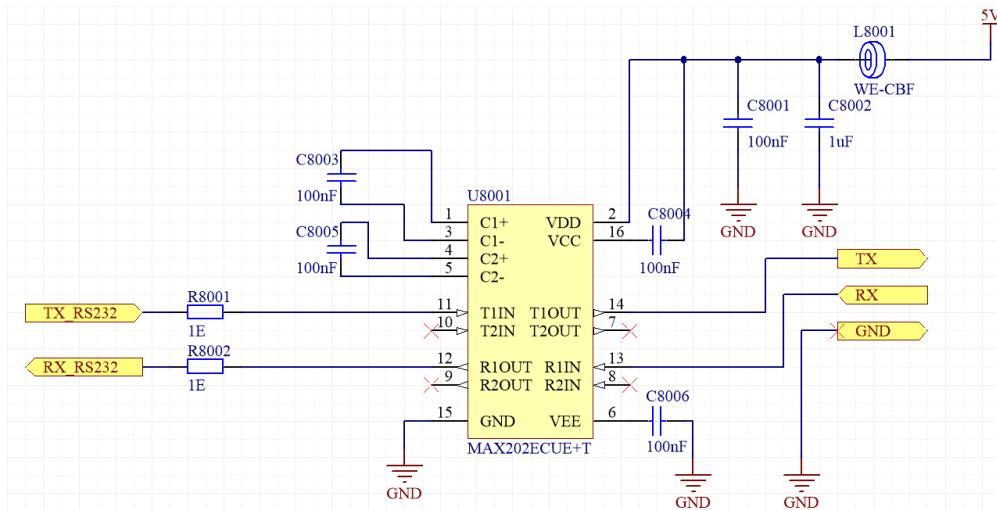
Door de limitatie van de weerstandmeting van het IC was het niet mogelijk om een $10k\Omega$ NTC te gebruiken. Het datasheet geeft aan in zijn key specificaties dat hij elke temperatuur range kan bereiken. Echter wordt dit gelimiteerd door de waarde van de NTC. Het IC doet een voltage meting van de spanningsdeler tussen R_{ext} en de NTC. Vervolgens zal het IC dit omzetten naar puls lengte. Bij een kortsluiting of open contact geeft het IC een extra korte puls weer om dit aan te geven. Nu heeft een $10K$ NTC bij $-40^\circ C$ een weerstand van honderden kilo ohms. Bij deze weerstand geeft het IC al een open contact aan. En hierdoor zijn deze temperaturen niet te meten. Dit is te veranderen door R_{ext} een hogere waarde te geven. Echter zorgt dit ervoor dat het probleem zich verschuift naar de bovenkant van de range. Hierbij zal bij $130^\circ C$ een kortsluiting worden gedetecteerd. Om toch de temperatuur range te kunnen meten wordt aanbevolen om $2k\Omega$ NTC-sensoren te gebruiken. Deze wordt bij $-40^\circ C$ en een beta van $3560K$ rond de $44k\Omega$. Hierdoor zal het IC geen open verbinding aangeven en is de gehele range te meten.

6.5 RS232module

Om de klimaat kast aan te sturen is er een RS232 verbinding met de interface. Om RS232 te maken van de UART lijnen van een processor is een conversie nodig. De niveaus van een processorlijn zijn namelijk 0 of $3.3V$ DC. Hierbij is 0 Volt een logische 0 en $3.3V$ DC een logische 1. Bij een RS232 verbinding is een logische 0 tussen de -5 en $-15V$ DC. Een logische 1 ligt tussen de 5 en $15V$ DC. In is een voorbeeld te zien van het signaal. In rust zal het signaal ook tussen de -5 en $-15V$ DC liggen. RS232 kan bestaan uit een aantal extra signalen de start en ontvangst van een datatransmissie aan te geven. Echter worden deze signalen niet ondersteund door de klimaatkast.



Figuur 6.5.1: RS232 niveaus



Figuur 6.5.2: Schema RS232 conversie

In het schema in Figuur 6.5.2: Schema RS232 conversie is te zien dat de transmissie en ontvangst lijnen van de processor komen. Deze lijnen zullen via een serie weerstand aan het IC worden aangesloten. Het IC maakt hier dan een ander niveau van om te voldoen aan de RS232 specificaties. Intern in het IC bevindt zich een geschakelde voeding die de spanningen creëert. De condensatoren zijn ten behoeve van deze voeding. De uitgang van de data signalen gaan rechtstreeks zonder ESD-protectie naar de RS232 connector. Dit is gedaan omdat de protectie zich bevindt in de MAX202. Door conditioning van de voeding zullen storingen van andere bronnen niet op de chip werken. Dit geld ook andersom de storingen die de chip eventueel zou produceren worden hierdoor ook worden tegen gehouden.

6.6 Processor module

Voor de processor dient een keuze gemaakt te worden. Wel is bekend dat het een 32 bit microprocessor zal zijn. In Tabel 6.6.1: Keuze processor wordt vergeleken met de belanghebbende functies ten opzichte van het project. Hieruit wordt vooral gekeken naar de ondersteuning en de implementatiemoeilijkheid. Ook wordt gecontroleerd of er aan de gestelde eisen wordt voldaan.

Tabel 6.6.1: Keuze processor

Mogelijkheid:	Eis	STM32F7	ATSAM70	LPC407	MK60DN512
Ondersteuning	-	Goed	Redelijk	Redelijk	Redelijk
Implementatiemoeilijkheid	-	Redelijk	Redelijk	Hoog	Hoog
Complexiteit	-				
Prijs	-	€ 15,86	€ 15,46	€ 13,63	€ 9,33
Programma geheugen	500Kb	1MB	2MB	512KB	512KB
Variabelen geheugen		384KB	384KB	96kb	128KB
Processor snelheid	100Mhz	216MHz	300MHz	120MHz	100MHz
Ethernet Mac	Ja	Ja	Ja	Ja	Ja
SPI	1	6	2	0	3
Uart	2	4	8	5	5
ADC	1	24	24	1	2
I/O	32	164	114	116	66
Keuze:					

Uit vergelijking is gebleken dat een STM32F7 processor de keuze is. Dit komt vooral door de ondersteuning die er binnen het bedrijf Brunelco voor is.

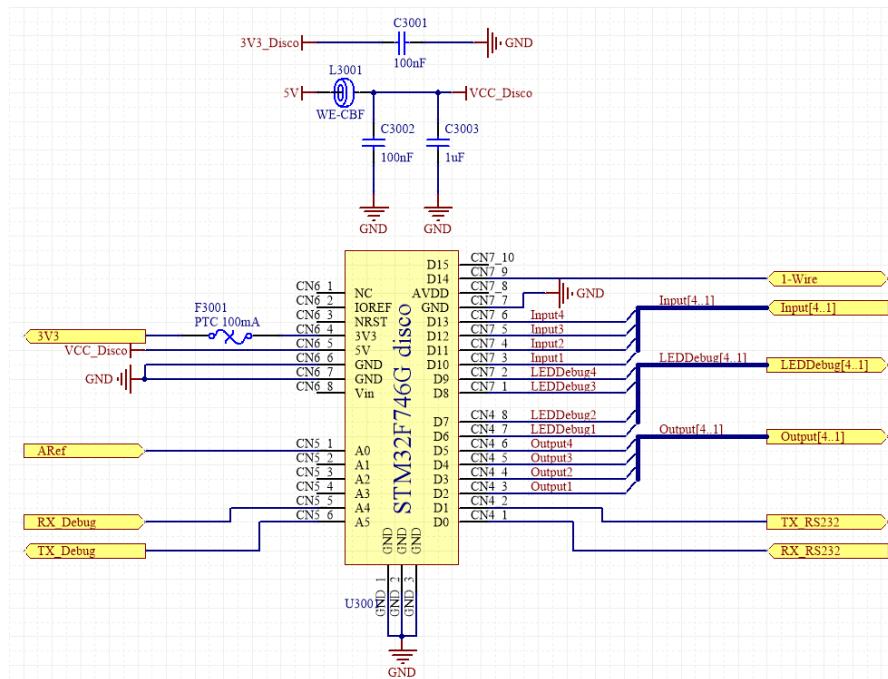
Om de processor te gebruiken is een platform nodig waar de conditionering, programmering, geheugens en timer modules op zitten. Er zijn hiervoor een aantal keuzemogelijkheden deze worden vergeleken in Tabel 6.6.2: Keuze processorplatform.

Tabel 6.6.2: Keuze processorplatform

Mogelijkheid	Eis	STM32F7-Discovery	STM32F7-Evaluation	Zelf maken
Implementatiemoeilijkheid	-	Redelijk	Redelijk	Hoog
Complexiteit	-	Middel	Middel	Hoog
Prijs	€ 80	€ 45,09	€ 563,41	€ 75(geschat)
Mogelijkheden	-	Middel	Hoog	Middel
Flash Geheugen	16Mbits	128Mbits	512Mbits	16Mbit
RAM geheugen	2Mbits	64Mbits	256Mbits	2Mbit
Keuze:				

Uit vergelijking is gebleken dat een STM32F7-Discovery het meest voldoet. Hierbij waren vooral de complexiteit en implementatiemoeilijkheid doorslaggevend. Ook is hiervoor gekozen om minder risico te lopen bij het project. Aangezien de implementatie tijd hoog is en de kansen op fouten groot wordt door de complexiteit van het ontwerp. Ook kunnen er problemen ontstaan bij het solderen van componenten met kleine afmetingen.

Het platform heeft een Arduino connector om verbindingen te leggen met ontworpen elektronica. Verder beschikt het platform over een TFT-scherm met capacitief aanraakscherm. Deze mogelijkheden kunnen eventueel later worden geïmplementeerd om het product te verbeteren maar behoren niet tot de vereisten.



Figuur 6.6.1: Schema processorplatform

In Figuur 6.6.1: Schema processorplatform is de Arduino connector te zien. Hieraan zitten alle logica pinnen die naar de verschillende onderdelen gaan. De voeding wordt na een lokale conditionering aangesloten op het platform. De uitgaande 3,3 Volt dc-voeding zal na een zelfherstellende zekering naar de andere modulen gaan. Deze uitgaande voeding dient voor het voeden van logica delen die in verbinding staan met de processor. Dit is gedaan om ervoor te zorgen dat de logica signalen de juiste niveaus hebben.

7 Interface software ontwerp

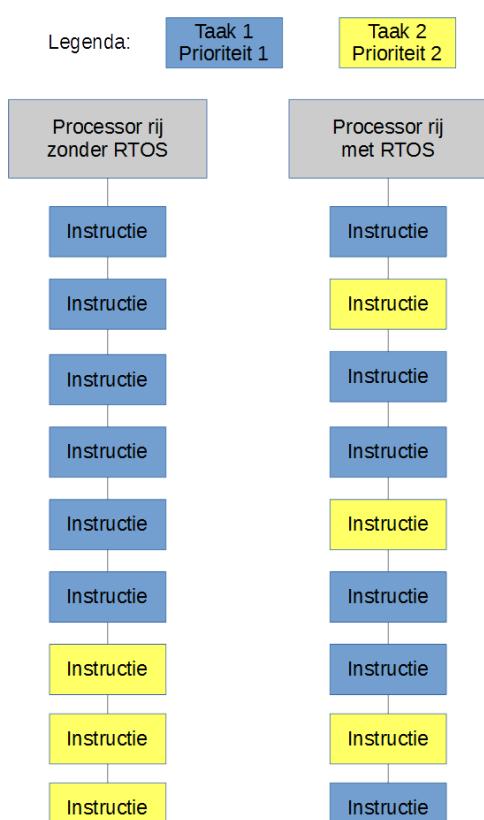
In dit hoofdstuk wordt de software besproken die zich in de processor van de interface bevindt. In deze processor worden een aantal taken uitgevoerd. Deze taken worden in de volgende paragrafen besproken.

7.1 Basis werking

De basisfunctie van de software is als volgt. Er wordt begonnen met het initialiseren van alle modulen zoals ethernet, UART, Timers en de RTOS. Vervolgens belandt de software in de hoofdtaak. Hiernaast draaien taken voor onder andere ethernet en het opslaan van geheugen. In deze hoofdtaak wordt door een lus gegaan. Deze lus voert elke keer als het de eerste seconde van de minuut is een functie uit. De functie wordt bepaald door de huidige modus die de mogelijkheden stop, programma of handmatig heeft. Tijdens de modus stop zal de module de temperaturen wel uitlezen maar hier verder niks mee doen. In de modus handmatig wordt de via de computersoftware ingestelde ingesteld en worden de temperaturen naar de computersoftware gestuurd. Ook wordt er gecontroleerd of de sensoren niet buiten de ingestelde grenzen vallen. Als dit wel het geval is zal daarop worden gereageerd met eventuele alarm meldingen en het schakelen van relais. In de modus programma wordthetzelfde gedaan als in de handmatige modus met de toevoeging van meerdere setpoints die vooraf zijn geprogrammeerd in plaats van de handmatig ingestelde temperatuur.

7.2 Real-time Besturingssysteem (RTOS)

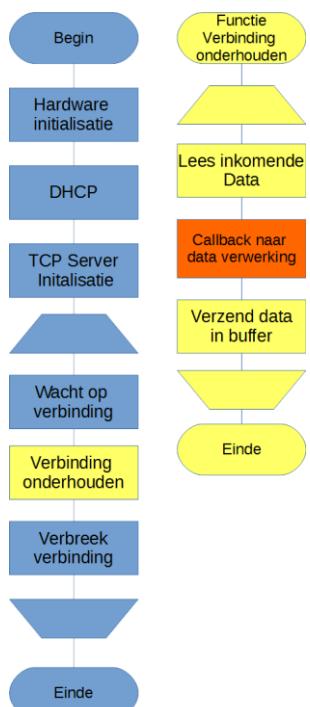
Om makkelijk te kunnen werken met de verschillende taken die moeten worden uitgevoerd in dit project wordt er gebruik gemaakt van een RTOS. Een RTOS is een besturing systeem dat meerder taken op dezelfde processor kan laten uitvoeren. Deze taken worden opgesplitst in instructies en aan de hand van hun prioriteit geplant om uit te voeren. Uit eindelijk kan de processor nog steeds maar een instructie uitvoeren maar door hoge snelheid lijkt het of er taken te gelijk worden uitgevoerd. In Figuur 7.2.1: Voorbeeld RTOS taken is het verschil te zien tussen een systeem met en zonder een RTOS. Beide systemen zijn even snel klaar alleen heeft het systeem met RTOS de taken verdeelt uitgevoerd naar hun prioriteit. In praktijk zou het systeem met RTOS langzamer zijn in deze situatie door de tijd die de RTOS nodig heeft om te schakelen tussen de instructies. Het voordeel van een RTOS wordt gehaald uit de wacht instructies bijvoorbeeld een delay. Normaal zou een processor in de wachttijd van een taak niks kunnen doen. Door deze tijd te benutten door andere taken is een RTOS in praktijk vaak vele malen sneller met het uitvoeren van taken. Denk bijvoorbeeld aan het laten knipperen van een led. Hierbij moet om de seconde een led aan en uit worden gezet. Als men dit doet met een delay zal de processor heel kort de led aan zetten en tussen tijds niks kunnen doen. Nu kan dit met een RTOS worden opgenomen in een taak en zou ondertussen een andere taak nuttige instructies kunnen uitvoeren.



Figuur 7.2.1: Voorbeeld RTOS taken

7.3 Ethernet communicatie

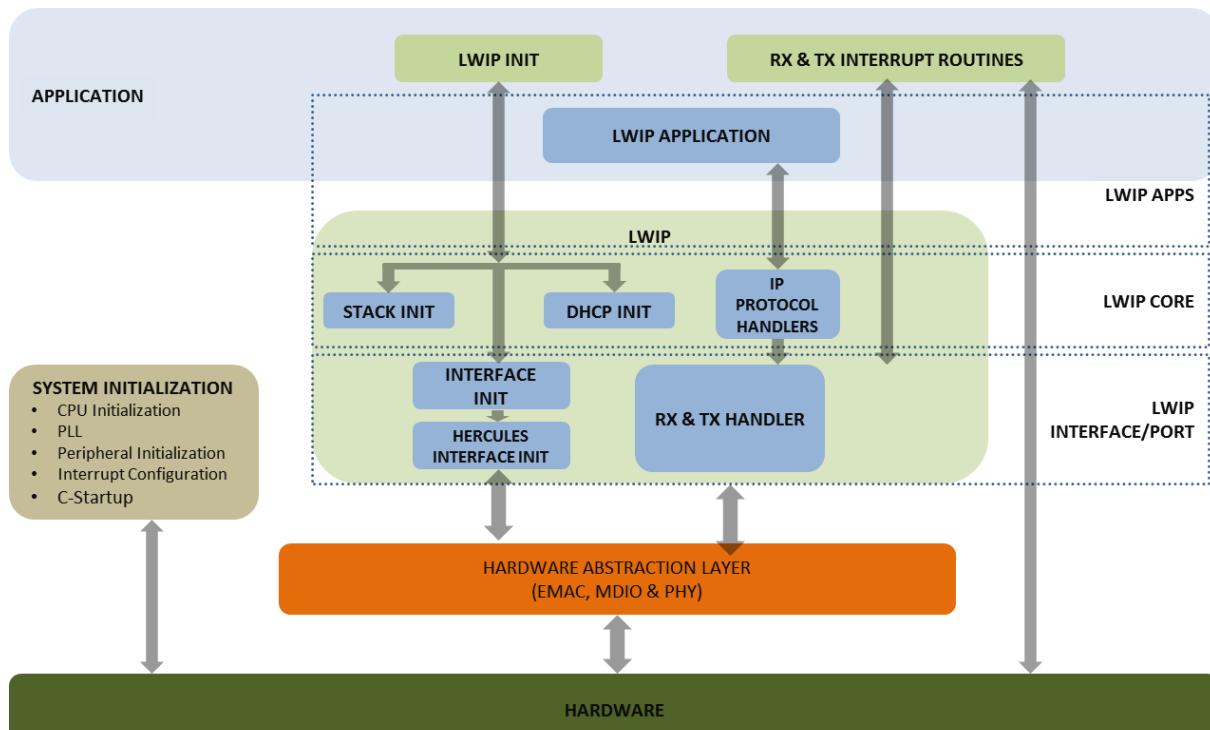
De communicatie tussen de gebruiksvriendelijke bediening en de computersoftware zal



Figuur 7.3.1: Ethernet routine

over ethernet gaan. Hierbij wordt vanaf de interface als eerste een DHCPdiscovery gedaan. Dit is een aanvraag naar de router om instellingen te verkrijgen. Dit wordt gedaan om zonder handmatige instellingen tocheen IPadres te krijgen. Dit adres wordt gebruikt om vanaf de computersoftware te kunnen verbinden met de interface. Na de DHCPdiscovery zal de interface een TCP-server opzetten om de binnenkomende verbindingen te kunnen accepteren. De computersoftware kan nu mits deze ook verbonden is met hetzelfdenetwerkverbinding maken met de interface. Er wordt eerst door de gebruiker een IP-adres ingevoerd waarna de TCP-verbinding wordt gemaakt. Er kunnen nu pakketten van en naar de interface worden gestuurd. Dit gebeurt volgens een ontworpen protocol boven op de TCP laag. Dit protocol is te vinden in de bijlage. In Figuur 7.3.1: Ethernet routine is de flowchart van de ethernet communicatie te zien. In de software wordt gebruik gemaakt van de LWIP-driver. Dit is een lichtgewicht driver die de basis functionaliteit van het ethernet op zich neemt. Er is gekozen voor LWIP omdat het een goed ondersteunde driver is. De LWIP-driver zal tussen de MAC en de applicatiesoftware gaan staan. De MAC bestaat uit een RMII

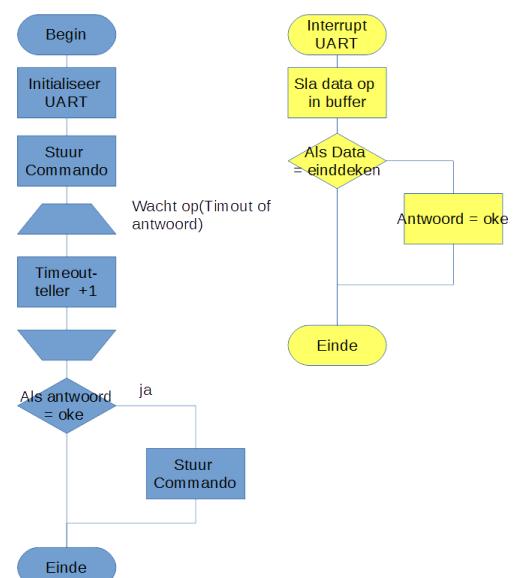
deze bevindt zich bij de gekozen processor in de chip. Dit is een stuk hardware dat de ethernet signalen van dePHY chip afhandelt. Dit om zo de binnen gekomen data en uitgaande data snel te kunnen verwerken. In



Figuur 7.3.2: LWIP software overzicht

7.4 RS232 communicatie

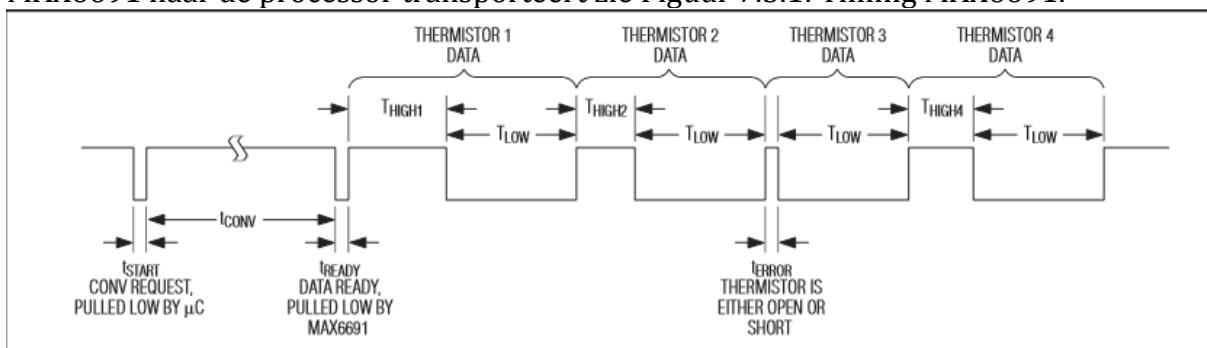
Om in het project te kunnen communiceren met de klimaatkast is RS232 communicatie nodig. Via deze communicatie zal de temperatuur in de kast worden afgelezen, temperatuur worden ingesteld en temperatuurstijging worden ingesteld. De communicatie wordt over een UART bus vanaf de processor omgezet naar RS232 signaal. De communicatie werkt met een meester en slaaf principe. Hierbij moet de meester in dit geval de interface altijd beginnen met het sturen van een aanvraag. Hierop zal de klimaatkast reageren. Er kunnen dus geen signalen uit de klimaatkast komen zonder aanvraag hiervoor. Als voorbeeld de aanvraag van de huidige temperatuur is de klimaatkast. Alvorens er kan worden begonnen wordt een initialisatie gedaan van de UART. Vervolgens wordt er vanuit de interface het commandolees de klimaatkast uit klaargezet. Hierna zal de data worden verstuurd door deze byte per byte klaar te zetten in het verzendt register. Voor het plaatsen wordt er eerst gekeken of de transmissie van de vorige byte gereed is. Vanuit het register zal de hardware in de processor de verdere transmissie doen. Op het moment dat het comando is verstuurd zal de klimaatkast antwoorden met het antwoord. Hier wordt een tijd op gewacht door de interface. Het antwoord wordt byte voor byte verstuurd door de klimaatkast. Bij iedere byte wordt een interrupt gegenereerd in de software. In deze interrupt routine zal het ingekomen byte worden opgeslagen in een buffer. Als het byte het eindaanduiding teken bevat zal de wachtlus worden doorbroken in de interface en zal de data verder worden verwerkt.



Figuur 7.4.1: UART Commando routine

7.5 Temperatuur processing

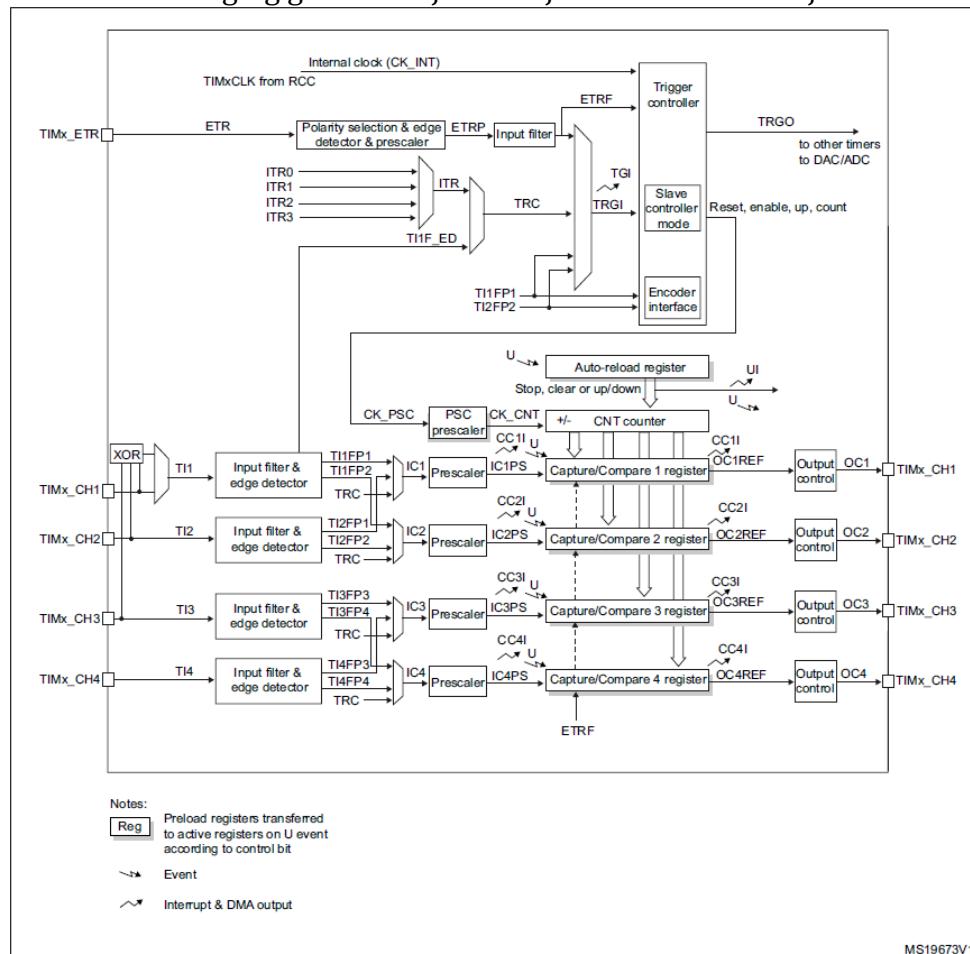
Om de temperaturen te bepalen is het nodig om de hoog en laag tijden te meten die de MAX6691 naar de processor transporteert zie Figuur 7.5.1: Timing MAX6691.



Figuur 7.5.1: Timing MAX6691

Om deze waarden zo precies mogelijk te bepalen wordt er gebruik gemaakt van een timer zie Figuur 7.5.2: Schema Timer 4. De timer heeft vier kanalen. Ieder kanaal heeft een opslag en vergelijkingsregister (CC1 - 4). Dit kan doormiddel van een verandering worden geladen. In het ontwerp komen de pulsen binnen op timer 4 kanaal 4. Hierdoor kan het signaal nu ook schakelen naar kanaal 3. Hierdoor kan op de hoge flank van een

puls het ene register gevult worden en bij een lage flank het andere. Zo ontstaat er geen software vertraging genaamd “jitter” bij het lezen van de tijden van de pulsen.



Figuur 7.5.2: Schema Timer 4

Na het ontvangen van de hoog- en laagtijden van het IC kan de weerstand en vervolgens de temperatuur worden bepaald. Om de berekening te doen voor de NTC-weerstand wordt er eerst een vervangingschema van het circuit opgesteld. Dit is te zien in Figuur 7.5.3: Vervangingsschema weerstandsmeting. Samen met de verkregen formule uit het datablad kan de weerstandswaarde worden bepaald.

Formule MAX6691:

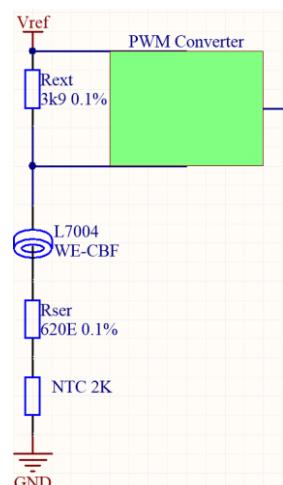
$$\frac{T_H}{T_L} = \frac{V_{ext}}{V_{ref}} - 0.0002 = \frac{R_{ext}}{R_{ext} + R_{th}} - 0.0002$$

Hieruit kunnen de volgende formule opmaken om de weerstand te bepalen:

$$R_{th} = \frac{R_{ext}}{\frac{T_H}{T_L} + 0.0002} - R_{ext} - R_{ser} - R_{fer} = \dots \Omega$$

Als de weerstand waarde en de NTC bekend is kan de temperatuur worden bepaald met de volgende formule:

$$Temp = \frac{1}{\ln\left(\frac{R_{th}}{R_{25}}\right) + \frac{1}{B_{25}}} - 273.15 = \dots ^\circ C$$



Figuur 7.5.3:
Vervangingsschema
weerstandsmeting

De waarde van R25, B25 en T25 zijn per NTC verschillend door deze instelbaar te maken is het mogelijk om meerde temperatuur sensoren te gebruiken.

7.6 Gegevens logging

Om ervoor te zorgen dat de log data bij uitval van de voeding toch bewaard blijft is er een geheugen. Dit geheugen bevat een aantal data die te zien zijn in Tabel 7.6.1: Geheugen indeling.

Tabel 7.6.1: Geheugen indeling

Instellingen
Datum en Modus
Data

In het geheugen wordt aan het begin van iedere file een header gemaakt met daarin de instellingen, datum en modus van het programma. Dit wordt gedaan om makkelijk te kunnen zien welke meting het is. Hierna volgt de data die elke seconde wordt aangevuld door een dataregel. Dit alles gebeurt in het bestandsformaat: ".CSV". Hierbij worden waarden gescheiden door een puntkomma.

De datapakketten worden opgeslagen in een format dat weergegeven is in Tabel 7.6.2: Pakket indeling.

Tabel 7.6.2: Pakket indeling

Nummer:	Waarde	Formaat
1	Tijd	HH:MM:SS
2	Cabinet Temperatuur	XXXX.X °C
3	Sensor 1	XXXX.X °C
4	Sensor 2	XXXX.X °C
5	Sensor 3	XXXX.X °C
6	Sensor 4	XXXX.X °C
7	Status input / relais	0bxxxxxxxx

Om een inschatting te doen hoeveel geheugen er nodig is, is er een berekening gedaan. Hierbij wordt uitgegaan van een dag logging. Dit omdat de klant niet langer als een dag een meting uitvoert.

$$\text{Totale Grote} \approx 24\text{uur} * 60\text{min} * 60\text{sec} * 55\text{Byte} \approx 4752000\text{ Byte}$$

$$\text{Totale grote} \approx 5\text{MByte}$$

Geheugen type

Het opslaan van de gemeten punten kan niet in het interne geheugen van de processor omdat dit geheugen te klein is. Hiervoor is gekozen om de data op een SD-flash geheugen op te slaan. Dit is gekozen om bij uitval de data ook via een computer kan worden makkelijk kan worden verkregen. Hierdoor is het eenvoudig te gebruiken. Voor het opslaan wordt de driver FATFS gebruikt. Dit omdat deze gratis te gebruiken is en het FAT-systeem ondersteund en hierdoor beschikbaar is op vrijwel ieder apparaat.

8 Computersoftware Ontwerp

De computersoftware is de aansturing voor de interface. Hier kan de meting worden ingesteld, temperatuurprofielen worden gemaakt en alarm waarden worden ingesteld. De computersoftware communiceert over een TCP-verbinding met interface.

8.1 Grafische interface



Figuur 8.1.1: Computer software

De grafische interface is het beeld wat de gebruiker ziet met daarbij de mogelijkheden die de gebruiker krijgt om deze te beïnvloeden. In Figuur 8.1.1: Computer software is de gebruikersinterface van het product te zien. Hierbij zijn er twee mogelijkheden profiel en manueel. Bij manueel kan de gebruiker op ieder moment de temperatuur wijzigen na behoefte. Bij profiel setup kan de gebruiker vooraf met behulp van de muis het temperatuurprofiel tekenen die de klimaatkast zal doorlopen. Dit om zo zonder extra acties in een keer een gehele test te laten draaien. Bij beide modussen kunnen van tevoren alarmen en in worden gesteld. Hierbij kan de waarde waarboven het alarm optreedt, een waarschuwing geven en de digitale uitgang schakelen. Wanneer men de instellingen heeft gedaan kan men beginnen door op start te klikken. Vervolgens worden via ethernet de instellingen naar de interface verstuurd. Hierna zal deze om de seconde een data punt versturen. Als men klaar is met meten kan er op stop worden gedrukt de module stopt dan zijn programma en het sturen van data punten. Vervolgens is het mogelijk om de data op te slaan. Hierbij worden alle punten in de grafiek opgeslagen in een ".CSV" bestand. Daarin worden de waarden opgeslagen in een tabelvorm waarbij een puntkomma de scheiding tussen de vakken is. Wanneer het systeem wordt afgesloten wanneer er een actieve meting gaande is zal de applicatie verdwijnen naar het systeemvak en niet afsluiten. Zo kan de gebruiker de computer verder gebruiken met het systeem op de achtergrond. Mocht een alarm zich voordoen zal deze weer naar voren komen.

8.2 Alarm meldingen

Als een profiel setup of een manual setup wordt gestart kunnen er alarmen en relais worden geschakeld. Dit gebeurt als een sensor boven de ingestelde temperatuur komt. Dit zal dan wanneer ingesteld een alarm geven in de form van een rood knipperend scherm in beeld. De gebruiker kan op dit moment op het scherm klikken. Hierna zal het hoofd venster weer naar voren komen. Hier is dan te zien welke alarmen zijn geactiveerd en welke relais zijn geschakeld. Deze melding heeft als doel de gebruiker direct te waarschuwen om zo te voorkomen dat er eventuele schade ontstaat. Dit kan worden gerealiseerd doordat de gebruiker op het schakelen van de digitale uitgang zijn process kan onderbreken. Denk hierbij aan het uitschakelen van de voeding van het geteste product.



Figuur 8.2.1: Alarm melding

8.3 Ethernet communicatie

Voor de ethernet communicatie wordt een aparte thread aangemaakt. Dit is net zoals bij de RTOS een taak die naast elkaar kan worden uit worden gevoert. Hierin kan los van het programma gewerkt worden om zo het programma niet te blokkeren. In deze thread bevindt een lus waarin wordt gekeken of er data klaar staat om te verzenden. Deze data staat in gedeelde variabelen waar vanuit de applicatie data in kan worden gezet. Deze kan dan zodra de ethernet thread er klaar voor is worden verstuurd. Ook wordt er gekeken of er data klaar staat om te ontvangen. Als dit het geval is zal de data worden opgeslagen en zal er een event worden geactiveerd. Dit event zal vervolgens in de applicatie een functie aanroepen. Hierin kan de ingekomen data worden verwerkt en de benodigde acties worden uitgevoerd. Dit alles werk volgens het protocol wat in de bijlage is bijgesloten. Hierin staan alle functies van en naar de interface beschreven.

9 Controle

Om ervoor te zorgen dat het product aan de gestelde eisen voldoet worden er verschillende testen gedaan. Uit deze testen kunnen dan conclusies, aanbevelingen en aanpassingen komen. De complete testen zijn te vinden in de bijlage. In dit hoofdstuk worden de resultaten besproken.

9.1 FMEA

In een FMEA wordt gekeken waar de zwakke punten van het product liggen. Hierbij wordt vooral de nadruk gelegd op de veiligheid risico's en de klant tevredenheid. Uit De FMEA van dit product kwam over het algemeen een goed beeld met één aandachtspunt namelijk de voedingsbuffer. Hierbij was de score hoog doordat het resultaat van het falen van de condensators de voedingsbuffercapaciteit het meetresultaat aantastte en dit niet direct door de klant kon worden ontdekt. Om dit mogelijke probleem op te lossen moeten er hoge kwaliteits condensatoren gebruikt worden en moet er een periodieke controle plaatsvinden om te bepalen of de interface nog de correcte temperaturen meet.

9.2 FAT

De FAT wordt gedaan om het product te controleren op zijn eigenschappen voordat het de productielocatie verlaat. Hierbij wordt het product getest op de functionele en technische vereisten. Uit de FAT test voor de klimaatkast zijn een aantal punten gekomen.

- Logpunten hierbij zou er bij ieder log punt een datum en tijd mee worden gestuurd. Om data te sparen is dit verandert na alleen de tijd per log punt. Wel wordt de datum boven het bestand geschreven zodat de datum toch bekend is.
- NTC-weerstand om de gehele temperatuur range te halen kan niet worden gewerkt met een 10kohm NTC. Er moet hiervoor een 2kohm NTC worden gebruikt.
- De verversingssnelheid is niet instelbaar. Dit omdat het niet mogelijk is om sneller te meten met het temperatuur IC.

Deze punten zijn besproken en de klant is hiermee akkoord gegaan. Wel wordt het meegenomen in de aanbevelingen.

9.3 SAT

De SAT test wordt gedaan om met de klant te kijken of het product voldoet aan zijn eisen. Hierbij ligt de nadruk op de klant zijn eisen. Uit SAT samen met de klant van het product kwam dat het product voldeed aan de klant zijn eisen. Hierbij kwamen nog een paar kleine software bugs naar voren. Deze werden direct verholpen. Verder werkte het product zoals de klant dit verwacht.

10 Conclusies en aanbevelingen

10.1 Conclusies

Het probleem van de slechte gebruikersinterface is opgelost. Er is een gebruiksvriendelijke gebruikersinterface gerealiseerd. Hierbij zijn ook mogelijkheden bij gekomen om de functionaliteit te verbeteren. Zo is het mogelijk om vier temperatuur sensoren aan te sluiten, een temperatuur log te maken, het monitoren van digitale ingangen, het maken van temperatuurtestprofielen en het activeren van digitale uitgangen op temperatuur. Dit alles kan worden ingesteld via een applicatie op de computer. Deze applicatie communiceert via ethernet met de interface waardoor het binnen Brunelco bereikbaar is.

10.2 Aanbevelingen

Om de gebruiksvriendelijkheid te verhogen is het aan te bevelen om de volgende punten te verbeteren.

UDP server

Op dit moment moet de gebruiker bij het maken van de verbinding het IP-adres van de module oplezen van het scherm en invoeren in de pc applicatie. Het is mogelijk om dit automatisch te laten verlopen doormiddel van een UDP-server. Bij dit ethernet protocol kan vanaf de pc-applicatie een pakket worden verstuurd naar alle gebruikers in het netwerk doormiddel van een broadcast. De interface kan dan reageren om er zo voor te zorgen dat de applicatie weet welk IP-adres de interface heeft. Dit is binnen dit project niet gerealiseerd dit niet haalbaar was binnen de tijd en hierbij andere functionaliteit meer prioritair had.

Temperatuur meting

Voor de temperatuurmeting wordt op dit moment een conversie IC gebruikt. Dit IC is niet instaat om alle weerstandswaarden te converteren. Hierdoor is het niet mogelijk om alle NTC types te gebruiken. Dit zou kunnen worden verbeterd een eigen ontworpen schakeling met ADC en een Gain control of een complexer IC zoals de eerder onderzochte LT2983.

Referenties

Media access control. (2015, 18 september). In Wikipedia. Geraadpleegd op 6 oktober, 2015, van https://en.wikipedia.org/wiki/Media_access_control

Media-independent interface. (2015, 17 augustus). In Wikipedia. Geraadpleegd op 7 oktober, 2015, van https://en.wikipedia.org/wiki/Media-independent_interface

Serial Peripheral Interface Bus. (2015, 3 oktober). In Wikipedia. Geraadpleegd op 13 oktober, 2015, van https://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus

Four-Channel Thermistor Temperature-to-Pulse-WidthConverter. (2007, Juli). In Farnell. Geraadpleegd op 16 oktober, 2015, van <https://datasheets.maximintegrated.com/en/ds/MAX6691.pdf>

STM32F75xxx Refence manual. (2015, december). In ST. Geraadpleegd op 21 oktober, 2015, van http://www.st.com/st-web-ui/static/active/en/resource/technical/document/reference_manual/DM00124865.pdf

Failure mode and effects analysis. (2015, 18 september). In Wikipedia. Geraadpleegd op 29 oktober, 2015, van https://nl.wikipedia.org/wiki/Failure_mode_and_effects_analysis

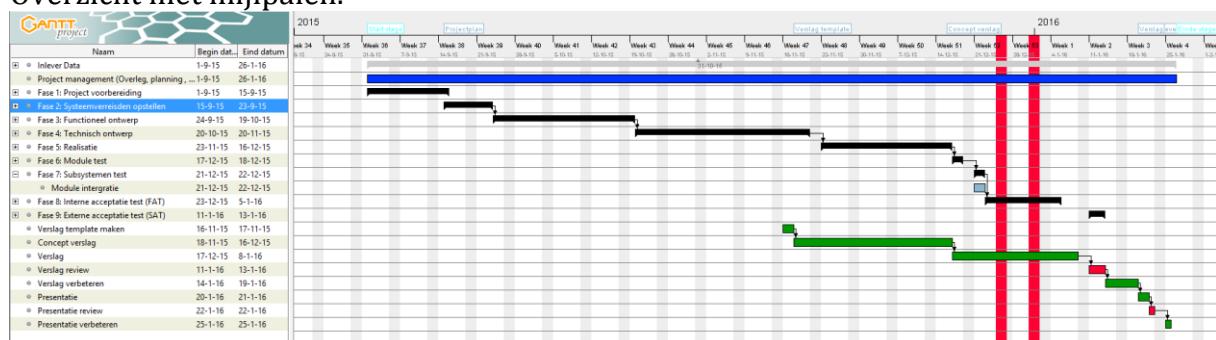
Kernal. (2014, 4 juli). In Wikipedia. Geraadpleegd op 11 november, 2015, van <https://nl.wikipedia.org/wiki/Kernal>

FreeRTOS Quick Start Guide. In FreeRTOS. Geraadpleegd op 4 december, 2015 van <http://www.freertos.org/FreeRTOS-quick-start-guide.html>

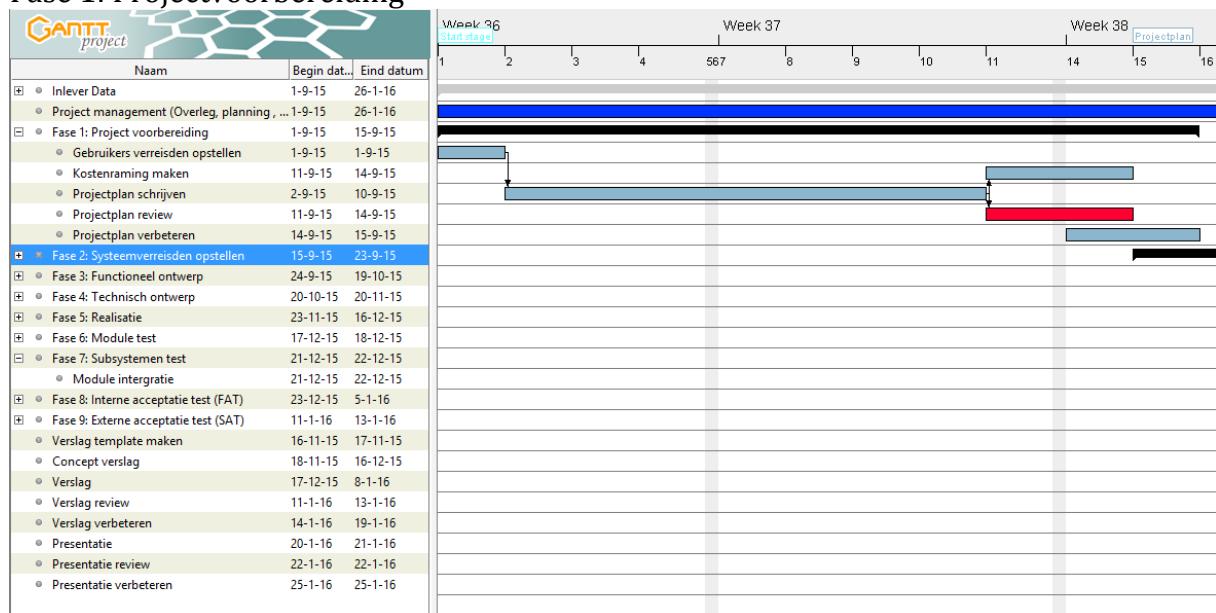
Description of STM32F4xx HAL drivers. (2015, september). In ST. Geraadpleegd op 5 december, 2015, van http://www.st.com/st-web-ui/static/active/jp/resource/technical/document/user_manual/DM00105879.pdf

Bijlage 1: Planning

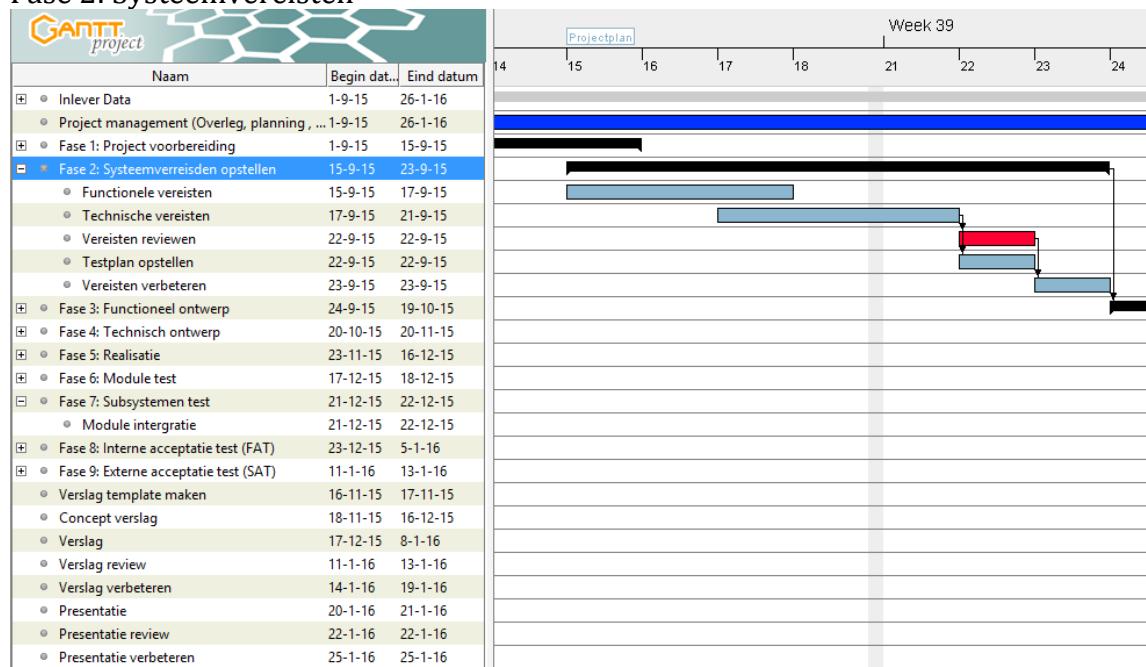
Overzicht met mijlpalen.



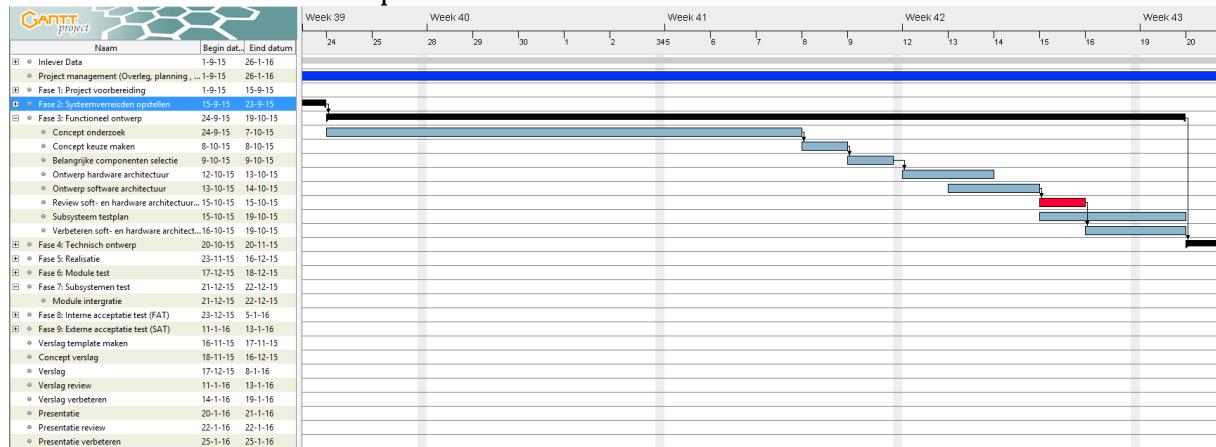
Fase 1: Projectvoorbereiding



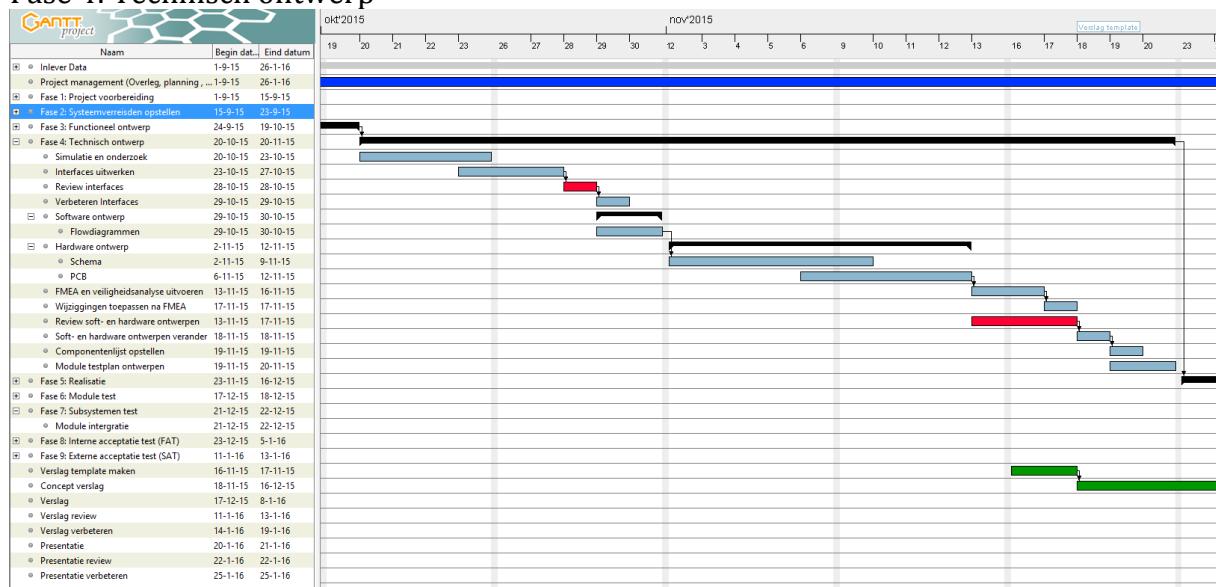
Fase 2: Systeemvereisten



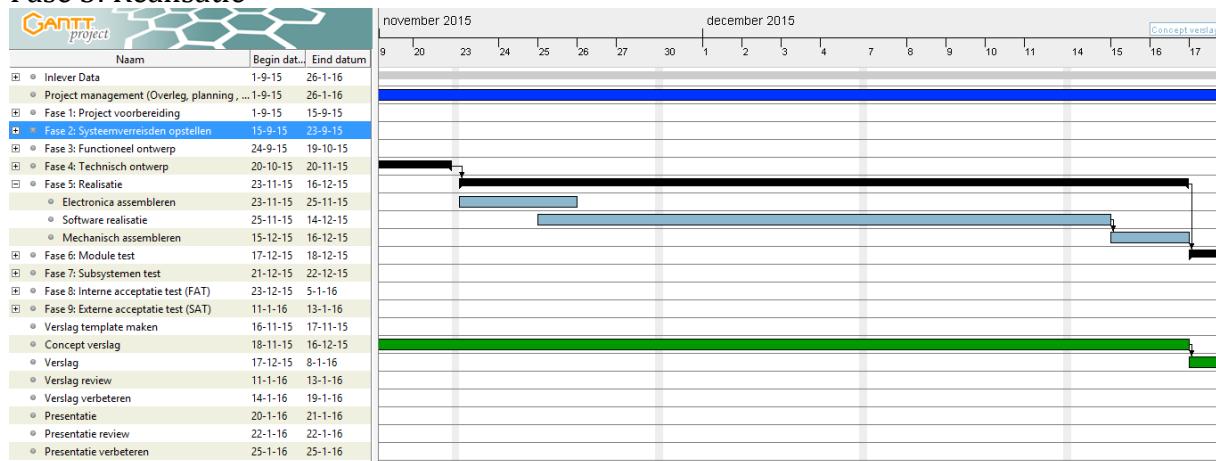
Fase 3: Functioneel ontwerp



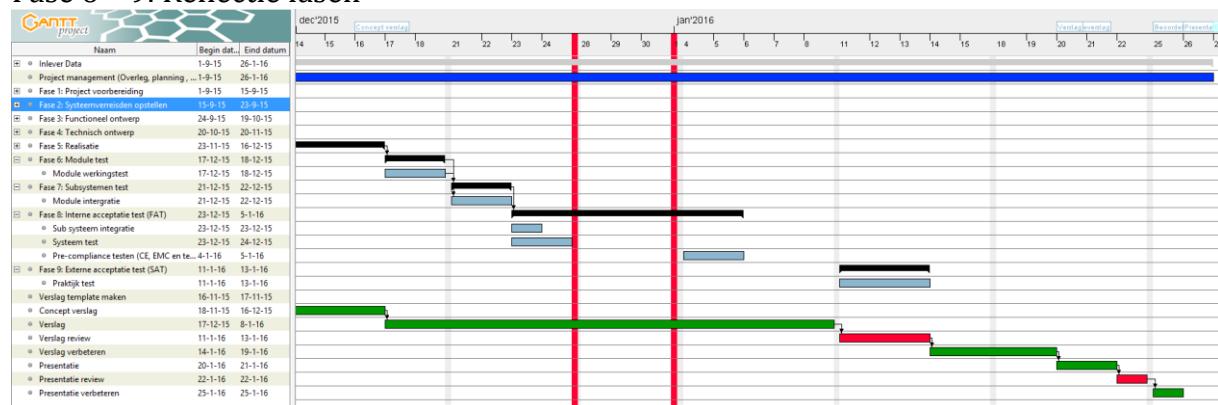
Fase 4: Technisch ontwerp



Fase 5: Realisatie

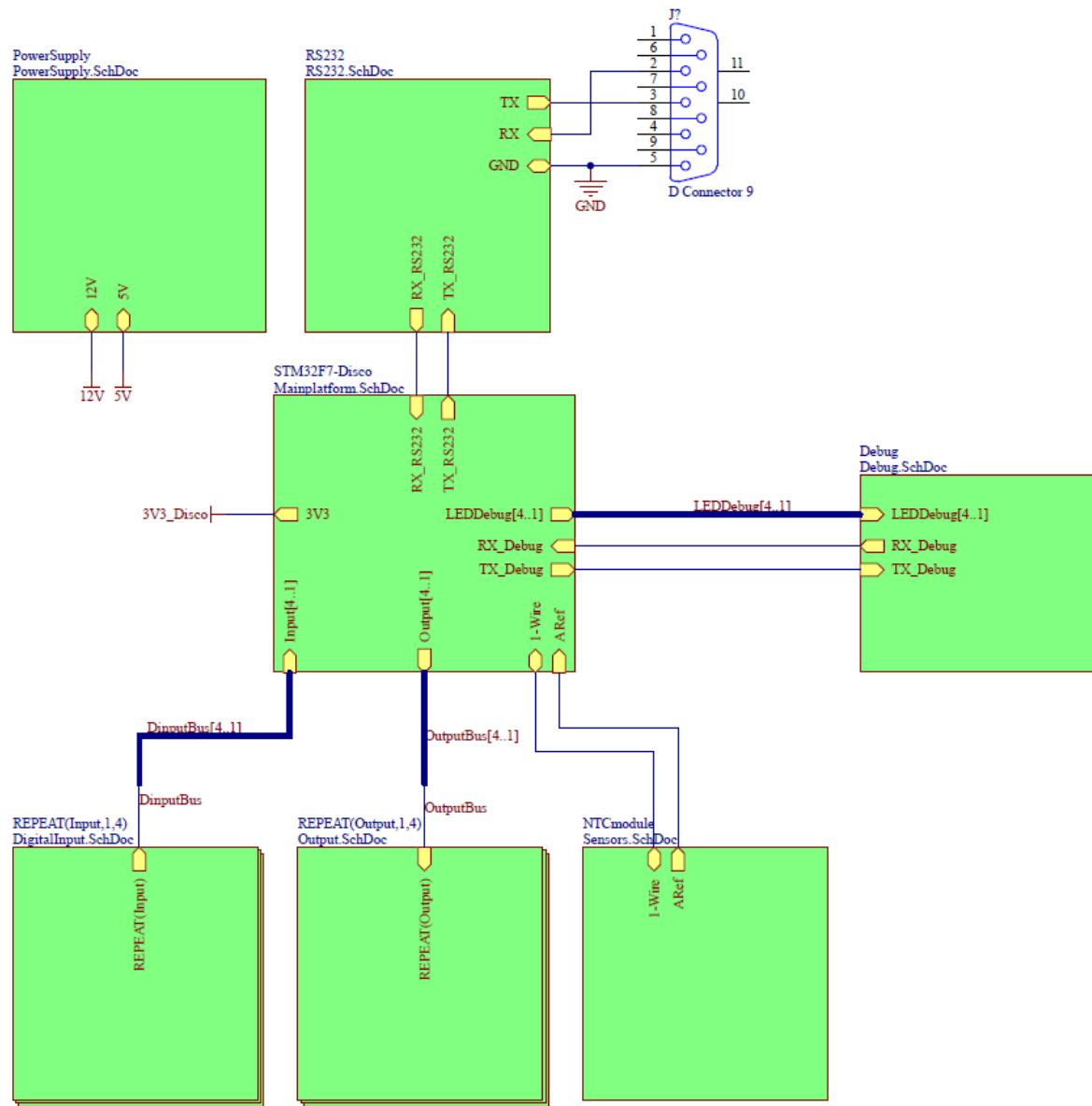


Fase 6 – 9: Reflectie fasen



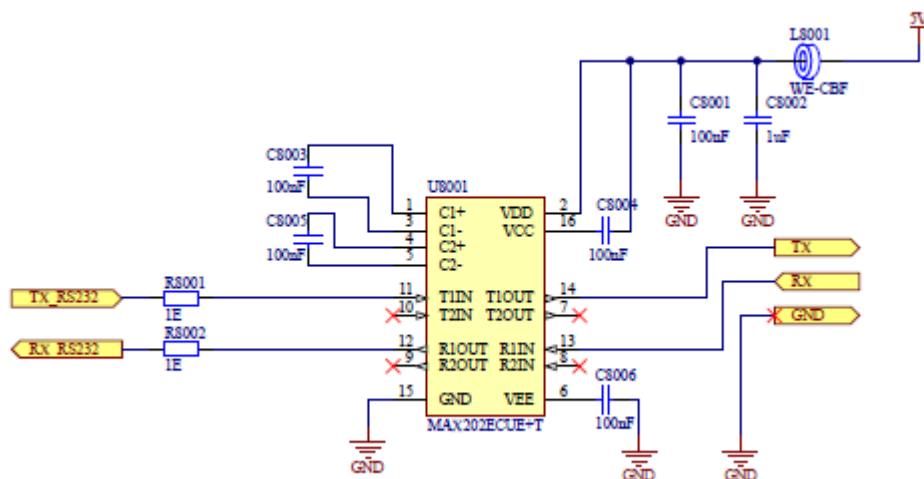
Bijlage 2: Print ontwerp

1. Blockschema

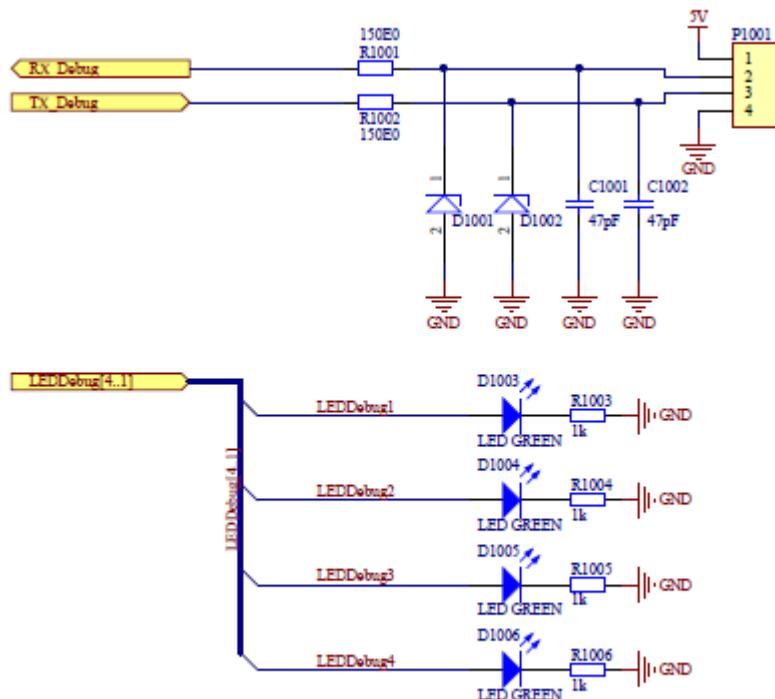


Figuur 1: blockschema

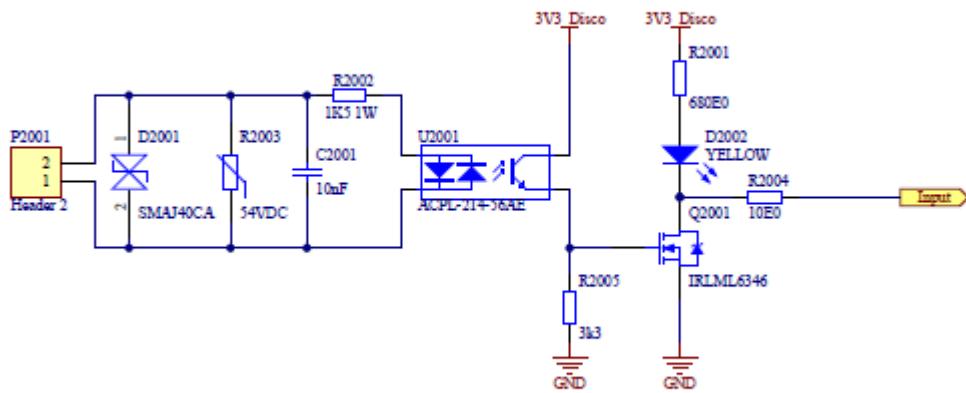
2. Schema's :



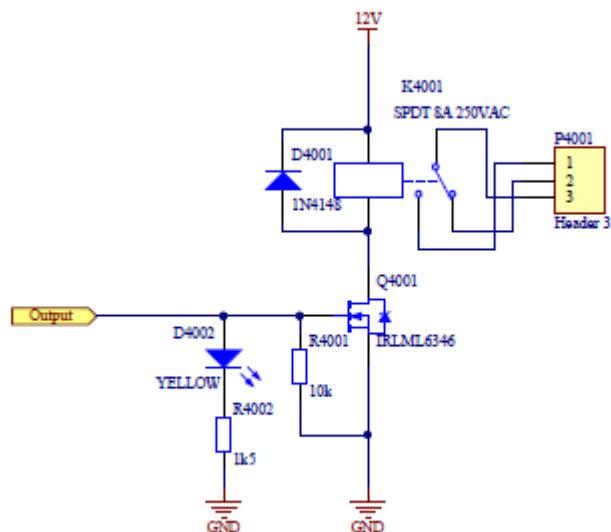
Figuur 2: Schema RS232 conversie



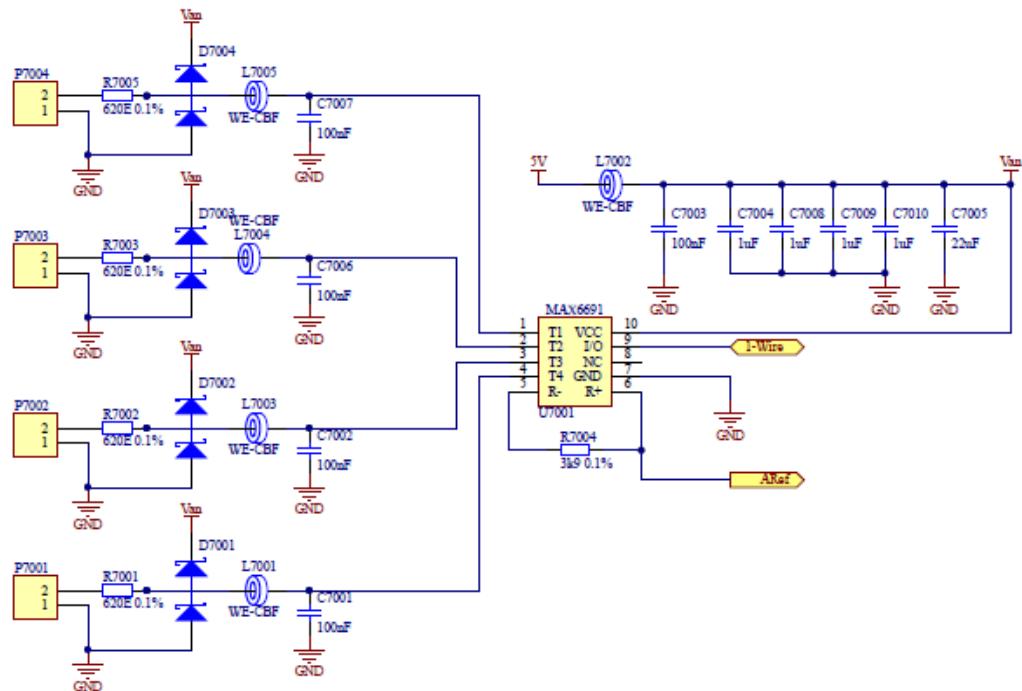
Figuur 3: Schema debug mogelijkheden



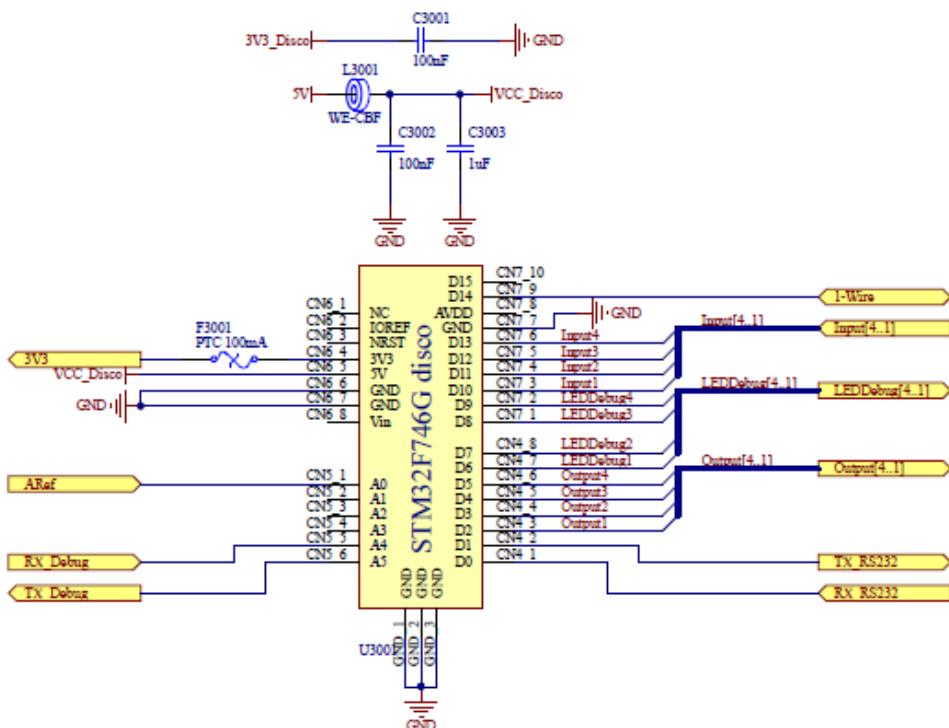
Figuur 4: Schema digitale ingang



Figuur 5: Schema Digitale uitgang

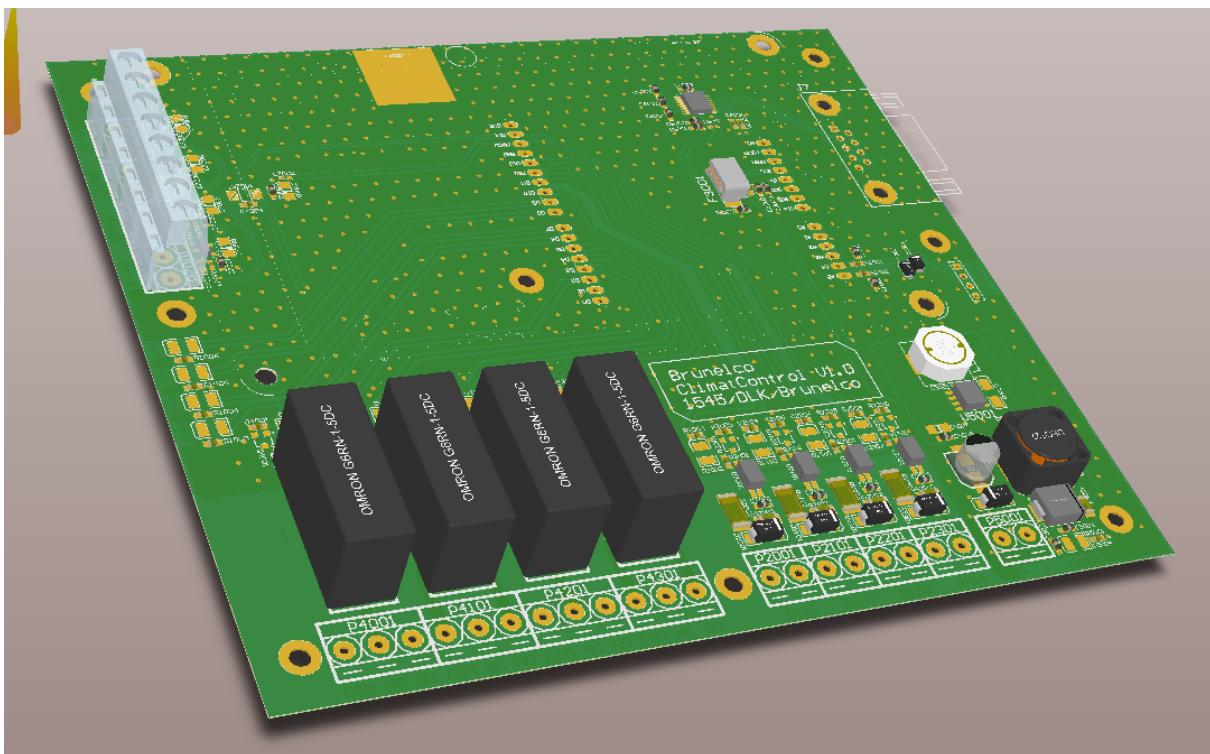


Figuur 6: Schema analoge ingangen

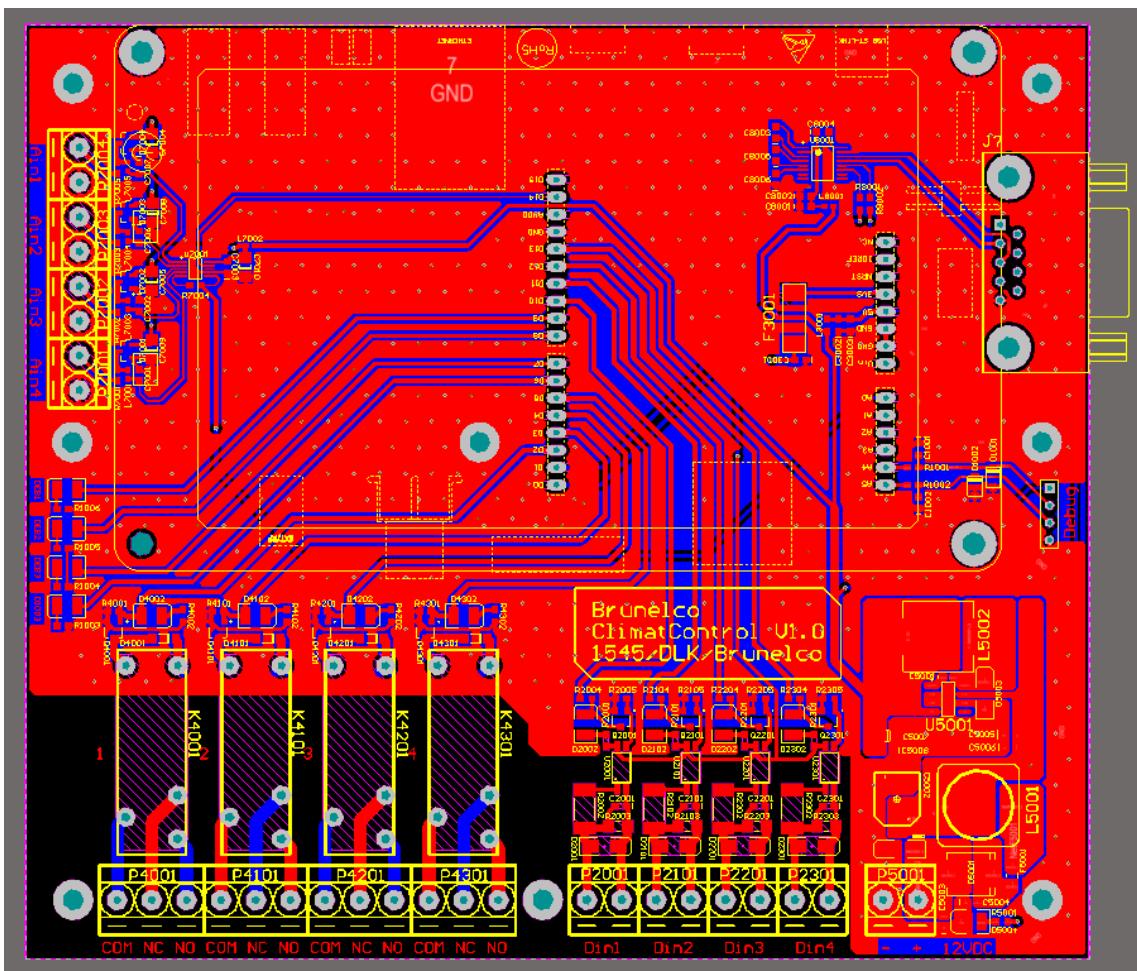


Figuur 7: Schema hoofdplatform

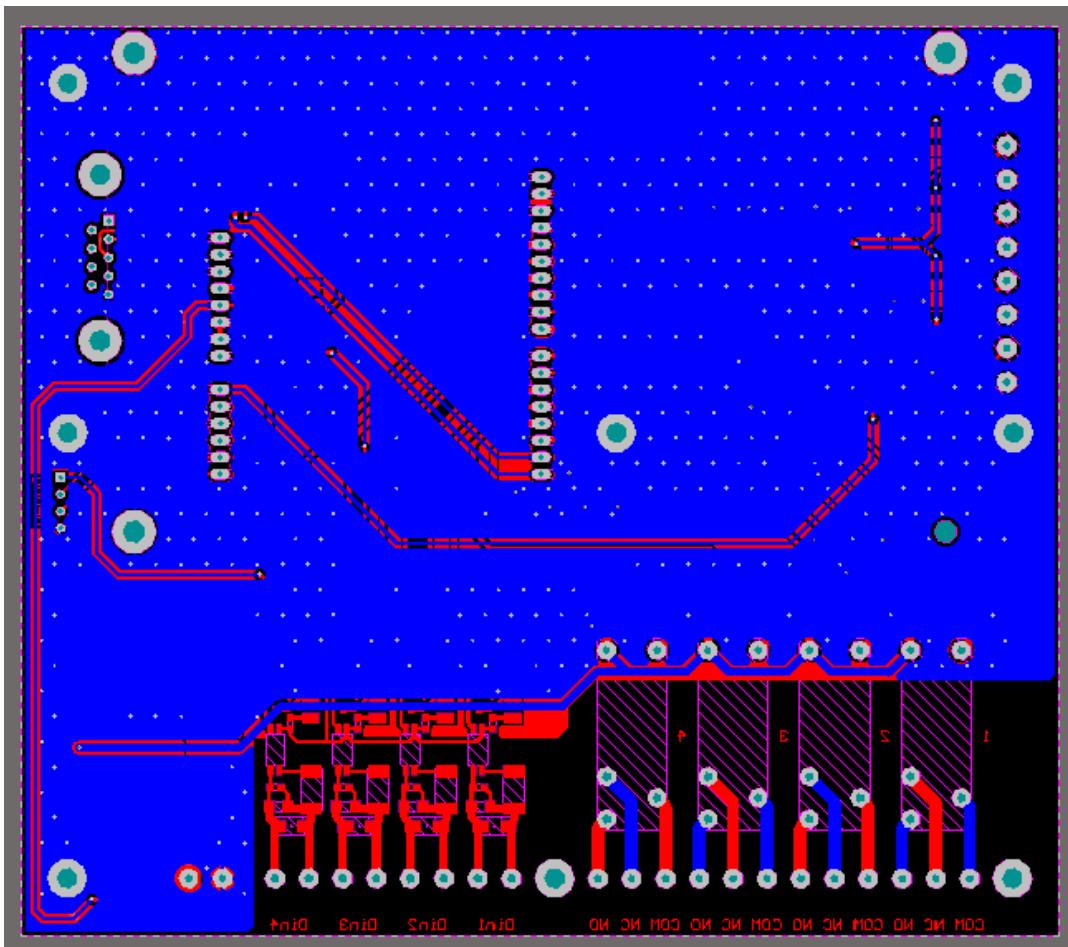
3. PCB



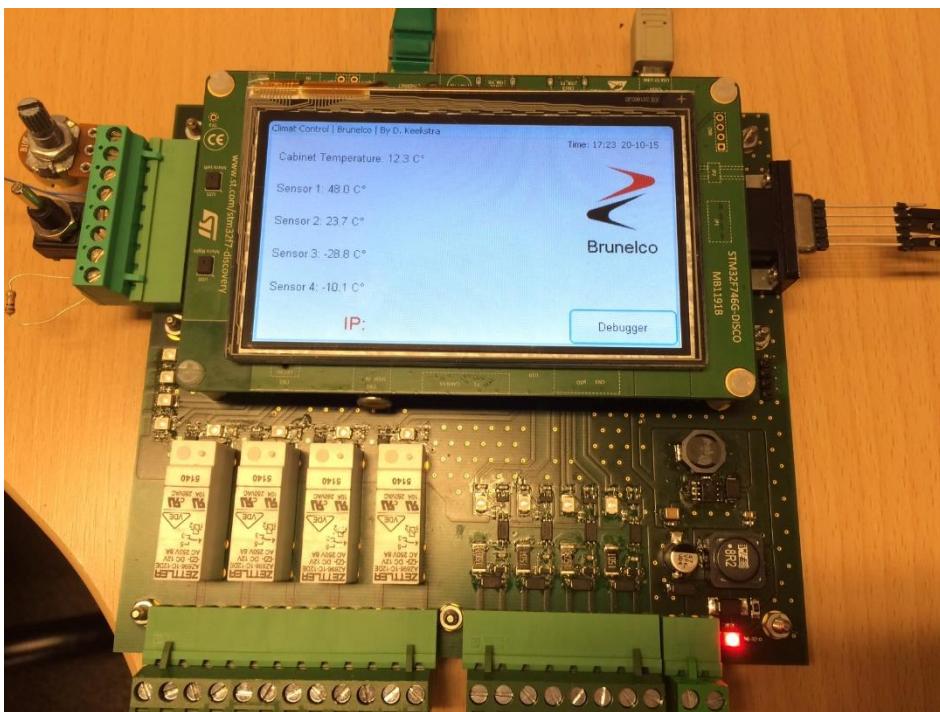
Figuur 8: PCB in 3D



Figuur 9: PCB top laag



Figuur 10: PCB bottom laag



Figuur 11: PCB Bestukt

4. Bill Of Materials

Component list									
Bill of Materials For PCB Document [KlimaatkastBesturing.PcbDoc]									
Zvt Product: ClimateControl									
Source Data From: KlimaatkastBesturing.PcbDoc									
Project: KlimaatkastBesturing.PjtPCB									
Variant: None									
Report Date: 11-11-2015									
Print Date: 16-Nov-15									
15:29:10									
1:36:15 PM									
ELECTRONIC INNOVATORS									
BRUNELCO									
www.brunelco.nl									
#	AnikaElectro Technicom	Part Number	Q'ty Delivered	Comment	Description	Designator	Footprint	Layer	Quantity
1	CCPFP047K0603		470F		SMD V-SOIC 47Pin 50V 5x7 NP010603	C1001, C1002	1608[0603]	Top	2
2	CWMP047K0603		10F		SMD V-SOIC 47Pin 50V 10x7 NP010603	C2001, C201, C201	1608[0603]	Top	4
3	CWMP100K0603	CWMP100K0603	100F		SMD V-SOIC 100Pin 50V 10x7 NP010603, SMD V-SOIC 100Pin 25V	C3001, C3002, C3003	1608[0603]	Top	16
4	EMK075A05K4-TR		10F		SMD Cap CER 10uF 16V X7R 0603	C3003, C3002	1608[0603]	Top	2
5	CLO58103-2-BENNNC		102.651		Cap Ceramic 0810 50V X7R 5x7 SMD 0402	C5001	1608[0402]	Top	1
6	CSMK047K0603		9698890		SMD V-SOIC 47Pin 50V 9005	C5002	NA/CE5.354	Top	1
7	GRM42R9E10K473L		236596		SMD V-SOIC 10F 25V X7R 9005	C5005	2012[0905]	Top	1
8	C1286CA78MPC-ACTU		2370374		CAPACITOR, 47 UF, 10V, 1200, X7R	C5007	3261[1206]	Top	1
9	CWUF001K02X		175312		SMD V-SOIC 10F 25V 10V X7R 1206, SMD	C7004, C7005, C7008	3261[1206]	Top	5
10	PTVS3YSSUR		1822523		PTVS3YSSUR	D1001, D1002	SOD223N	Top	2
11	LGT1679			LED GREEN	TOPLED PLCC2 GREEN	D1003, DEB1, DEB2,	3.5x2.8x1.9	Top	4
12	SMA440CA		2254548	SMA440CA	TOPLED Voltage Suppression Diode, 40V, 400W, Bidirectional	D5001	DO-244C	Top	1
13	LY1679		2392345	LED YELLOW	TOPLED PLCC2 YELLOW	D2002, D202, D202	3.5x2.8x1.9	Top	8
14	LL1448		9543986	LED YELLOW	High Conductance Fast Diode	D4001, D401, D401	MELF-05512-405	Top	4
15	30EQ0010TRBF		165216	SMD 3A 10V	Schottky Rectifier, 3A, 10V	D5001	DO-204AB	Top	1
16	SMA440CA		242074	SMA440CA	Transient Voltage Suppression Diode, 42V, 400W, Bidirectional	D5002	DO-244C	Top	1
17	B340A		1843658	B340A	Schottky Rectifier, 41V, 3A, SMA	D5003	SMA	Top	1
18	LS1679		2392344	LED RED	TOPLED PLCC2 RED	D5004	3.5x2.8x1.9	Top	1
19	BA175MS		108114	BA175MS	Schottky Barrier Diode	D7001, D7002, D7003	SOT23 N	SOT23 N	1
20	HEADER4		*	Header 4	Header, 4-pin, 5.0mm	D801	HDR1X4	Top	1
21	MF-SM075		1651787	MF-SM075	MF-SM Series - PTC Resettable Fuses	F3001	MF-SM075	Top	1
22	FUSE 106, 50 2A		1857478	2A	Fuse, Slow Blow 2A	F5001	3261[206]	Top	1
23	TE-HFS3-384-2		*	SubD9	Sub-D 9-pins, PCB angle	J1	SUB-D9.3, Female Angle	Top	1
24	A2386IC-24DE		1891734	*	Relay SPDT	K4001, K4101, K4201	ZETTLER A2386	Top	4
25	FEER30		*	FEER30	SPDT Relay RY61002 8A, 250V AC/DC	L3001	1608[0303]	Top	1
26	444771008		1635905	8.20H 6.25A	Inductor 8.20H 6.25A	L5001	WE-PI-0001	Top	1
27	744406681		1635893	680uH 450mA	POWER-CHOKE WE-TFC 74406680	L5002	WE-TFC 1028-XLH	Top	1
28	442782730		*	WE-CGF 0402	WE-CGF SMD EMI Suppression Ferrite, 220E @ 100MHz, Rad. =	L7001, L7002, L7003	1608[0402]	Top	6
29	CLL508-2		*	Header 2	Header 2-Pin, 0.8mm	CLL508-2	CLL508-2	Top	4
30	CMK030P025		*	Connector 5.08mm 3p	Male connector 3-Pin right pitch 5.08mm 50HS	P4001, P4101, P4201	PKS3-018	Top	4
31	PR2-5108		*	Header 2	Header 2-Pin, 0.8mm	P5001	PKS2-5108	Top	1
32	TYCO 1-282841-2		*	Header 2	Header 2-Pin, 0.8mm	P7001, P7002, P7003	TYCO 1-282841-2	Top	4
33	FLML546TRBF		1857300	*	HEXFET Power MOSFET, 1N-Ch, 30V, 3.4A	P7001	TYCO 1-282841-2	Top	8
34	CR60603B104-1CR60603B104		2284410	10.0% 0603	Resistor 0603, Resistor 0603, Resistor 0603 10%, 25ppm,	P7001, P7002, P7003	1608[0603]	Top	34
35	ER-MTM15F5MU		1283124	1.5 Ohm 1W	1.5 Ohm 1W SMD	P7002, P7002	6332[2121]	Top	4
36	CT1208K41G		1634953	Vanson 40W RMS 200V SMD	P7003, P7003, P7003	3261[206]	Top	4	
37	ACPL-24-864E		*	ACPL-24-864E	AC Input - Hall-Effect Phototransistor Optocoupler, CTR > 5%	U7001, U7001, U7001	1608[0303]	Top	4
38	STM32F46G discovered		*	STM32F46G discovered	STM32F46G discovered	U7001	STM32F46G discovered	Top	1
39	LM62675M-3.3		9488900	LM62675M-3.3	Opamp, N	U7001	LM62675M-3.3	Top	1
40	MAX46891		Mouse, 200, MAX46891MIBT	four-channel thermistor temperature-d.c.	MAX46891MIBT	U7001	UMAX46891	Top	1
41	MAX232AESE		2335478	MAX232AESE	MAX232AESE	U7001	TSSOP16-N	Top	1

Approved

Notes

Figuur 13: Component kosten calculatie

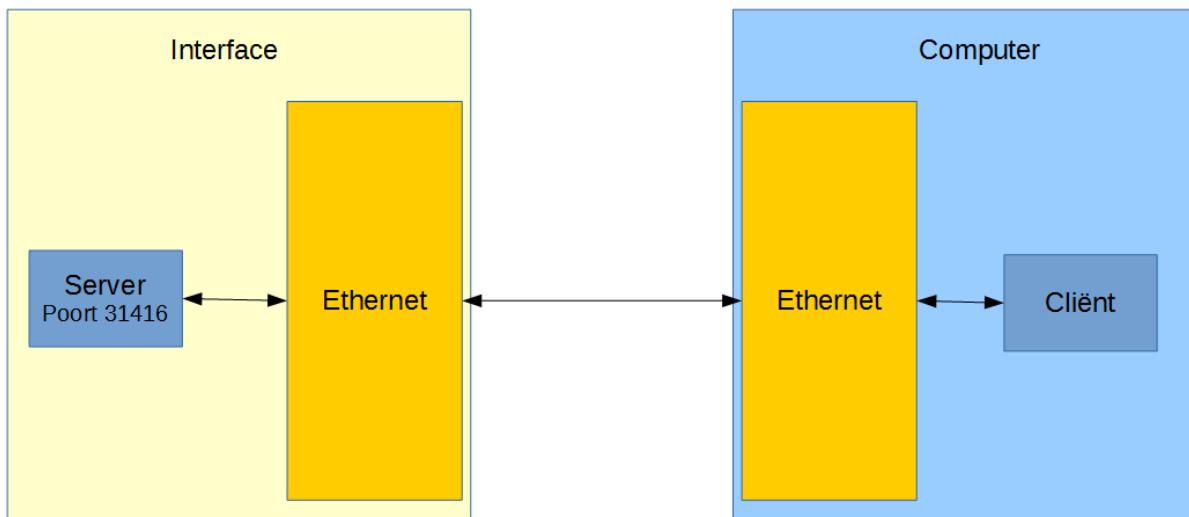
5. Component kosten calculatie

Beschrijving	Farnell nr	Aantal	Prijs	Totaal
ACPL-214-50AE	1602611	5	€ 0,89	€ 4,47
Capacitor 47µF 35V EEEFK1V470P	9695850	1	€ 0,29	€ 0,29
Capacitor 10µF 25V CER	2346927	1	€ 0,75	€ 0,75
Capacitor 47uF 16V CER	2407351	1	€ 1,51	€ 1,51
Capacitor 1uF 25V CER	9227865	5	€ 0,07	€ 0,35
SMAJ40CA 40V Tranzorb	1827675	4	€ 0,35	€ 1,42
SMAJ12CA 12V Tranzorb	2429074	1	€ 0,26	€ 0,26
BAT54S diode dual	1081194	5	€ 0,05	€ 0,23
MULTIFUSE, SMD, 0.3A	9350195	5	€ 0,80	€ 4,02
TR/3216FF2R FUSE, SMD, 2A, 1206	1155748	4	€ 0,91	€ 3,64
SCHRACK RT114012 SPDT Relay	1629036	4	€ 2,55	€ 10,20
SMD induktor 744771008 8.2 µH, ± 20%	1635905	1	€ 2,16	€ 2,16
SMD induktor 744066681 680 µH, ± 30%	1635899	1	€ 1,84	€ 1,84
FERRITE, BEAD, 0402, 0.49OHM, 0.35A	2443221	10	€ 0,01	€ 0,13
MostFET N Channel, 3.4 A, 30 V, 0.046 ohm	1857300	10	€ 0,32	€ 3,17
3k9 0,1% 0603	1506012	2	€ 0,30	€ 0,60
3k24 0,1% 0603	2116733	2	€ 0,28	€ 0,55
620Ohm 0,1% 0603	1670173	4	€ 0,24	€ 0,97
1k5 1Watt 2512	1283124	4	€ 0,25	€ 0,99
LM2675M-3.3	9489800	1	€ 2,87	€ 2,87
MAX202IPW TSSOP-16	2335478	2	€ 1,10	€ 2,20
PCB		1	€ 106,00	€ 106,00
Aanwezige componenten				€ 5,00
			Eindtotaal:	€ 153,62

Figuur 13: Component kosten calculatie

Bijlage 3: Protocol beschrijving

1. Data protocol interface – computer software



Figuur 1a: Communicatie tussen interface en computersoftware

Zoals in Figuur 1a te zien is, is er tussen de interface en de computer een ethernet verbinding. Over deze verbinding zal verschillende informatie worden verzonden via het IP/TCP-protocol. Door deze verbinding is het mogelijk om van beide kanten naar elkaar pakketten te sturen. Dit is nodig om vanaf de computersoftware op elk moment te kunnen ingrijpen op de interface en op elk moment vanaf de interface alarmen en datapunten te kunnen versturen. De computer moet zijn gestuurde data controleren. Dit kan door middel van de opvraag functies.

Preamble	Functie	Lengte	Data
Byte 0-1	Byte 1	Byte 2-3	Byte 4-x
0xFFFF	0x00-0xFF	0x0-0xFFFF	-

Figuur 1b: Opbouw van een pakket

In Figuur 1b te zien hoe een pakket uit het protocol is opgebouwd. De volgorde van op elkaar volgende bytes is little endian. Dit betekent dat de hoogste byte als eerste wordt verstuurd.

In het protocol worden temperaturen verstuurd. Deze temperaturen worden beschreven in 2 bytes. De temperaturen bevinden zich tussen de -40 en +130 graden. Hierbij hebben deze temperaturen een stapgrootte van 0.005 graden. Dit houdt in dat een temperatuur van -40 gelijk staat aan een decimale waarde van 0. En een temperatuur van 130 graden gelijk staat aan een waarde van 33999.

Met behulp van IP/TCP wordt de data verstuurt. Deze data heeft zijn eigen protocol voor de communicatie. Het basis protocol bestaat uit een header en een stuk data. Bij het dataprotocol is geen datacontrole nodig omdat deze zich al in het IP/TCP-protocol bevindt.

Functie	Data bytes	Beschrijving	Richting
0x00	4	Voeg setpoint toe	➤ Interface
0x01	5	Stel tijd en datum in	➤ Interface
0x02	1	Vraag data op	➤ Interface
0x03	1	Start/stop programma	➤ Interface
0x04	32	Poll	➤ Interface
0x05	3	Set manuele temperatuur	➤ Interface
0x06	20	Set alarm niveaus	➤ Interface
0x07	0	Echo alarm	➤ Interface
0x08	0	Clear Setpoints	➤ Interface
0x09	1	Reset Relay	➤ Interface
Functie	Data bytes	Beschrijving	Richting
0xF0	4	Stuur alarm	➤ Computer
0xF1	19	Stuur data punt	➤ Computer
0xF2	1	Stuur status	➤ Computer
0xF3	19	Stuur alarmen	➤ Computer
0xF4	0	Poll echo	➤ Computer
0xF5	720	Stuur geheugen block	➤ Computer
0xF6	I x 4 + 1	Stuur programma rij	➤ Computer

Figuur 2: Communicatie functies

1.1 Functies naar interface

Deze functies kunnen worden verstuurd naar de interface.

Functie: Voeg setpoint toe

	Preamble	Functie	Lengte	Duur	Temperatuur	Stijging
Byte	0-1	2	3-4	5	6 - 7	8
Data	0xFFFFF	0x00	0x04	0-240 min	-40 - 130 C°	0.1 - 2.0C°

Figuur 3: Functie voeg setpoint toe

Deze functie zal een setpoint toevoegen aan het proces. Een setpoint bestaat uit een aantal data. Hieronder vallen duur, temperatuur en stijgingssnelheid. Het eerste setpoint in de rij zal worden uitgevoerd als het programma wordt gestart. Er kunnen Maximaal 100 setpoints worden geprogrammeerd. Bij overschrijding zullen de setpoints niet worden opgeslagen. De volgorde van programmeren is hetzelfde als de volgorde van aflopen. Dit houdt in het eerste setpoint geprogrammeerd is het eerste setpoint wat wordt afgelopen.

Functie: Stel datum en tijd in

	Preamble	Functie	Lengte	Jaar	Maand	Uur	Min	Sec
Byte	0-1	2	3-4	5	6	7	8	9
Data	0xFFFFF	0x01	0x05	0-255	1-12	0-23	0-59	0-59

Figuur 4: Functie steldatum en tijd in

Deze functie dient voor het instellen van de tijd op de interface. Dit is de tijd die wordt gebruikt bij het genereren van datapunten en het opslaan naar het geheugen. Om het jaartal in te stellen moet men het jaartal minus 2000 invoeren.

Functie: Vraag data op

	Preamble	Functie	Lengte	Keuze		Beschrijving
Byte	0-1	2	3-4	5		
Data	0xFFFFF	0x02	0x01	0-3		
				0	Vraag Alarmen op	
				1	Vraag Status op	
				2	Vraag geheugen data op	
				3	Vraag programma rij	

Figuur 5: Functie vraag data op

Deze functie wordt gebruikt om data uit de interface op te vragen. Hierbij zijn er een aantal mogelijkheden die te vinden zijn in Figuur 5.

Functie: Start/Stop programma

	Preamble	Functie	Lengte	Keuze
Byte	0-1	2	3-4	5
Data	0xFFFFF	0x03	0x01	0-2
Keuze	Beschrijving			
0	Stop			
1	Start Programma			
2	Start Manueel			

Figuur 6: Functie start / stop programma

Deze functie dient voor het stoppen en starten van de besturing. Hierbij is er een mogelijkheid bij het starten om te kiezen voor manueel en programma.

Functie: poll interface

	Preamble	Functie	Lengte	sleutel
Byte	0-1	2	3-4	5-36

Data	0xFFFF	0x04	0x20	febb9dc570881d413ec088d48a478411
------	--------	------	------	----------------------------------

Figuur 7: functie poll interface

Deze functie dient voor het vinden van de module. Hierbij wordt het pakket naar alle gebruikers in het netwerk gestuurd. Vervolgens zal er worden gekeken wie hier op antwoord met het juiste antwoord. Er bevindt zich een sleutel in dit pakket zodat de kans kleiner wordt dat door een willekeurige aansturing de interface reageert.

Functie: Set manuele temperatuur

	Preamble	Functie	Lengte	Temperatuur	Stijging
Byte	0-1	2	3-4	5-6	7
Data	0xFFFF	0x05	0x03	-40 - 130 C°	0.1 - 2.0C°

Figuur 8: Functie set manuele temperatuur

Deze functie programmeert de manuele temperatuur in de interface. Deze zal bij een gestart manuele programma direct worden overgenomen en direct worden uitgevoerd.

Functie: Set alarm niveaus

	Preamble	Functie	Lengte	Temp 0 waarschuwing	Temp 1 waarschuwing	Temp 2 waarschuwing
Byte	0-1	2	3-4	5-6	7-8	9-10
Data	0xFFFF	0x05	0x14	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°
Temp 3 waarschuwing	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm	Alarm/Relais aan	
11-12	13-14	15-16	17-18	19-20	21	
-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	0-255	

Figuur 9: Functie set alarm niveaus

Met deze functie worden de alarm- en waarschuwingsniveaus ingesteld. De waarschuwingstemperatuur. Zal alleen een melding geven en de alarm temperatuur zal het bijbehorende relais schakelen. Met de laatste byte is het mogelijk om het relais wel of niet te gebruiken en de alarmen aan of uit te zetten. Hierbij is de eerste nibble de relais en de tweede nibble de alarmen.

Functie: Echo alarm

	Preamble	Functie	Lengte
Byte	0-1	2	3-4
Data	0xFFFF	0x04	0x00

Figuur 10: Functie echo alarm

Deze functie dient voor het accepteren van een alarm. Dit wordt gedaan om zeker te weten of het alarm aan is gekomen.

Functie: Clear Setpoints

	Preamble	Functie	Lengte
Byte	0-1	2	3-4
Data	0xFFFF	0x08	0x00

Deze functie verwijdert de klaargezette setpoints in de profiel modus

Functie: Reset Relais

	Preamble	Functie	Lengte	Relais nummer

Byte	0-1	2	3-4	1
Data	0xFFFF	0x09	0x01	1-4

Als een alarm zich heeft voor gedaan en het relais is geschakeld kan met deze functie het relais weer af worden geschakeld.

1.2 Functies naar de computersoftware

Deze functies worden vanuit de interface gestuurd en worden door verschillende manieren aangeroepen.

Functie: Stuur alarm

	Preamble	Functie	Lengte	Sensor nr	Niveau	Temperatuur
Byte	0-1	2	3-4	5	6	7 - 8
Data	0xF1F0	0xF0	0x04	0-4	0- 1	-40 - 130 C°

Figuur 11: Functie stuur alarm

Op het moment dat er een voor ingesteld alarm niveau wordt overschreden zal er een alarm of waarschuwing worden verstuurd. Het onderscheid wordt gemaakt door de niveau waarde waarbij 0 waarschuwing en 1 alarm is.

Functie: Stuur data punt

	Preamble	Functie	Lengte	Temp 0	Temp 1	Temp2
Byte	0-1	2	3-4	5 - 6	7 - 8	9 - 10
Data	0xF1F0	0xF1	0x13	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°
Temp 3	Temp kast	Jaar	Maand	Uur	Min	Sec
11 - 12	13 - 14	15	16	17	18	19
-40 - 130 C°	-40 - 130 C°	0-255	0-12	0-24	0-60	0-60
I/O						

Figuur 12: Functie stuur data punt

Deze functie zal om de seconde worden uitgevoerd mits er een verbinding is. Er zal een data punt worden gestuurd met als inhoud temperaturen, tijd, datum en input en output staten.

Functie: Stuur status

	Preamble	Functie	Lengte	Status
Byte	0-1	2	3-4	5
Data	0xF1F0	0xF2	0x01	0-2

Figuur 13: Functie stuur status

Deze functie geeft de status van de interface weer. Hierbij is te zien of de interface is gestart of gestopt en of hij manueel of automatisch is ingesteld. De functie wordt aangeroepen door de "vraag data op" functie vanuit de computersoftware naar de interface.

Functie: Stuur alarmen

	Preamble	Functie	Lengte	Temp 0 waarschuwing	Temp 1 waarschuwing	Temp 2 waarschuwing
Byte	0-1	2	3-4	5-6	7-8	9-10
Data	0xF1F0	0xF3	0x13	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°
Temp 3 waarschuwing	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm	Temp 0 alarm
11-12	13-14	15-16	17-18	19-20		
-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	-40 - 130 C°	

Figuur 14: Functie stuur alarmen

Deze functie laat de ingestelde alarm en waarschuwingstemperaturen zien. De functie wordt aangeroepen door de “vraag data op” functie vanuit de computersoftware naar de interface.

Functie: Poll echo

	Preamble	Functie	Lengte	Unique ID
Byte	0-1	2	3-4	5-7
Data	0xF1F0	0xF4	0x03	0xXXX

Figuur 15: Functie poll echo

Deze functie wordt opgeroepen door de poll functie. Deze functie zorgt ervoor dat de module makkelijk lokaliseerbaar is. De module zal na het ontvangen van een poll aanvraag dit pakket versturen. Hierin staat de chip zijn unieke ID-code. Deze wordt mee gestuurd om zo te zorgen dat modulen kunnen worden onderscheiden van elkaar.

Functie: Stuur geheugen block

	Preamble	Functie	Lengte	Block teller	Punt1	...	Punt60
Byte	0-1	2	3-4	5-6	7-19	...	715-727
Data	0xF1F0	0xF4	0x2D2	0x0-0xFFFF	12	...	12

Figuur 16: Functie stuur geheugenblock

Deze functie zal een block van 1min datapunten uit het geheugen sturen. De functie wordt aangeroepen door de “vraag data op” functie vanuit de computersoftware naar de interface. Als deze functie is aangeroepen zal de interface alle geheugen data sturen. Hierbij wordt de data opgesplitst in pakketten van 1 minuut.

Functie: Stuur programma rij

	Preamble	Functie	Lengte	Huidige tijd	Setpoint 0 (Huidige)	Setpoint 1	...
Byte	0-1	2	3-4	5	6-9	10-13	...
Data	0xF1F0	0xF4	I x 4 + 1	0-240 Min	Setpoint	Setpoint	...

Figuur 17: Functie stuur programma rij

Deze functie dient voor het sturen van de ingestelde setpoints die in de rij staan. De functie wordt aangeroepen door de "vraag data op" functie vanuit de computersoftware naar de interface. De grootte van deze functie is variabel. De berekening voor de lengte is $I \times 4 + 1$. Hierbij staat I voor het aantal setpoints wat die klaar staan in de rij. De lengte van het pakket zal bepaald worden door het aantal setpoints. De huidige tijd staat voor de tijd die waarop hij staat in het huidige programma. Het draaiende setpoint is het eerste in het pakket. Vervolgens zal als tweede pakket het tweede setpoint worden verstuurd enzovoorts. Het maximum setpoints is 100.

Duur	Temperatuur	Stijging
5	6 - 7	8
0-240 min	-40 - 130 C°	0.1 - 2.0C°

Bijlage 4: Testresultaten

FMEA

No.	Onderdeel	Sub onderdeel	Functie (functie)	mogelijke fault	mogelijke effecten van de fault	Ervaring	mogelijk herstellen van de fault	Waarde	Waarde	Detectie + R	Risicoanalyse
								N	R	P	H
	voeding										
	Omloop protec tie		Beschermt het systeem voor het verkeerd aanhalen van de voeding	Defect raken van de diode	schade bij het verkeerd aansluiten van de voeding of geen werking van het systeem	1	0	0	0	0	
	voedingsbaff er		Zorgt voor een geconditioneerd voeding signalen	De condensator kan zijn waardes verliesen	Het voedingspulsat is afwijken vertonen. Hierdoor zullen metingen minder accurate zijn en kan in het ergste gevallen de module uitschakelen.	4	0	0	0	0	Goede kwaliteit Condensatoren gebruiken in het productieproces
	zekering		Beschermt het systeem bij kortsluiting	De zekering kan defect raken bij een kortsluiting	De module zal geen voeding meer krijgen en hierdoor buitenwerking zijn	8	1	0	0	0	
	5V voeding		Zorgt voor de voeding van het processorbord en de loze modulen	Het schakel element van de geschakelaar voeding kan niet raken	De module zal geen voeding meer krijgen en hierdoor buitenwerking zijn	8	1	0	0	0	
	Power led		het aanzetten van de werking van de voeding van de gabinete	De condensator kan zijn waardes verliesen	De voeding spanning zal niet te wissel worden geconditioneerd.	6	1	0	0	0	
	Digitale inputs	Overspanningsprot ectie	Beschermt de ingang tegen het aansluiten van te hoge spanningen.	Gebruiker niet of te laat	Bij een overspanning zal de ingang defect raken.	6	0	0	0	0	
	Stroom bresensing		Zorgt voor de juiste stroom door de ingang	Wederstand gaat stuk	Overbelasting van de weerstand	1	2	12	0	0	
	States led		Werking van de ingang kan niet zijn	led is defect	Gebruiker ziet niet of het signaal in werking is	3	1	1	3	0	
	RS232										
	U8001	Convertiser IC	Het omzetten van signaal niveaus van de seriele transmissielijnen	Defect raken van de max202	geen communicatie van en naar de klimaatkast	8	1	0	0	0	
	Sensor output	Relais	De schakelen van door de gebruiker gestofneerde bron	het werkplaats van de contacten	Hetzelfd ingeschakeld blijft van het relais en hierdoor geen veiligheidseisen vervullen.	6	1	0	0	0	
		Relais Aansturing	het aanzetten van de relais	Moefet tract defect	Door de moefet tract defect kan de relais niet meer functioneren.	6	1	2	0	0	
		States led	De gebruikte staten weken of het relais is geschilderd of niet	led is defect	De gebruiker weet niet of het relais ingeschakeld	3	1	1	3	0	
	Sensor inlezing	Diode beveiliging	Overspanningsbeveiliging	het pot gaan van de dioden	het kapot maken van voeding of andere componenten	6	0	0	0	0	
	Hoofdplaat	LCD scherm	Zorgt voor de voeding van de processor	Relais defect	Het niet werken van de module.	6	1	12	0	0	
			het omzetten van een weerstandswaarde naar digitale signalen	Het glas breekt	gebruiker kan de meetwaarden niet lezen op het scherm	2	2	4	4	0	
		Ethernet	Zorgt voor de communicatie tussen computer en interface	De connecteur die play of andere elektronica kan defect raken	Er zal geen verbinding meer zijn en de klimaatkast kan niet worden aangesloten. Hierdoor is het niet mogelijk om de interface te gebruiken	8	1	0	0	0	
	Processor		zorgt voor alle digitale communicatie	functioneert niet meer naast bovenstaan	de interface zal niet meer werken met de processor	8	1	0	0	0	

FAT

Factory Acceptance Test			Revisie 1.0
FAT ID:			Datum: 13-01-2016

Naam Controleur:	Daniel Keekstra	Bedrijf Controleur:	Brunelco
------------------	-----------------	---------------------	----------

Product:	Klimaatkast besturing ten behoeve van Votsch VT4002		
Bedrijf:	Brunelco		
Proces:			
Plaats van afname:	Haaksbergen, Brunelco, Textielstraat 3A		
Highlevel-software Onderdelen	Computersoftware	Revisie	0.7
Low-level-Software Onderdelen	Software in de interface	Revisie	0.8
Hardware Onderdelen	Interface board in combinatie met STM32f7 - disco	Revisie	1.0

	FAT Checklist Functionaliteit	Ja/ Nee	Commentaar
Hardware			
FV0101	Is er mogelijkheid om 4 NTC-temperatuur sensoren aan te sluiten?	Ja	Een NTC van 2kohm
FV0102	Zijn er 4 monitorbare ingangen?	Ja	Te zien op de pc-software
FV0103	Zijn er 4 uitgangen beschikbaar?	Ja	Relais NO-NC
FV0201	Is een net adapter aansluitbaar?	Ja	Via een terminal block
FV0301	Is het mogelijk om op afstand met de interface te verbinden?	Ja	Via ethernet
FV0302	Is de interface grafisch benaderbaar binnen Brunelco	Ja	Via ethernet dat overal bereikbaar is via wifi of kabel
FV0401	Wordt de klimaatkast aangestuurd?	Ja	Als deze is aangesloten via RS232
Software			
FV0501	Is er een mogelijkheid om te kiezen tussen manueel en profiel?	Ja	Dit kan in de computersoftware
FV0502	Is het mogelijk om temperatuur profielen te configureren	Ja	Dit kan in de computersoftware
FV0503	Is het mogelijk om per sensor een limiet waarde in te stellen?	Ja	Dit wordt ingesteld via de computersoftware en naar de interface gestuurd
FV0504	Wordt er een alarm melding gegeven bij een overschrijving van een limiet waarde?	Ja	Deze zal worden weergegeven op de computer
FV0601	Blijft de log data opgeslagen op wanneer de voeding wegvalt?	Ja	Deze wordt na de SD kaart geschreven in een .csv bestand
FV0602	Is het mogelijk om de log data te exporteren	Ja	Via de computersoftware en bij uitval via de SD kaart in de interface.

FV0603	Wordt de volgende data gelogd? - Setpoint - Limietwaarden - Sensorstemperaturen - Klimaatkasttemperatuur - Status digitale-ingangen - Status digitale-uitgangen - Tijdstempel	Nee	De limietwaarden worden een keer in de header van de logfile geschreven en niet per punt.
FV0701	Is het mogelijk om de data van de sensoren zichtbaar te maken in een grafiek?	Ja	Dit gebeurt automatisch in de computersoftware
FV0702	Is er data van actuele status, grenswaarden en temperaturen zichtbaar?	Ja	Dit is zichtbaar in de computersoftware
FV0703	Is er een mogelijk om een keuze te maken in het zichtbaar maken van sensoren in de grafiek?	Ja	Deze zijn in het view menu aan en uit te zetten en van kleur en dikte te veranderen.

	FAT Checklist Technische aspecten	Ja/Nee	Commentaar
TV0101	Bevind de dc-voedingsspanning zich tussen 11 en 13 Volt DC?	Ja	
TV0102	Bevind zich er geen groter AC-component op de voeding dan 200mV?	Ja	
TV0103	Is de maximale opname van de interface niet meer dan 1,5A?	Ja	Deze ligt er ver onder met 0,5 ampère
TV0104	Bevind er een zekering in de voedingslijn?	Ja	In SMD-formaat van 2 Ampère
TV0201	Is er en communicatie tussen de interface en de klimaatkast volgens het RS232 protocol?	Ja	
TV0401	Werkt de interface bij een temperatuur van 0 graden Celsius?	x	Deze test is niet uitgevoerd omdat mocht het product stuk gaan tijdens de test er geen tweede is.
TV0402	Werkt de interface bij een temperatuur van 50 graden Celsius?	x	""
TV0403	Werkt de interface bij een luchtvochtigheid van 80%?	x	De benodigde test apparatuur is niet aanwezig hiervoor
TV0501	Werkt de interface nog naar opslag bij -20 graden Celsius?	x	Deze test is niet uitgevoerd omdat mocht het product stuk gaan tijdens de test er geen tweede is.
TV0502	Werkt de interface nog naar opslag bij 90 graden Celsius?		""
TV0503	Werkt de interface nog na een opslag in een luchtvochtigheid van 80%?		De benodigde test apparatuur is niet aanwezig hiervoor
TV0601	Voldoet de interface aan de IP20 specificaties?	x	Er Bevind zich geen behuizing om het product

TV0701	Is de data- verversingsnelheid 1Hz?	Ja	
TV0702	Is de grafische versnelling instelbaar van 1 tot 0.06 HZ?	Nee	Dit bleek niet nodig te zijn aangezien de waarden niet sneller als 1Hz ververst worden.
TV0703	Is de resolutie van de temperatuurmeting minimaal 0.1 graden Celsius?	Ja	
TV0704	Licht het meetbereik van de temperatuurmeting tussen - 40-130 graden Celsius?	Ja	
TV0705	Is de nauwkeurigheid over de volledige schaal gelijk aan of kleiner dan 2 graden Celsius?	X	De benodigde kalibratie apparatuur is niet aanwezig. Wel is gebleken dat de temperatuur in vergelijking met niet geïjkte meters redelijk overeenkomt.
TV0706	Is de aansluitmogelijkheid geschikt voor 10K NTC-temperatuursensoren?	Ja	Dit is mogelijk wel wordt geadviseerd om gebruik te maken van 2K NTC-weerstanden. Het is wel mogelijk om een 10K NTC te gebruiken alleen wordt de gehele range niet behaald.
TV0801	Geeft de digitale ingang een 0 aan bij een spanning van 0 tot 0.3V DC?	Ja	Dit is getest doormiddel van een regelbare voeding.
TV0802	Geeft de digitale ingang een 1 aan bij een waarde van 2.8 tot 30V DC?	Ja	Dit is getest doormiddel van een regelbare voeding. Bij 30 V worden de weerstanden redelijk warm maar vallen ze nog steeds binnen de norm.
TV0901	Kunnen de uitgangen een spanning 250V AC voeren ?	Ja	Dit is duidelijk te halen uit het ontwerp waarbij de relais de stroom bepalen.
TV0902	Kunnen de uitgangen een stroom 8A AC voeren ?	Ja	Dit is duidelijk te halen uit het ontwerp waarbij de relais de stroom bepalen.
TV0903	Kunnen de uitgangen een spanning 30V DC voeren ?	Ja	Dit is duidelijk te halen uit het ontwerp waarbij de relais de stroom bepalen.
TV0904	Kunnen de uitgangen een stroom 8A DC voeren ?	Ja	Dit is duidelijk te halen uit het ontwerp waarbij de relais de stroom bepalen.
TV1001	Voldoet de interface aan de CE eisen voor industrieel gebruik?	x	Deze test is niet uitgevoerd omdat mocht het product stuk gaan tijdens de test er geen tweede is.

Beschrijving van de testen:

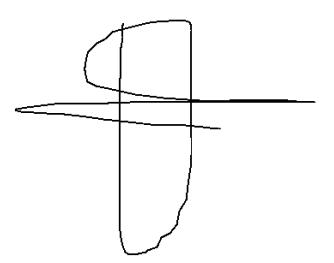
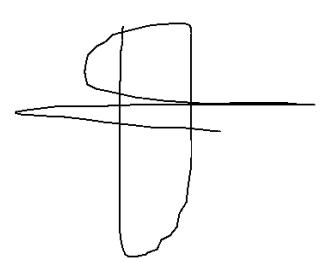
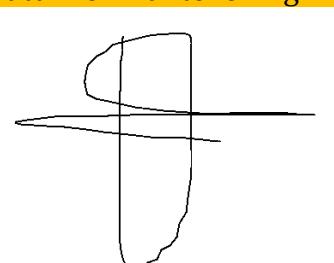
Werkingstest: In deze test is de module op zijn werking getest. Dit is gebeurt door een profiel te schrijven die de temperatuur laat variëren over een aantal uur.

Electrische testen: In deze test is de voeding gemeten voor de gesteld eisen.

Mogelijkheden test: Hierbij is gekeken of de gebruiker de mogelijkheden heeft die zijn gesteld en of deze werken.

Testen meetbereik: dit is gedaan doormiddel van een potmeter die de NTC simuleerde.

Testen Digitale ingangen: er is met een regelbare voeding na gegaan of de ingangen bij de juiste spanningen aan en uit gaan.

<p>De geteste criteria zijn behaald (Ja/Nee):</p> <p>Ja, met uitzondering van:</p> <ul style="list-style-type: none"> - De logpunten. Hierbij wordt niet per log punt de datum opgeslagen maar een keer globaal in de header. Dit zorgt voor een overzichtelijk bestand. De klant is hiermee akkoord. - De NTC weerstanden. Verandering van 10kohm naar 2kohm. Dit om met het converter ic toch de benodigde range te behalen. Dit is overlegd met de klant en is in orde. - De verversing snelheid is niet instelbaar omdat bleek dat dit geen meerwaarde aan het project gaf. Dit is overlegd met de klant en akkoord bevonden. 	<p>Datum en Hantekening:</p>  <p>18 - 01 - 16</p>
<p>Beschrijving van het behaalde resultaten:</p> <p>De resultaten waren na overleg met de klant in orde.</p>	<p>Datum en Hantekening:</p>  <p>18 - 01 - 16</p>
<p>Resterende opdrachten:</p> <p>Vocht test</p> <p>CE Test</p> <p>Temperatuur testen</p>	<p>Datum en Hantekening:</p>  <p>18 - 01 - 16</p>

Datum: 18-01-16

Naam controleur: Danel Keekstra

Bedrijf controleur: Brunelco

SAT

Site Acceptance Test			Revisie 1.0
ID:01		Datum:	13-01-2016

Naam Controleur:	Niek Voogsgeerd	Bedrijf Controleur:	Brunelco
------------------	-----------------	---------------------	----------

Product:	Klimaatkast besturing		
Bedrijf:	Brunelco		
Proces:			
Plaats van afname:	Haaksbergen, Brunelco, Textielstraat 3A		
Highlevel-software Onderdelen	C# Computersoftware	Revisie	0.8
Low-level-Software Onderdelen	C software in de interface	Revisie	0.9
Hardware Onderdelen	Interface board + stm32 - Disco	Revisie	1.0

	FAT Checklist Functionaliteit	Ja/ Nee	Commentaar
Hardware			
FV0101	Is er mogelijkheid om 4 NTC temperatuur sensoren aan te sluiten?	Ja	
FV0102	Zijn er 4 monitorbare ingangen?	Ja	
FV0103	Zijn er 4 uitgangen beschikbaar?	Ja	
FV0201	Is een net adapter aansluitbaar?	Ja	
FV0301	Is het mogelijk om op afstand met de interface te verbinden ?	Ja	
FV0302	Is de interface grafisch benaderbaar binnen Brunelco	Ja	
FV0401	Wordt de klimaatkast aangestuurd?	Ja	
Software			
FV0501	Is er een mogelijkheid om te kiezen tussen manueel en profiel?	Ja	
FV0502	Is het mogelijk om temperatuur profielen te configureren	Ja	
FV0503	Is het mogelijk om per sensor een limiet waarde in te stellen?	Ja	
FV0504	Wordt er een alarm melding gegeven bij een overschrijving van een limiet waarde?	Ja	
FV0601	Blijft de log data opgeslagen op wanneer de voeding wegvalt ?	Ja	
FV0602	Is het mogelijk om de log data te exporteren	Ja	

FV0603	Wordt de volgende data gelogd? - Setpoint - Limietwaarden - Sensors temperaturen - Klimaatkasttemperatuur - Status digitale-ingangen - Status digitale-uitgangen - Tijdstempel	Ja	
FV0701	Is het mogelijk om de data van de sensoren zichtbaar te maken in een grafiek?	Ja	
FV0702	Is er data van actuele status, grenswaarden en temperaturen zichtbaar?	Ja	
FV0703	Is er een mogelijk om een keuze te maken in het zichtbaar maken van sensoren in de grafiek ?	Ja	

De geteste criteria zijn behaald (Ja/Nee): Ja	Datum en Hantekening:
Resterende opdrachten: Bugs oplossen	Datum en Hantekening:

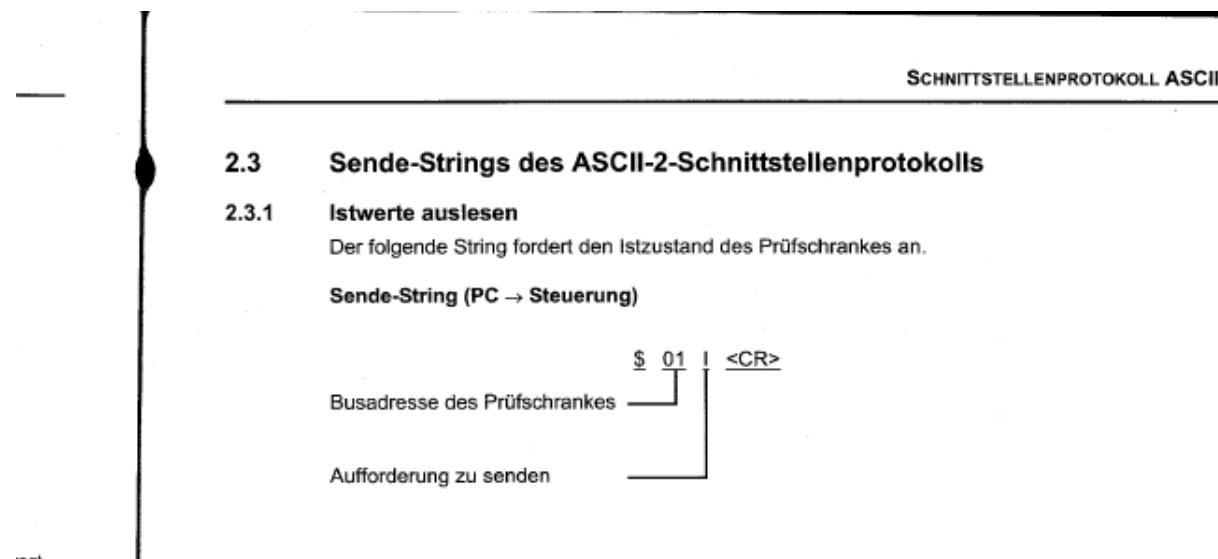
Datum: 19-01-16

Naam controleur: Niek Voogsgeerd

Bedrijf controleur: Brunelco

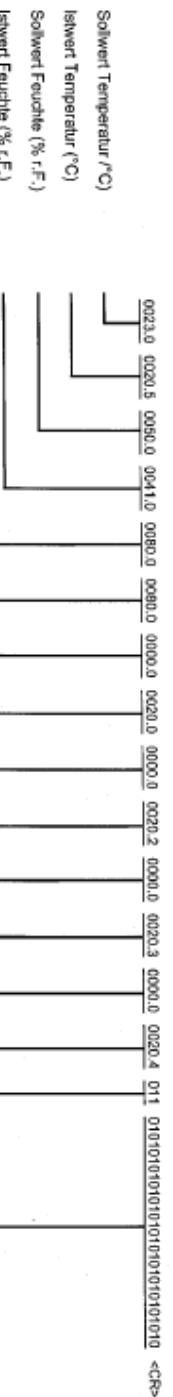
Bijlage 5: Datablad klimaatkast

In deze bijlage bevindt zich het protocol om de klimaat kast aan te sturen. Deze is in het duits.



2.3.2 Antwort-String (Steuerung → PC)

Der folgende String enthält Informationen über Ist- und Sollwert von Temperatur und Feuchte²⁾ sowie weitere Einstellwerte.
Die Werte beziehen sich auf den momentanen Zeitpunkt.



²⁾) Option
³⁾) nur Klimapuffer
nur Klimapufferstranke

Sollwert Temperatur (°C)	
Istwert Temperatur (°C)	
Sollwert Feuchte (% r.F.)	
Istwert Feuchte (% r.F.)	
Stellwert Ventilatordrehzahl (%)	
Stellwert Ventilatordrehzahl (%)	
unbenutzt	
Istwert Pt100-1 (°C, Analog-IO-Karte) ¹⁾	
unbenutzt	
Istwert Pt100-2 (°C, Analog-IO-Karte) ¹⁾	
unbenutzt	
Istwert Pt100-3 (°C, Analog-IO-Karte) ¹⁾	
unbenutzt	
Istwert Pt100-4 (°C, Analog-IO-Karte) ¹⁾	
unbenutzer digitaler Ausgang 0	
digitaler Ausgang Start	
digitaler Ausgang Feuchte	
weitere Ausgängige der Steuerung	

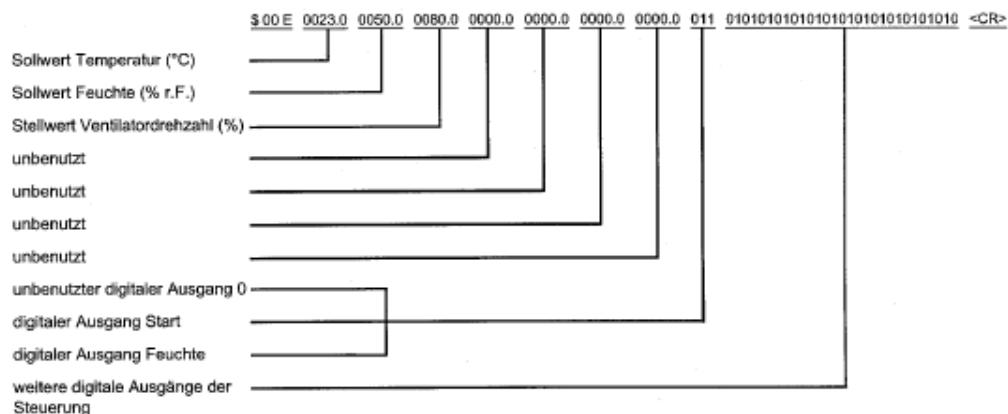


HINWEIS

Es werden immer 31 Digitalkanäle übertragen (mit Start und Feuchte, ohne digitalen Ausgang 0). Falls die Steuerung weniger Ausgänge besitzt, werden die nicht vorhandenen Kanäle mit 0 dargestellt.
Die Analogswerte sind jeweils durch ein Leerzeichen getrennt.

2.3.3 Sollwerte für Temperatur und Feuchte²⁾ einstellen

Der folgende String stellt die Sollwerte für Temperatur und Feuchte²⁾ auf 23 °C und 50% relative Feuchte ein und startet die Regelung.



2.3.4 Änderungsgeschwindigkeiten vorgeben¹⁾

Mit folgendem String ist es möglich, Gradienten für die Änderungsgeschwindigkeit der Sollwerte vorzugeben. Für Temperatur und relative Feuchte werden je zwei Gradienten für Heizen/Kühlen bzw. Be- und Entfeuchten festgelegt.

Der String lautet:

\$xxU aaaa.a bbbb.b cccc.c dddd.d <CR>

aaaaa.a Gradient Heizen

bbbb.b Gradient Kühlen (positives Vorzeichen!)

cccc.c Gradient Befeuchten

dddd.d Gradient Entfeuchten (positives Vorzeichen!)

xx Adresse des Prüfschrankes (1 bis 32)

SCHNITTSTELLENPROTOKOLL ASCII-2

2.3.5 Prüfprogramme starten und stoppen

Der folgende String startet ein Prüfprogramm.

Sende String (PC → Steuerung)

\$xxPyyy<CR>

xx Busadresse des Prüfschrances (1 bis 32)

yyy Nummer des Prüfprogramms (1 bis 120)

Antwort String (Steuerung → PC)

ø <CR> wenn das Programm gestartet werden konnte

Ein laufendes Programm kann durch den folgenden String gestoppt werden:

Sende String (PC → Steuerung)

\$xxP0000<CR>

xx Busadresse des Prüfschrances (1 bis 32)

yyy Nummer des Prüfprogramms (1 bis 120)

12 – 14

- 1) Option
- 2) nur Klimaprüfschränke
- 3) nur Superuser

eMincontrol
Schnitt.fm 10.2003 D

eMincontrol
Schnitt.fm 10.2003 D

SCHNITTSTELLENPROTOKOLL ASCII

2.3.6 Fehlermeldungen auslesen

Der folgende String gibt ein Bitmuster aller Fehlermeldungen aus.

Sende String (PC → Steuerung)

\$xxF<CR>

xx Busadresse des Prüfschrances (1 bis 32)

Antwort String (Steuerung → PC)

<Fehlernummer><Leerzeichen><Fehlertext><CR>

z.B.: 16 Netziederkehr<CR>

oder:

Ø <CR> wenn kein Fehler ansteht

HINWEIS

Es wird immer nur der erste Fehler angezeigt.

2.3.7 Fehlermeldungen quittieren

Der folgende String quittiert alle Fehler:

Sende String (PC → Steuerung)

\$xxQ<CR>

xx Busadresse des Prüfschrances (1 bis 32)

Antwort String (Steuerung → PC)

xx<CR>

xx Anzahl der noch anstehender Fehler

Bijlage 6: Logboek

01-09-2015

Vandaag begonnen met het doorlezen van het bhv plan hierna begonnen met oriënteren.

02-09-2015

Vandaag ben ik bezig geweest met het maken van planning en een deel van het project plan.

03-09-2015

Vandaag bezig geweest met het project plan en de eerste test gedaan met de klimaat kast en het aanstuur protocol

04-09-2015

Vandaag het test programma uitgebreid om maandag verder te testen.

07-09-2015

Vandaag verder met het project plan en het test programma iets aangepast.

08-09-2015

Het test programma getest en verder gewerkt aan het project plan.

09-09-2015

Verder met het projectplan en vooronderzoek gedaan naar ARM processoren.

10-09-2015

Het project plan voor de review afmaken

11-09-2015

Vandaag wordt het projectplan gereviewd en besproken. Ik ga vandaag ook verder met het opstellen van de kosten raming.

14-09-2015

Vandaag de laatste wijzigingen in het projectplan gedaan hierna begonnen met de functionele vereisten .

15-09-2015

Inzending voortgangsverslag en verder gaan met functionele vereisten en beginnen met technische vereisten.

16-09-2015

Afmaken van functionele en technische vereisten en laten reviewen. Beginnen met het testplan.

17-09-2015

Vandaag de functionele en technische vereisten verbeterd en besproken. En verder gegaan met het fat test plan.

18-09-2015

Verder met het fat test plan

21-09-2015

Beginnen met het onderzoeken van concepten

22-09-2015

Verder met het concept onderzoek oa blockdiagrammen gemaakt.

23-09-2015

Verder met het concept onderzoek. Communicatie mogelijkheden bekeken en het functionele blockdiagram afgemaakt.

24-09-2015

Verder met het concept onderzoek. Bezig met het onderzoeken van processors

25-09-2015

Verder met het maken van blockschema's en het concept onderzoek

28-09-2015

Verder gegaan met het concept onderzoek. Blockschema gemaakt voor de gehele interface.

29-09-2015

Later vanwege doktersbezoek daarna verder met het concept verslag.

30-09-2015

Onderzoek gedaan naar STM32 processoren en verwerkt in het concept verslag.

01-10-2015

Stage bezoek gehad van dhr. Stokkink beoordeling gekregen en verder met het concept verslag. Led test gedaan op een STM32 bordje.

02-10-2015

Bezig met het onderzoeken van een STM32 bord met usb.

05-10-2015

Alle diagrammen en flowcharts afgemaakt van het functioneel ontwerp

06-10-2015

Beschrijvingen bij de diagrammen gemaakt en een afspraak ingeplant om het functioneel ontwerp te bespreken met de opdrachtgever.

07-10-2015

Besproken met mijn begeleider of het functionele verslag de juiste inhoud had. En verder gegaan met het functionele verslag.

08-10-2015

Beschrijving van functionele blokken in het functionele ontwerp en begonnen met het testen van ethernet oplossingen voor de component keuze.

09-10-2015

Afronden van het functionele ontwerp en verder met het ethernet onderzoek.

12-10-2015

Vandaag testen gedaan met het stm32f7 bordje waarna deze stuk vervolgens met het test plan en de component keuze verder gegaan.

13-10-2015

Vandaag de protocol beschrijving van de communicatie tussen de interface en de computersoftware gemaakt.

14-10-2015

Vandaag een nieuw demo bordje binnen gekregen verder gaan met de TCP/IP stacks demo.

15-10-2015

Vandaag de key-component keuzes afgemaakt.

16-10-2015

Vandaag de eerste testen gedaan met een grafisch display. Hierbij een on screen debugger ontworpen.

19-10-2015

De debugger afgemaakt zodat het makkelijk is om te debuggen tijdens het project.

20-10-2015

Begonnen met het ontdekken van Altium hardware ontwerper.

21-10-2015

Begonnen met het opzetten van het print ontwerp.

22-10-2015

Vandaag verder gegaan met het schema al een aantal schema's afgemaakt.

23-10-2015

Schema's aangepast naar korte review door mijn stagebegeleider. Verder extra functionaliteit aangebracht aan de debug software.

26-10-2015

Vandaag verder gegaan met de IP/TCP Stack om de communicatie te voorzien.

27-10-2015

Vandaag het project geïntegreerd met de librarys en verder gegaan met de TCP stack

28-10-2015

Vandaag is het gelukt om met ethernet een dhcp te doen om een IP op te vragen en vervolgens pakketten te ontvangen en vervolgens te antwoorden. Ook heb ik het ST Discovery bordje toegevoegd aan het schema.

29-10-2015

Vandaag bezig geweest met de software. Ik heb de code opgeruimd en verduidelijkt.

30-10-2015

Vandaag begonnen met het bestukken van de pcb en de regels omtrent het bestukken (afstand tussen geleiders enz.)

02-11-2015

Vandaag verder gegaan met het bestukken van de modulen in de pcb.

03-11-2015

Afmaken van de module bestukking en vervolgens verder met het software design. Hierbij een TCP verbinding open gehouden zodat deze continu kan worden gebruikt zonder eerst een nieuwe verbinding te maken.

04-11-2015

Review gedaan van de Schema's vervolgens begonnen met aanpassen

05-11-2015

Schema aanpassingen afgemaakt en opnieuw begonnen met het pcb design.

06-11-2015

Pcb design bijna afgemaakt en daarbij een library gemaakt met een nieuwe connector.

09-11-2015

Test doen met TCP en een timer voor het versturen van data.

10-11-2015

Het nalopen van de PCB

11-11-2015

Het template verslag aan het maken. Beginnen met een opzet voor de computer software.

12-11-2015

Verder met het maken van het template verslag. Vervolgens verder gegaan met de computer user interface

13-11-2015

Het maken van een aanpasbare grafiek voor het instellen van programma's

16-11-2015

Het afmaken van het template verslag voor een review van Niek. Vervolgens verder met de user interface.

17-11-2015

Vandaag begonnen met het bestukken van de pcb.

18-11-2015

De pcb bestukt en de converter getest na dat bleek dat de converter een verkeerde spanning had is er een nieuwe besteld.

19-11-2015

Converter omgewisseld op de pcb en deze getest.

20-11-2015

Software geschreven voor de module aansturing van de I/O's

23-11-2015

Volledige test gedaan met de i/o's en software. Begonnen met het UART-protocol.

24-11-2015

Het UART-protocol afgemaakt en begonnen aan het inlezen van weerstanden met de max6691.

25-11-2015

Bezig met het nameten van de inlezen en de juiste sensor bepalen. Kleine aanpassingen doen in de pc-software.

26-11-2015

Het verder vormen van de pc-software.

27-11-2015

De grafische laag afgemaakt van de computersoftware.

30-11-2015

Ziek

01-12-2015

Ziek

02-12-2015

Ziek

03-12-2015

Vandaag begonnen met het maken van de timer interupts om er voor te zorgen dat er zo precies mogelijk wordt ingelezen.

04-12-2015

De timer weten te laten lopen met de juiste instellingen. Interrupts willen nog niet worden geactiveerd.

07-12-2015

Verder gegaan met het concept verslag. vervolgens de timer samen met Tim en Tom aan de praat gekregen naar heel wat debuggen.

08-12-2015

Verder gegaan met het concept verslag. en daarna de berekeningen voor rth in geprogrammeerd.

09-12-2015

Vandaag bezig geweest met de omzetting van Rth naar een temperatuur. Hierna naar school geweest voor een gesprek met mijn mentor.

10-12-2015

Vandaag aan de software gewerkt en een deel van het verslag afgemaakt.

11-12-2015

Een grafisch tabblad gemaakt voor de hardware interface.

14-12-2015

Vandaag het hardware hoofdstuk van het verslag geschreven.

15-12-2015

Verder aan het verslag gewerkt.

16-12-2015

Samen met mijn stagebegeleider het verslag doorgenomen en wijzigingen toegepast.

17-12-2015

De software aangepast voor de ethernet communicatie.

18-12-15

Het maken van grafische buttons en opschonen van code

Vakantie: twee weken thuis gewerkt aan de software hierbij een aansturing voor het display gecreerd en ethernet communicatie verbeterd en herschreven.

04-01-2016

Verder gegaan met de aansturing en koppeling van de klimaat besturing

05-01-2016

Eerste testen gedaan met de klimaat kast.

06-01-2016

Aansturing verdere getest en aangepast hierna een bug proberen te verwijderen in het touchscreen

07-01-2016

ethernet communicatie uitgebreid met een functie

08-01-2016

touch screen bug opgelost fout in STM library

11-01-2016

Software testen gedaan en testen voorbereiden

12-01-2016

Duur test gedaan met software en werken aan het verslag.

13-01-2016

applicatie software aangepast op taal en interface

14-01-2016

average filter toegevoegd voor extra stabiliteit.

15-01-2016

voorlichting voor een vervolgstudie en verder aan het verslag werken.

18-01-2016

Laatste testen gedaan om verder te gaan met het verslag

19-01-2016

SAT gedaan samen met de stage begeleider en verwerkt in het verslag. en verder gegaan met het verslag.

20-01-2016

verder gegaan met het verslag en een paar bugs opgelost.

21-01-2016

22-01-2016

Bijlage 7: Interface Software

In deze bijlage zijn twee type bestanden te vinden.

- Source file '.c'
- Header File '.h'

De interface is geheel in GNU C geschreven. Hierbij zijn alleen de eigen geschreven bestanden toegevoegd de software werkt samen met de Hal bibliotheken van STM die te verkrijgen zijn op www.st.com. Hierbij is de STM32CubeF7 Firmware Package V1.2.0(21-September-2015) gebruikt.

```
/************************************************************************\ 
* File:          main.c
* Project:      Climate Control
* Unit:         Interface v1
* Goal:         Main file of the project
*
* Platform:     STM32F746
* Created:      01-12-2015
* Author:        DanjelKeekstra
* Company:      Brunelco
\************************************************************************\ 
\***** Includes *****/
\***** Functions Declarations *****/
staticvoidMPU_Config(void);
staticvoidCPU_CACHE_Enable(void);
staticvoidSystemClock_Config(void);
staticvoidGUIThread(voidconst * argument);
staticvoidTimerCallback(voidconst *n);
staticvoidInitDateTime(void);
voidStartDefaultTask(voidconst * argument);
voidTouchThread(voidconst * argument);
voidInit_HAL(void);
voidGUI_init(void);

\***** Globals *****/
osThreadId defaultTaskHandle;
osTimerId lcd_timer;
osTimerId refresh_timer;
uint8_t ucHeap[ configTOTAL_HEAP_SIZE ] __attribute__ ((section(".RamData2")));
// time
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
\***** Main *****/
intmain(void)
{
    // initialize Hardware
    Init_HAL();
    k_BspInit();
    GUI_init();
    CC_Init();

    // Create default tasks
    osThreadDef(defaultTask, StartDefaultTask, osPriorityNormal, 0, 1024);
    defaultTaskHandle = osThreadCreate(osThread(defaultTask), NULL);
    // Create GUI tasks
    osThreadDef(GUI_Thread, GUIThread, osPriorityNormal, 0, 2048);
    osThreadCreate (osThread(GUI_Thread), NULL);

    // Start Kernal
    osKernelStart ();

    /* We should never get here as control is now taken by the scheduler */
    while(1);
}
\***** Default thread *****/
\***** StartDefaultTask *****/
voidStartDefaultTask(voidconst * argument)
{
    unsignedchar start = 1;
```

```

unsignedchar oldtime = 0;
SD_Init();
CC_WIN_Start();
Ethernet_Init(31415,5);

while(1)
{
    if(oldtime != time.Seconds)
    {
        oldtime = time.Seconds;
        CC_DebugLed_On(DEB1);
        switch(CC_State)
        {
            case CC_STOP:
                if (start == 0)
                {
                    CC_DebugLed_Off(DEB3);
                    CC_DebugLed_Off(DEB4);
                    SD_Deinit();
                    start = 1;
                }
                CC_Stop();
                break;
            case CC_MANUAL:
                CC_Run_Manual(start);
                start = 0;
                break;
            case CC_PROGRAM:
                CC_Run_Program(start);
                start = 0;
                break;
        }
        CC_DebugLed_Off(DEB1);
    }
    osDelay(10);
    k_GetTime(&time);
    k_GetDate(&date);
}

voidInit_HAL(void)
{
    /* Configure the MPU attributes as Write Through */
    MPU_Config();

    /* Enable the CPU Cache */
    CPU_CACHE_Enable();

    HAL_Init();

    /* Configure the system clock @ 200 Mhz */
    SystemClock_Config();
}
voidGUI_init(void)
{
    /* Initialize GUI*/
    GUI_Init();
    WM_MULTIBUF_Enable(1);
    GUI_SetLayerVisEx (1, 0);
    GUI_SelectLayer(0);
    GUI_SetBkColor(GUI_BLACK);
    GUI_Clear();
}
staticvoidInitDateTime(void)
{
    RTC_DateTypeDef date;
    date.Year = 15;
    date.Month = 10;
    date.Date = 20;
    date.WeekDay = RTC_WEEKDAY_MONDAY;
    k_SetDate(&date);
    RTC_TimeTypeDef time;
    time.Hours = 17;
    time.Minutes = 15;
    time.Seconds = 0;
    time.TimeFormat = RTC_HOURFORMAT_24;
    k_SetTime(&time);
}
/* Hal Functions
 */
/** 
 * @brief GUI task to update the screen
 * @param argument: pointer that is passed to the thread function as start argument.
 * @retval None
 */

```

```

/*
static void GUI_Thread(void const * argument)
{
/* Gui background Task */
while(1)
{
    CC_WIN_UIO(CC_ReadInputs()); // check if inputs changed en set buttons
    UpdateProcess();
    GUI_Exec(); /* Do the background work ... Update windows etc. */
    osDelay(30); /* Nothing left to do for the moment ... Idle processing */
}
/**/
/* @brief System Clock Configuration
*   The system Clock is configured as follow :
*       System Clock source      = PLL (HSE)
*       SYSCLK(Hz)              = 200000000
*       HCLK(Hz)                = 200000000
*       AHB Prescaler          = 1
*       APB1 Prescaler          = 4
*       APB2 Prescaler          = 2
*       HSE Frequency(Hz)       = 25000000
*       PLL_M                   = 25
*       PLL_N                   = 400
*       PLL_P                   = 2
*       PLLSAI_N                = 384
*       PLLSAI_P                = 8
*       VDD(V)                  = 3.3
*       Main regulator output voltage = Scale1 mode
*       Flash Latency(WS)       = 6
* @param None
* @retval None
*/
void SystemClock_Config(void)
{
RCC_ClkInitTypeDef RCC_ClkInitStruct;
RCC_OscInitTypeDef RCC_OscInitStruct;
HAL_StatusTypeDef ret = HAL_OK;

/* Enable HSE Oscillator and activate PLL with HSE as source */
RCC_OscInitStruct.OscillatorType = RCC_OSCILLATORTYPE_HSE;
RCC_OscInitStruct.HSEState = RCC_HSE_ON;
RCC_OscInitStruct.PLL.PLLState = RCC_PLL_ON;
RCC_OscInitStruct.PLL.PLLSource = RCC_PLLSOURCE_HSE;
RCC_OscInitStruct.PLL.PLLM = 25;
RCC_OscInitStruct.PLL.PLLN = 400; //400
RCC_OscInitStruct.PLL.PLLP = RCC_PLLP_DIV2;
RCC_OscInitStruct.PLL.PLLQ = 9;
HAL_RCC_OscConfig(&RCC_OscInitStruct);

ret = HAL_PWREx_EnableOverDrive();

if(ret != HAL_OK)
{
while(1) { ; }
}

/* Select PLL as system clock source and configure the HCLK, PCLK1 and PCLK2
   clocks dividers */
RCC_ClkInitStruct.ClockType = (RCC_CLOCKTYPE_SYSCLK | RCC_CLOCKTYPE_HCLK | RCC_CLOCKTYPE_PCLK1 |
RCC_CLOCKTYPE_PCLK2);
RCC_ClkInitStruct.SYSCLKSource = RCC_SYSCLKSOURCE_PLLCLK;
RCC_ClkInitStruct.AHBCLKDivider = RCC_SYSCLK_DIV1;
RCC_ClkInitStruct.APB1CLKDivider = RCC_HCLK_DIV4;
RCC_ClkInitStruct.APB2CLKDivider = RCC_HCLK_DIV2;
HAL_RCC_ClockConfig(&RCC_ClkInitStruct, FLASH_LATENCY_6);
}
/**/
/* @brief This function provides accurate delay (in milliseconds) based
   on SysTick counter flag.
* @note This function is declared as __weak to be overwritten in case of other
   implementations in user file.
* @param Delay: specifies the delay duration in milliseconds.
* @retval None
*/
void HAL_Delay(__IO uint32_t Delay)
{
while(Delay)
{
if (SysTick->CTRL & SysTick_CTRL_COUNTFLAG_Msk)
{
    Delay--;
}
}
}
/**/
/* @brief Configure the MPU attributes as Write Through for SRAM1/2.
* @note The Base Address is 0x20010000 since this memory interface is the AXI.

```

```

/*
 *      The Region Size is 256KB, it is related to SRAM1 and SRAM2 memory size.
 * @param None
 * @retval None
 */
static void MPU_Config(void)
{
    MPU_Region_InitTypeDef MPU_InitStruct;

    /* Disable the MPU */
    HAL_MPU_Disable();

    /* Configure the MPU attributes as WT for SRAM */
    MPU_InitStruct.Enable = MPU_REGION_ENABLE;
    MPU_InitStruct.BaseAddress = 0x20010000;
    MPU_InitStruct.Size = MPU_REGION_SIZE_256KB;
    MPU_InitStruct.AccessPermission = MPU_REGION_FULL_ACCESS;
    MPU_InitStruct.IsBufferable = MPU_ACCESS_NOT_BUFFERABLE;
    MPU_InitStruct.IsCacheable = MPU_ACCESS_CACHEABLE;
    MPU_InitStruct.IsShareable = MPU_ACCESS_NOT_SHAREABLE;
    MPU_InitStruct.Number = MPU_REGION_NUMBER0;
    MPU_InitStruct.TypeExtField = MPU_TEX_LEVEL0;
    MPU_InitStruct.SubRegionDisable = 0x00;
    MPU_InitStruct.DisableExec = MPU_INSTRUCTION_ACCESS_ENABLE;

    HAL_MPU_ConfigRegion(&MPU_InitStruct);

    /* Enable the MPU */
    HAL_MPU_Enable(MPU_PRIVILEGED_DEFAULT);
}
/***
 * @brief CPU L1-Cache enable.
 * @param None
 * @retval None
 */
static void CPU_CACHE_Enable(void)
{
    /* Enable branch prediction */
    SCB->CCR |= (1 <<18);
    __DSB();

    /* Enable I-Cache */
    SCB_EnableICache();

    /* Enable D-Cache */
    SCB_EnableDCache();
}

void vApplicationStackOverflowHook( TaskHandle_t xTask, signedchar *pcTaskName )
{
    __asm( "BKPT 0;" );
}

void vApplicationMallocFailedHook( void )
{
    __asm( "BKPT 0;" );
}

#ifndef USE_FULL_ASSERT
/***
 * @brief assert_failed
 * Reports the name of the source file and the source line number
 * where the assert_param error has occurred.
 * @param File: pointer to the source file name
 * @param Line: assert_param error line source number
 * @retval None
 */
void assert_failed(uint8_t* file, uint32_t line)
{
    /* User can add his own implementation to report the file name and line
     * number, ex: printf("Wrong parameters value: file %s on line %d\r\n",
     * file, line) */

    /* Infinite loop */
    while (1)
        {};
}
#endif
/*********************************************

```

```
*****\n* File: main.h\n* Project: Climate Control\n* Unit: Interface v1\n* Goal: Main header file of the project\n*\n* Platform: STM32F746\n* Created: 01-12-2015\n* Auhtor: Danjel Keekstra\n* Company: Brunelco\n*****\n#ifndef __MAIN_H\n#define __MAIN_H\n\n#ifdef __cplusplus\nextern"C" {\n#endif\n\n/* Includes -----*/\n#include"stm32f7xx_hal.h"\n#include"stm32f7xx_it.h"\n/* EVAL includes components */\n#include"stm32746g_discovery.h"\n#include"stm32746g_discovery_sdram.h"\n#include"stm32746g_discovery_ts.h"\n#include"stm32746g_discovery_sd.h"\n#include"stm32746g_discovery_qspi.h"\n#include"stm32746g_discovery_eeprom.h"\n/* GUI includes components */\n#include"gui.h"\n#include"dialog.h"\n#include"STM32746G_Discovery_STemWin_Addons.h"\n/* Kernel includes components */\n#include"k_module.h"\n#include"k_storage.h"\n#include"k_rtc.h"\n#include"k_bsp.h"\n/* utilities and standard includes components */\n#include"cpu_utils.h"\n#include<stddef.h>\n#include<stdlib.h>\n#include<string.h>\n#include<math.h>\n\n// eigen includes\n#include"ClimateModule.h"\n#include"Protocol.h"\n#include"DIALOG.h"\n#include"DebugWindow.h"\n#include"Ethernet.h"\n#include"ClimateModule_WIN.h"\n\n#endif __cplusplus\n}\n#endif/* __MAIN_H */\n*****\n* File: ClimateModule.c\n* Project: Climate Chamber Control\n* Unit: Main Interface\n* Goal: To control the addon board\n* Platform: STM32F7-Disco\n* Created: 20-11-2015\n* Auhtor: D. Keekstra\n* Company: Brunelco\n*****\n*****\n* Includes\n*****\n#include"ClimateModule.h"\n#include"stm32f7xx_hal.h"\n#include"DebugWindow.h"\n#include"stm32f7xx_hal.h"\n#include"stm32746g_discovery.h"\n#include"cmsis_os.h"\n#include"stm32f7xx_hal_gpio.h"\n#include"stm32f7xx_hal_tim.h"\n#include"math.h"\n#include"stm32f7xx_hal.h"\n#include<stdlib.h>\n#include<stdio.h>\n#include"Storage.h"\n//#include "errno.h"
```

```

// WARNING: HACK
int __errno=0;

/* Function Declarations
 */
voidCC_GPIO_Init(void);
voidCC_InitInput(GPIO_TypeDef *GPIOX, unsignedshort Pin);
voidCC_InitOutput(GPIO_TypeDef *GPIOX, unsignedshort Pin);
// Uart
voidCC_UART_Init(void);
voidCC_UART_THREAD(voidconst * argument);
// CU
voidMX_TIM4_Init(void);
// mem
voidCC_Flash_Init(void);

voidCC_FormString(char *str, short temp);

/* Globals
 */
osThreadId UARTPOLL;
unsignedchar PacketData[50];
// timer
volatileint TIM_CNT = 0;
volatilechar OW_Error = 0;
volatileunsignedshort TIM_Val[13][2]; // 13x hoogeneenlagewaabit 0 laag 1 hoogenhoogdecntbepaalt.
// uart
volatileunsignedshort UART_CNT = 0; // counter for uart stream
volatileunsignedchar UART_RXData[150]; // uart buffer
volatileunsignedchar UART_RXBuf[150]; // uart buffer
volatileunsignedchar UART_TXData[100]; // uart buffer
volatileunsignedchar UART_State = UART_DEINIT;
volatileunsignedchar UART_Send = 0;
volatileunsignedchar Uart_Inc[20];
__IO ITStatus UartReady = RESET;
__IO uint32_t UserButtonStatus = 0; /* set to 1 after User Button interrupt */
unsignedchar Run_Counter = 0;
unsignedchar Alarmbuf[4] = {0,0,0,0};
// sd
char FileNameBuf[50];
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
/* Basic Functions
 */
voidCC_Init(void)
{
    CC_GPIO_Init();
    CC_UART_Init();
    MX_TIM4_Init();
    CC_Update_Counter = 0;
}

voidCC_Stop(void)
{
    if(Run_Counter == 10) // eens per 10 secondeafvragen
    {
        CC_CAB_ReadRoomTemp(&CurrentTemp);
        Run_Counter = 0;
        CC_WIN_UTime();

    }
    CC_ReadOW(&CurrentTemp);
    CC_WIN_UTemp(&CurrentTemp);
    CC_WIN_UMode(CC_State, -50);
    Run_Counter++;
}
voidCC_Run_Manual(char Start)
{
    short settemp = 0;
    if(Start)
    {
        CC_DebugLed_On(DEB3);
        SD_CreateFile(&FileNameBuf, 0);
    }
    CC_ReadOW(&CurrentTemp);
    CC_CAB_CheckAlarm(&CurrentTemp);
    CC_WIN_UTemp(&CurrentTemp);
    SD_WriteTemps(&CurrentTemp, CC_ReadInputs(), FileNameBuf, CC_Manual.Temp);
    PC_SendTemps(&CurrentTemp, CC_ReadInputs());
    CC_WIN_UMode(CC_State, CC_Manual.Temp);

    if(Run_Counter == 10) // eens per 10 secondeafvragen
    {
        settemp = CC_CAB_ReadRoomTemp(&CurrentTemp);
        if(settemp != CC_Manual.Temp) CC_Manual.Send = 255;
    }
}

```

```

        Run_Counter = 0;
        CC_WIN_UTime();
    }
    if(Run_Counter == 4 && CC_Manual.Send)
    {
        CC_CAB_SetPoint(CC_Manual.Temp);
    }
    if(Run_Counter == 7 && CC_Manual.Send)
    {
        CC_Set_Steep(CC_Manual.Steep);
        CC_Manual.Send = 0;
    }
    Run_Counter++;
}
voidCC_Run_Program(char Start)
{
    short settemp = CC_Setpoints[CC_Update_Counter].Temp;
    if(Start)
    {
        CC_DebugLed_On(DEB4);
        SD_CreateFile(&FileNameBuf, 1);
    }

    CC_ReadOW(&CurrentTemp);
    CC_CAB_CheckAlarm(&CurrentTemp);
    CC_WIN_UTemp(&CurrentTemp);
    SD_WriteTemps(&CurrentTemp, CC_ReadInputs(), FileNameBuf, CC_Manual.Temp);
    PC_SendTemps(&CurrentTemp, CC_ReadInputs());
    CC_WIN_UMode(CC_State, CC_Setpoints[CC_Update_Counter].Temp);
    Run_Counter++;

    if(Run_Counter == 10) // eens per 10 seconde afvragen
    {
        settemp = CC_CAB_ReadRoomTemp(&CurrentTemp);
        if(settemp != CC_Setpoints[CC_Update_Counter].Temp) CC_Setpoints[CC_Update_Counter].Send =
255;
        Run_Counter = 0;
        CC_WIN_UTime();
    }
    if(Run_Counter == 4 && CC_Setpoints[CC_Update_Counter].Send)
    {
        CC_CAB_SetPoint(CC_Setpoints[CC_Update_Counter].Temp);
    }
    if(Run_Counter == 7 && CC_Setpoints[CC_Update_Counter].Send)
    {
        CC_Set_Steep(CC_Setpoints[CC_Update_Counter].Steep);
        CC_Setpoints[CC_Update_Counter].Send = 0;
    }

    // bepaal setpoint
    CC_Setpoints[CC_Update_Counter].Length--; // decrease the length
    if(CC_Setpoints[CC_Update_Counter].Length == 0)
    {
        CC_Update_Counter++; // if the setpoint is done
        if(CC_Update_Counter == CC_SetpointCount) // program done
        {

        }
    }
    if (CC_SetpointCount == 1)
    {

    }
}
voidCC_WriteOutput(unsignedchar ch, unsignedchar on);
//****************************************************************************
/* GPIO Functions
//****************************************************************************
voidCC_GPIO_Init(void)
{
    /* GPIO Ports Clock Enable */
    __GPIOG_CLK_ENABLE();
    __GPIOB_CLK_ENABLE();
    __GPIOA_CLK_ENABLE();
    __GPIOH_CLK_ENABLE();
    __GPIOI_CLK_ENABLE();

    //relays
    CC_InitOutput(RL1PO,RL1PI);
    CC_InitOutput(RL2PO,RL2PI);
    CC_InitOutput(RL3PO,RL3PI);
    CC_InitOutput(RL4PO,RL4PI);
    // Debug Leds
    CC_InitOutput(DEB1PO,DEB1PI);
    CC_InitOutput(DEB2PO,DEB2PI);
    CC_InitOutput(DEB3PO,DEB3PI);
    CC_InitOutput(DEB4PO,DEB4PI);
}

```

```

// Inputs
CC_InitInput(INP1PO, INP1PI);
CC_InitInput(INP2PO, INP2PI);
CC_InitInput(INP3PO, INP3PI);
CC_InitInput(INP4PO, INP4PI);
}
void CC_InitOutput(GPIO_TypeDef *GPIOx, unsignedshort Pin)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
}
void CC_InitInput(GPIO_TypeDef *GPIOx, unsignedshort Pin)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = Pin;
    GPIO_InitStruct.Mode = GPIO_MODE_INPUT;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    HAL_GPIO_Init(GPIOx, &GPIO_InitStruct);
}
void CC_Run_Test(void)
{
    PrintL((constchar*)"Running I/O Test");
    int delay = 500;
    // checking Relays
    CC_WIN_UIO(0b00000001);
    HAL_GPIO_WritePin(RL1PO, RL1PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(RL1PO, RL1PI, GPIO_PIN_RESET);
    CC_WIN_UIO(0b00000010);
    HAL_GPIO_WritePin(RL2PO, RL2PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(RL2PO, RL2PI, GPIO_PIN_RESET);
    CC_WIN_UIO(0b00000100);
    HAL_GPIO_WritePin(RL3PO, RL3PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(RL3PO, RL3PI, GPIO_PIN_RESET);
    CC_WIN_UIO(0b00001000);
    HAL_GPIO_WritePin(RL4PO, RL4PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(RL4PO, RL4PI, GPIO_PIN_RESET);
    CC_WIN_UIO(0b00000000);
    osDelay(delay);
    // checking Debug leds
    osDelay(delay);
    HAL_GPIO_WritePin(DEB1PO, DEB1PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(DEB1PO, DEB1PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(DEB2PO, DEB2PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(DEB2PO, DEB2PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(DEB3PO, DEB3PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(DEB3PO, DEB3PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(DEB4PO, DEB4PI, GPIO_PIN_SET);
    osDelay(delay);
    HAL_GPIO_WritePin(DEB4PO, DEB4PI, GPIO_PIN_RESET);
    osDelay(delay);

    // Checking input leds
    HAL_GPIO_WritePin(INP1PO, INP1PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(INP2PO, INP2PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(INP3PO, INP3PI, GPIO_PIN_RESET);
    HAL_GPIO_WritePin(INP4PO, INP4PI, GPIO_PIN_RESET);
    // set inputs as output low!
    CC_InitOutput(INP1PO, INP1PI);
    CC_InitOutput(INP2PO, INP2PI);
    CC_InitOutput(INP3PO, INP3PI);
    CC_InitOutput(INP4PO, INP4PI);
    CC_WIN_UIO(0b11110000);
    osDelay(delay);
    osDelay(delay);

    // Reset Inputs
    CC_InitInput(INP1PO, INP1PI);
    CC_InitInput(INP2PO, INP2PI);
    CC_InitInput(INP3PO, INP3PI);
    CC_InitInput(INP4PO, INP4PI);
    CC_WIN_UIO(0b00000000);
    PrintF("Test done");
}
unsignedchar CC_ReadInputs(void)
{
    char IO = 0;
    IO = IO + (0x01 & !HAL_GPIO_ReadPin(INP1PO, INP1PI) << 0);
}

```

```

        IO = IO + (0x02 & !HAL_GPIO_ReadPin(INP2PO, INP2PI) << 1);
        IO = IO + (0x04 & !HAL_GPIO_ReadPin(INP3PO, INP3PI) << 2);
        IO = IO + (0x08 & !HAL_GPIO_ReadPin(INP4PO, INP4PI) << 3);
        IO = IO + (0x10 &HAL_GPIO_ReadPin(RL1PO, RL1PI) << 4);
        IO = IO + (0x20 &HAL_GPIO_ReadPin(RL2PO, RL2PI) << 5);
        IO = IO + (0x40 &HAL_GPIO_ReadPin(RL3PO, RL3PI) << 6);
        IO = IO + (0x80 &HAL_GPIO_ReadPin(RL4PO, RL4PI) << 7);
        return IO;
    }

voidCC_WriteOutputs(unsignedchar Out)
{
    HAL_GPIO_WritePin(RL1PO, RL1PI, (Out >> 0)&1);
    HAL_GPIO_WritePin(RL2PO, RL2PI, (Out >> 1)&1);
    HAL_GPIO_WritePin(RL3PO, RL3PI, (Out >> 2)&1);
    HAL_GPIO_WritePin(RL4PO, RL4PI, (Out >> 3)&1);
}

voidCC_WriteOutput(unsignedchar ch, unsignedchar on)
{
    switch(ch)
    {
        case 0:
            HAL_GPIO_WritePin(RL1PO, RL1PI, on);
            break;
        case 1:
            HAL_GPIO_WritePin(RL2PO, RL2PI, on);
            break;
        case 2:
            HAL_GPIO_WritePin(RL3PO, RL3PI, on);
            break;
        case 3:
            HAL_GPIO_WritePin(RL4PO, RL4PI, on);
            break;
    }
}

/* **** Debug Leds **** */
voidCC_DebugLed_On(char LED)
{
    switch(LED)
    {
        case DEB1:
            HAL_GPIO_WritePin(DEB1PO, DEB1PI, GPIO_PIN_SET);
            break;
        case DEB2:
            HAL_GPIO_WritePin(DEB2PO, DEB2PI, GPIO_PIN_SET);
            break;
        case DEB3:
            HAL_GPIO_WritePin(DEB3PO, DEB3PI, GPIO_PIN_SET);
            break;
        case DEB4:
            HAL_GPIO_WritePin(DEB4PO, DEB4PI, GPIO_PIN_SET);
            break;
    }
}

voidCC_DebugLed_Off(char LED)
{
    switch(LED)
    {
        case DEB1:
            HAL_GPIO_WritePin(DEB1PO, DEB1PI, GPIO_PIN_RESET);
            break;
        case DEB2:
            HAL_GPIO_WritePin(DEB2PO, DEB2PI, GPIO_PIN_RESET);
            break;
        case DEB3:
            HAL_GPIO_WritePin(DEB3PO, DEB3PI, GPIO_PIN_RESET);
            break;
        case DEB4:
            HAL_GPIO_WritePin(DEB4PO, DEB4PI, GPIO_PIN_RESET);
            break;
    }
}

voidCC_DebugLed_Toggle(char LED)
{
    switch(LED)
    {
        case DEB1:
            HAL_GPIO_TogglePin(DEB1PO, DEB1PI);
            break;
        case DEB2:
            HAL_GPIO_TogglePin(DEB2PO, DEB2PI);
            break;
        case DEB3:
            HAL_GPIO_TogglePin(DEB3PO, DEB3PI);
            break;
        case DEB4:
            HAL_GPIO_TogglePin(DEB4PO, DEB4PI);
            break;
    }
}

```

```

        break;
    }

/***** MAX6691 Functions *****/
/* Function will return error state */
unsignedchar CC_ReadOW(Temp *temp)
{
    OW_Error = 0; // set error on zero
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = OWPI;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
    HAL_GPIO_Init(OWPO, &GPIO_InitStruct);

    // geef pulse omsequentsetest starten
    HAL_GPIO_WritePin(OWPO, OWPI, GPIO_PIN_RESET);
    osDelay(10);
    HAL_GPIO_WritePin(OWPO, OWPI, GPIO_PIN_SET);

    GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
    GPIO_InitStruct.Mode = GPIO_MODE_AF_PP;
    HAL_GPIO_Init(OWPO, &GPIO_InitStruct);
    // lees pulse lengten
    TIM_CNT = 0;
    int timeout = 150;
    while(TIM_CNT < 10 && timeout)
    {
        osDelay(1);
        timeout--;
    }
    if (timeout <= 0 || OW_Error || TIM_Val[1][1] != 1) return 0xFF; // error detected
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    //-----|-----|-----|-----|-----|-----|-----|-----|
    float TlTh[4];
    int Rth[4];
    float Tempf[4];
    // calculate Thigh / Tlow
    TlTh[0] = (TIM_Val[2][0] - TIM_Val[1][0]) / (float)(TIM_Val[3][0] - TIM_Val[2][0]);
    TlTh[1] = (TIM_Val[4][0] - TIM_Val[3][0]) / (float)(TIM_Val[5][0] - TIM_Val[4][0]);
    TlTh[2] = (TIM_Val[6][0] - TIM_Val[5][0]) / (float)(TIM_Val[7][0] - TIM_Val[6][0]);
    TlTh[3] = (TIM_Val[8][0] - TIM_Val[7][0]) / (float)(TIM_Val[9][0] - TIM_Val[8][0]);
    // calculate Rthermistor
    Rth[0] = (OW_Rext / (TlTh[0] + 0.0002f) - OW_Rext) - OW_Rser;
    Rth[1] = (OW_Rext / (TlTh[1] + 0.0002f) - OW_Rext) - OW_Rser;
    Rth[2] = (OW_Rext / (TlTh[2] + 0.0002f) - OW_Rext) - OW_Rser;
    Rth[3] = (OW_Rext / (TlTh[3] + 0.0002f) - OW_Rext) - OW_Rser;
    // calculate temp
    Tempf[0] = 1 / (log((double)Rth[0] / NTC_R25) / NTC_B25 + 1 / NTC_T25) - 273.15;
    Tempf[1] = 1 / (log((double)Rth[1] / NTC_R25) / NTC_B25 + 1 / NTC_T25) - 273.15;
    Tempf[2] = 1 / (log((double)Rth[2] / NTC_R25) / NTC_B25 + 1 / NTC_T25) - 273.15;
    Tempf[3] = 1 / (log((double)Rth[3] / NTC_R25) / NTC_B25 + 1 / NTC_T25) - 273.15;

    temp->TempSensor[0] = Tempf[0] * 10;
    temp->TempSensor[1] = Tempf[1] * 10;
    temp->TempSensor[2] = Tempf[2] * 10;
    temp->TempSensor[3] = Tempf[3] * 10;

    return 0x0;
}

void MX_TIM4_Init(void)
{
    // gpio
    GPIO_InitTypeDef GPIO_InitStruct;
    GPIO_InitStruct.Pin = OWPI;
    GPIO_InitStruct.Mode = GPIO_MODE_OUTPUT_PP;
    GPIO_InitStruct.Pull = GPIO_NOPULL;
    GPIO_InitStruct.Speed = GPIO_SPEED_LOW;
    GPIO_InitStruct.Alternate = GPIO_AF2_TIM4;
    HAL_GPIO_Init(OWPO, &GPIO_InitStruct);
    __TIM4_CLK_ENABLE();

    // structinits
}

```

```

TIM_ClockConfigTypeDef sClockSourceConfig;
TIM_SlaveConfigTypeDef sSlaveConfig;
TIM_MasterConfigTypeDef sMasterConfig;
TIM_IC_InitTypeDef sConfigIC;

// timer setup
htim4.Instance = TIM4;
htim4.Init.Prescaler = 64;//256;
htim4.Init.CounterMode = TIM_COUNTERMODE_UP;
htim4.Init.Period = 0xffff;
htim4.Init.ClockDivision = TIM_CLOCKDIVISION_DIV1;
HAL_TIM_Base_Init(&htim4);

// clock setup
sClockSourceConfig.ClockSource = TIM_CLOCKSOURCE_INTERNAL;
HAL_TIM_ConfigClockSource(&htim4, &sClockSourceConfig);

// init timer
HAL_TIM_IC_Init(&htim4);

// slave and master mode setup
sSlaveConfig.SlaveMode = TIM_SLAVEMODE_TRIGGER;
sSlaveConfig.InputTrigger = TIM_TS_ITR0;
HAL_TIM_SlaveConfigSynchronization(&htim4, &sSlaveConfig);

sMasterConfig.MasterOutputTrigger = TIM_TRGO_RESET;
sMasterConfig.MasterSlaveMode = TIM_MASTERSLAVEMODE_DISABLE;
HAL_TIMEx_MasterConfigSynchronization(&htim4, &sMasterConfig);

// channel 3 setup
sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_RISING;
sConfigIC.ICSelection = TIM_ICSELECTION_INDIRECTTI;
sConfigIC.ICPrescaler = TIM_ICPSC_DIV1;
sConfigIC.ICFilter = 0;
HAL_TIM_IC_ConfigChannel(&htim4, &sConfigIC, TIM_CHANNEL_3);

// channel 4 setup
sConfigIC.ICPolarity = TIM_INPUTCHANNELPOLARITY_FALLING;
sConfigIC.ICSelection = TIM_ICSELECTION_DIRECTTI;
HAL_TIM_IC_ConfigChannel(&htim4, &sConfigIC, TIM_CHANNEL_4);

HAL_NVIC_SetPriority(TIM4_IRQn, 0xE, 0);
HAL_NVIC_EnableIRQ(TIM4_IRQn);
HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_3);
HAL_TIM_IC_Start_IT(&htim4, TIM_CHANNEL_4);
PrintF("Timer Interrupt On");
}

void HAL_TIM_IC_CaptureCallback(TIM_HandleTypeDef *htim)
{
    if(TIM_CNT == 0) TIM4->CNT = 0;
    if (htim == &htim4 && htim->Channel == HAL_TIM_ACTIVE_CHANNEL_3) // hooge flank
    {
        TIM_Val[TIM_CNT][0] = HAL_TIM_ReadCapturedValue(&htim4, TIM_CHANNEL_3);
        TIM_Val[TIM_CNT][1] = 1;
    }
    if (htim == &htim4 && htim->Channel == HAL_TIM_ACTIVE_CHANNEL_4) // lage flank
    {

        TIM_Val[TIM_CNT][0] = HAL_TIM_ReadCapturedValue(&htim4, TIM_CHANNEL_4);
        TIM_Val[TIM_CNT][1] = 0;
    }
    TIM_CNT++;
    if(TIM_CNT == 13) // onewire error
    {
        OW_Error = 0xff;
        TIM_CNT = 0;
    }
}
}

/***** Uart Functions *****/
void CC_UART_Printf(unsignedchar data)
{
    while (!(USART6->ISR& USART_ISR_TXE));
    USART6->TDR = data;
}
void CC_UART_Printf(unsignedchar *data)
{
    while (*data)
    {
        CC_UART_Printf(*data);
        data++;
    }
}
void CC_UART_Printfl(unsignedchar *data, unsignedint len)
{
    unsignedint i;
}

```

```

        for(i = 0; i < len; i++)
    {
        CC_UART_Printf(*data);
        data++;
    }
}

voidCC_UART_Init(void)
{
    GPIO_InitTypeDef GPIO_InitStruct;
    __USART6_CLK_ENABLE();
    __GPIOC_CLK_ENABLE();
    // UART TX GPIO pin configuration
    GPIO_InitStruct.Pin      = TXPI;
    GPIO_InitStruct.Mode     = GPIO_MODE_AF_PP;
    GPIO_InitStruct.Pull     = GPIO_PULLUP;
    GPIO_InitStruct.Speed    = GPIO_SPEED_HIGH;
    GPIO_InitStruct.Alternate = UARTAF;

    HAL_GPIO_Init(UARTPO, &GPIO_InitStruct);

    // UART RX GPIO pin configuration
    GPIO_InitStruct.Pin      = RXPI;
    GPIO_InitStruct.Alternate = UARTAF;
    GPIO_InitStruct.Mode     = GPIO_MODE_AF_PP;
    HAL_GPIO_Init(UARTPO, &GPIO_InitStruct);

    // enable and set clock source
    USART6->CR1 = USART_CR1_UE | USART_CR1_TE | USART_CR1_RE;

    USART6->BRR = (uint16_t)(UART_DIV_SAMPLING16(HAL_RCC_GetPCLK2Freq(), 9600));

    HAL_NVIC_SetPriority(USART6_IRQn, 0xE, 0);
    HAL_NVIC_EnableIRQ(USART6_IRQn);
    USART6->CR1 |= USART_CR1_RXNEIE;
    UART_State = UART_OKE;
    UART_CNT = 0;
    UART_Send = 255;
}

/********************* \
 * Climat Control Cabinet Functions
\*********************/
voidCC_FormString(char *str, short temp)
{
    char ito[6];
    itoa(temp, ito, 10);
    int len = strlen(&ito);
    switch(len)
    {
        case 1:
        {
            str[0] = '0';
            str[1] = '0';
            str[2] = '0';
            str[3] = '0';
            str[4] = '.';
            str[5] = ito[0];
        }
        break;
        case 2:
        {
            str[0] = '0';
            str[1] = '0';
            str[2] = '0';
            str[3] = ito[0];
            str[4] = '.';
            str[5] = ito[1];
        }
        break;
        case 3:
        {
            str[0] = '0';
            str[1] = '0';
            str[2] = ito[0];
            str[3] = ito[1];
            str[4] = '.';
            str[5] = ito[2];
        }
        break;
        case 4:
        {
            str[0] = '0';
            str[1] = ito[0];
            str[2] = ito[1];
            str[3] = ito[2];
            str[4] = '.';
            str[5] = ito[3];
        }
    }
}

```



```

        buf[0] = UART_RXData[7];
        buf[1] = UART_RXData[8];
        buf[2] = UART_RXData[9];
        buf[3] = UART_RXData[10];
        buf[4] = UART_RXData[12];
        buf[5] = '\0';
        temp->Tempkast = atoi(buf);
        buf[0] = UART_RXData[0];
        buf[1] = UART_RXData[1];
        buf[2] = UART_RXData[2];
        buf[3] = UART_RXData[3];
        buf[4] = UART_RXData[5];
        buf[5] = '\0';
        settemp = atoi(buf);
        printf("VT4002 Read: Oke");
        UART_State = UART_OKE;

    }
else
{
    temp->Tempkast = -500;
    printf("VT4002 Read: Error");
    UART_State = UART_DATAERROR;
}
for(i = 0; i < 100; i++) UART_RXData[i] = 0;
return settemp;
}
/*char CC_CAB_Start(void) // deze functies zijn disabled de kast kan alleen handmatig worden gestart
{
    while(UART_Send == 0) osDelay(10);
    int timeout = 0;
    CC_UART_Printf("$00P0001\r");
    while(UART_State != UART_RXDONE)
    {
        osDelay(1);
        timeout++;
        if(timeout == UART_Timeout)
        {
            printf("VT4002 Start: Timeout");
            UART_State = UART_TIMEOUT;
            return UART_State;
        }
    if(strncmp((const char *)UART_RXData, "0", 1) == 0)
    {
        printf("VT4002 Start: Oke");
        UART_State = UART_OKE;
        return UART_State;
    }
    else
    {
        printf("VT4002 Start: Error");
        UART_State = UART_DATAERROR;
        return UART_State;
    }
}
/*char CC_CAB_Stop(void)
{
    while(UART_Send == 0) osDelay(10);
    int timeout = 0;
    CC_UART_Printf("$00P0000\r");
    while(UART_State != UART_RXDONE)
    {
        osDelay(1);
        timeout++;
        if(timeout == UART_Timeout)
        {
            printf("VT4002 Stop: Timeout");
            UART_State = UART_TIMEOUT;
            return UART_State;
        }
    if(strncmp((const char *)UART_RXData, "0", 1) == 0)
    {
        printf("VT4002 Stop: Oke");
        UART_State = UART_OKE;
        return UART_State;
    }
    else
    {
        printf("VT4002 Stop: Error");
        UART_State = UART_DATAERROR;
        return UART_State;
    }
}
char CC_CAB_CheckAlarm(Temp *temp)
{

```

```

int i;
unsignedchar buf[10];
for(i = 0; i < 4; i++)
{
    if(CC_Setings.AlarmRelayOn& (1 << i)&& temp->TempSensor[i] > CC_Setings.AlarmTemp[i] &&
Alarmbuf[i] == 0)
    {
        PC_SendAlarm(temp->TempSensor[i], i);
        Alarmbuf[i] = 1;
        osDelay(400);
    }
    else
    {
        if(temp->TempSensor[i] < (CC_Setings.AlarmTemp[i] - 10)) Alarmbuf[i] = 0;
    }
    if(CC_Setings.AlarmRelayOn& (1 << (i + 4))&& temp->TempSensor[i] > CC_Setings.WarnTemp[i])
    {
        //set relay
        CC_WriteOutput(i, 1);
    }
    else
    {
        //CC_WriteOutput(i, 0); // nietuitzetten !
    }
}
return 0;
}
//*****************************************************************************
* Interrupts
//****************************************************************************/
void USART6_IRQHandler(void)
{
    if (USART6->ISR& USART_ISR_ORE)
    {
        USART6->ICR |= USART_ICR_ORECF;
    }
    if(USART6->ISR& USART_ISR_RXNE)
    {
        UART_RXBuf[UART_CNT] = USART6->RDR;
        UART_CNT++;
        if (UART_CNT == 149) UART_CNT = 0;
        if(USART6->RDR == 13)
        {
            int i;
            for(i = 0; i < UART_CNT; i++) UART_RXData[i] = UART_RXBuf[i];
            i++;
            UART_RXData[i] = '\0';
            UART_State = UART_RXDONE;
            UART_CNT = 0;
        }
        USART6->RQR |= USART_RQR_RXFRQ;
    }
}

```

```

/*
 * File: ClimateModule.h
 * Project: Climate Chamber Control
 * Unit: Main Interface
 * Goal: To control the addon board
 * Platform: STM32F7-Disco
 * Created: 20-11-2015
 * Auhtor: D. Keekstra
 * Company: Brunelco
 */
#ifndef APPLICATION_USER_MODULES_CLIMATEMODULE_H_
#define APPLICATION_USER_MODULES_CLIMATEMODULE_H_
/* Includes
\*****
#include"stm32f756xx.h"
#include"stm32f7xx_hal.h"
#include"stm32746g_discovery.h"
\*****
 * Pinning:
\*****
// relays
#define RL1PO GPIOG
#define RL1PI GPIO_PIN_6
#define RL2PO GPIOB
#define RL2PI GPIO_PIN_4
#define RL3PO GPIOG
#define RL3PI GPIO_PIN_7
#define RL4PO GPIOI
#define RL4PI GPIO_PIN_0
// Debugleds
#define DEB1PO GPIOH
#define DEB1PI GPIO_PIN_6
#define DEB2PO GPIOI
#define DEB2PI GPIO_PIN_3
#define DEB3PO GPIOI
#define DEB3PI GPIO_PIN_2
#define DEB4PO GPIOA
#define DEB4PI GPIO_PIN_15
// Inputs
#define INP1PO GPIOA
#define INP1PI GPIO_PIN_8
#define INP2PO GPIOB
#define INP2PI GPIO_PIN_15
#define INP3PO GPIOB
#define INP3PI GPIO_PIN_14
#define INP4PO GPIOI
#define INP4PI GPIO_PIN_1
//OneWire
#define OWPO GPIOB
#define OWPI GPIO_PIN_9
//Aref
#define REFPO GPIOA
#define REFPI GPIO_PIN_0
// Uart
#define UARTPO GPIOC
#define TXPI GPIO_PIN_6
#define RXPI GPIO_PIN_7
#define UARTAF GPIO_AF8_USART6
\*****
 * Defines:
\*****
// MAX6691
#define OW_Rser 620 // serie resistor
#define OW_Rext 3900 // external resistor
#define NTC_B25 3560 // Beta
#define NTC_R25 2000 // resistance @ 25 graden
#define NTC_T25 298.15 // 25 graden in kelvin
// Flash
#define SizeOfPoint 16 // grote van een data point
#define PointOffset 0x0000ffff // offset om te beginnen te schrijven
// Debug leds
#define DEB1 1
#define DEB2 2
#define DEB3 3
#define DEB4 4
// UART state
#define UART_DEINIT -4
#define UART_DATAERROR -3
#define UART_TIMEOUT -2
#define UART_Busy -1
#define UART_OKE 0
#define UART_RXDONE 1
#define UART_Timeout 200 //ms

```

```

#define CC_STOP 0
#define CC_MANUAL 1
#define CC_PROGRAM 2

/* **** */
/* Globals */
\****

struct Setpoint
{
    unsigned short Length;
    short Temp;
    unsigned char Steep;
    unsigned char Send;
};

struct Logpoint
{
    unsigned short TempSensor[4];
    unsigned short Tempkast;
    unsigned char IO;
    unsigned char Status;
    unsigned char PM;
    unsigned char Hours;
    unsigned char Minutes;
    unsigned char Seconds;
    unsigned short CRC16;
};

struct Settings
{
    unsigned short WarnTemp[4];
    unsigned short AlarmTemp[4];
    unsigned char AlarmRelayOn;
};

typedef struct
{
    // temperatures from -40 - +130 * 10
    // 115.4 = 1154
    short TempSensor[4];
    short Tempkast;
} Temp;
Temp CurrentTemp;

//char ChannelName[][] = {"Sensor 1: "}, {"Sensor 2: "}, {"Sensor 3: "}, {"Sensor 4: "};
volatile unsigned char CC_State;
volatile struct Setpoint CC_Setpoints[100];
volatile unsigned char CC_SetpointCount;
volatile unsigned int CC_Update_Counter;

volatile struct Setpoint CC_Manual;
volatile struct Settings CC_Setings;

TIM_HandleTypeDef htim4;
UART_HandleTypeDef UartHandle;

/* **** */
/* Function declarations */
\****

void CC_Init(void);
// basic functional
void CC_ReadTemperatures(void);
void CC_Run_Manual(char Start);
void CC_Run_Program(char Start);
void CC_Stop(void);

// GPIO
void CC_Run_Test(void);
unsigned char CC_ReadInputs(void);
void CC_WriteOutputs(unsigned char Out);

// Cabinet
char CC_CAB_SetPoint(short Temp);
void CC_Set_Steep(unsigned char Steepness);
char CC_CAB_Start(void);
char CC_CAB_Stop(void);
short CC_CAB_ReadRoomTemp(Temp *temp);
char CC_CAB_CheckAlarm(Temp *temp);

// Uart
void CC_UART_Printf(unsigned char data);
void CC_UART_Printf(unsigned char *data);

// debug leds
void CC_DebugLed_On(char LED);
void CC_DebugLed_Off(char LED);
void CC_DebugLed_Toggle(char LED);
// OW
unsigned char CC_ReadOW(Temp *temp);
\****

#endif/* APPLICATION_USER_MODULES_CLIMATEMODULE_H_ */

```

```

/************************************************************************\
* File:          ClimateModule_WIN.h
* Project:       Climat Control
* Unit:          Interface
* Goal:          to display the GUI elements
*
* Platform:     ST32F746
* Created:      17-12-2015
* Author:        DanjeKeekstra
* Company:      Brunelco
\************************************************************************/
#ifndef APPLICATION_USER_MODULES_ClimateModule_WIN_H_
#define APPLICATION_USER_MODULES_ClimateModule_WIN_H_

/************************************************************************\
* Includes
\************************************************************************/
#include "DebugWindow.h"
#include "ClimateModule.h"
#include "DIALOG.h"
#include "Images.h"
/************************************************************************\
* Constant declarations
\************************************************************************/

/************************************************************************\
* Functions Declarations
\************************************************************************/
void CC_WIN_Start(void);
void CC_WIN_UIO(unsignedchar IO);
void CC_WIN_UTemp(Temp *temp);
void CC_WIN_UTime(void);
void CC_WIN_UIP(char *IP);
void CC_WIN_UIO(unsignedchar IO);
void CC_WIN_UMode(unsignedchar mode, short settemp);
/************************************************************************\
* Globals
\************************************************************************/

#endif/* APPLICATION_USER_MODULES_ClimateModule_WIN_H_ */

```

```

*****\

* File:          ClimateModule_WIN.C
* Project:       Climat Control
* Unit:          Interface
* Goal:          to display the GUI elements
*
* Platform:      ST32F746
* Created:       17-12-2015
* Author:         DanielKeekstra
* Company:       Brunelco
\*****\

/*****\
* Includes
\*****\
#include "ClimateModule_WIN.h"
/*****\
* Defines
\*****\
#define ID_WINDOW_0 (GUI_ID_USER + 0x00)

#define SDIMG          (GUI_ID_USER + 0x01)
#define CLOCK         (GUI_ID_USER + 0x02)
#define IPBAR         (GUI_ID_USER + 0x03)

#define CH1RL          (GUI_ID_USER + 0x04)
#define CH1INP         (GUI_ID_USER + 0x05)
#define CH2RL          (GUI_ID_USER + 0x06)
#define CH2INP         (GUI_ID_USER + 0x07)
#define CH3RL          (GUI_ID_USER + 0x08)
#define CH3INP         (GUI_ID_USER + 0x09)
#define CH4RL          (GUI_ID_USER + 0x0A)
#define CH4INP         (GUI_ID_USER + 0x0B)

#define CH1BAR         (GUI_ID_USER + 0x0C)
#define CH2BAR         (GUI_ID_USER + 0x0D)
#define CH3BAR         (GUI_ID_USER + 0x0E)
#define CH4BAR         (GUI_ID_USER + 0x0F)

#define CABBAR         (GUI_ID_USER + 0x10)

#define TX_SENS1        (GUI_ID_USER + 0x11)
#define TX_SENS2        (GUI_ID_USER + 0x12)
#define TX_SENS3        (GUI_ID_USER + 0x13)
#define TX_SENS4        (GUI_ID_USER + 0x14)

#define TX_Time          (GUI_ID_USER + 0x15)
#define TX_Date          (GUI_ID_USER + 0x16)
#define TX_IP            (GUI_ID_USER + 0x17)
#define TX_CAB           (GUI_ID_USER + 0x18)

#define SET_IMG          (GUI_ID_USER + 0x19)
#define SET_TEMP         (GUI_ID_USER + 0x1A)
#define SET_MOD          (GUI_ID_USER + 0x1B)

//_____
#define P_Xoffset        10
#define P_Yoffset        10
#define P_YStepoffset    35
#define P_XStepoffset    5

// dialog
static const GUI_WIDGET_CREATE_INFO _aDialogCreate[] = {
    // Window
    { WINDOW_CreateIndirect, "Window", ID_WINDOW_0,           0, 0, 480, 272, 0, 0, 0 },

    // clock
    { IMAGE_CreateIndirect, "Clock",     CLOCK,                326, 0, 154, 272, 0, 0, 0 },
    { IMAGE_CreateIndirect, "ipbar",     IPBAR,               142, 252, 196, 20, 0, 0, 0 },

    // sensor bars
    { IMAGE_CreateIndirect, "Cab bar4", CABBAR,             P_Xoffset, (P_Yoffset + 0 * P_YStepoffset), 200, 30,
0, 0, 0 }, { IMAGE_CreateIndirect, "Sensor bar1", CH1BAR,   P_Xoffset, (P_Yoffset + 1 * P_YStepoffset),
200, 30, 0, 0, 0 }, { IMAGE_CreateIndirect, "Sensor bar2", CH2BAR,   P_Xoffset, (P_Yoffset + 2 * P_YStepoffset),
200, 30, 0, 0, 0 }, { IMAGE_CreateIndirect, "Sensor bar3 ", CH3BAR,   P_Xoffset, (P_Yoffset + 3 * P_YStepoffset), 200, 30,
0, 0, 0 }, { IMAGE_CreateIndirect, "Sensor bar4", CH4BAR,   P_Xoffset, (P_Yoffset + 4 * P_YStepoffset),
200, 30, 0, 0, 0 }

    // Indicators

```

```

        { IMAGE_CreateIndirect, "ImageOff", CH1RL,
+ 1 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOn", CH1INP,
(P_Yoffset + 1 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOff", CH2RL,
+ 2 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOn", CH2INP,
(P_Yoffset + 2 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOff", CH3RL,
+ 3 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOn", CH3INP,
(P_Yoffset + 3 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOff", CH4RL,
+ 4 * P_YStepoffset), 30, 30, 0, 0, 0 },
        { IMAGE_CreateIndirect, "ImageOn", CH4INP,
(P_Yoffset + 4 * P_YStepoffset), 30, 30, 0, 0, 0 },

        // Temperatures Text
        { TEXT_CreateIndirect, "Cabinet : ", TX_CAB,
180, 30, 0, 0, 0 },
        { TEXT_CreateIndirect, "Sensor 1: ", TX_SENS1,
P_YStepoffset), 180, 30, 0, 0, 0 },
        { TEXT_CreateIndirect, "Sensor 2: ", TX_SENS2,
P_YStepoffset), 180, 30, 0, 0, 0 },
        { TEXT_CreateIndirect, "Sensor 3: ", TX_SENS3,
P_YStepoffset), 180, 30, 0, 0, 0 },
        { TEXT_CreateIndirect, "Sensor 4: ", TX_SENS4,
P_YStepoffset), 180, 30, 0, 0, 0 }

        // Info text
        { TEXT_CreateIndirect, "Time: ", TX_Time,
{ TEXT_CreateIndirect, "Date: ", TX_Date,
{ TEXT_CreateIndirect, "IP: ", TX_IP,

        //SD Card img
        { IMAGE_CreateIndirect, "SD Card Image", SDIMG,
350, 222, 46, 50, 0, 0, 0 },

        // setting img
        { IMAGE_CreateIndirect, "Settings image", SET_IMG,
{ TEXT_CreateIndirect, "Offline", SET_MOD,
{ TEXT_CreateIndirect, "", SET_TEMP,
345, 74, 140, 40, 0, 0, 0 },
365, 115, 80, 20, 0, 0, 0 },
190, 254, 108, 15, 0, 0, 0 },

};

//*****************************************************************************
* Functions Declarations
//*****************************************************************************
staticvoid_cbDialog(WM_MESSAGE * pMsg);
//*****************************************************************************
* Globals
//*****************************************************************************
volatileunsignedchar CC_IO = 255;
unsignedchar bufm = 0;
unsignedshort bufs = 0;
//*****************************************************************************
* Public Functions
//*****************************************************************************
voidCC_WIN_Start(void)
{
    CCWin = GUI_CreateDialogBox(_aDialogCreate, GUI_COUNTOF(_aDialogCreate), _cbDialog, WM_HBKWIN, 0, 0);
}
voidCC_WIN_UTemp(Temp *temp)
{
    char text[40];
    WM_HWIN hItem;
    short d1;
    short d2;

    hItem = WM_GetDialogItem(CCWin, TX_SENS1);
    if(temp->TempSensor[0] > -420)
    {
        d1 = temp->TempSensor[0] / 10;
        d2 = (temp->TempSensor[0] - d1 * 10);
        if(d2 < 0) d2 = 0 - d2;
        sprintf(text, "Sensor 1: %d.%d °C", d1,d2);
        TEXT_SetText(hItem, text);
    }
    elseTEXT_SetText(hItem, "Sensor 1: NC");

    hItem = WM_GetDialogItem(CCWin, TX_SENS2);
    if(temp->TempSensor[1] > -420)
    {
        d1 = temp->TempSensor[1] / 10;
        d2 = (temp->TempSensor[1] - d1 * 10);
        if(d2 < 0) d2 = 0 - d2;
        sprintf(text, "Sensor 2: %d.%d °C", d1,d2);
    }
}

        P_Xoffset + 200 + P_XStepoffset, (P_Yoffset
P_Xoffset + 200 + 30 + 2 * P_XStepoffset,
P_Xoffset + 200 + P_XStepoffset, (P_Yoffset
P_Xoffset + 200 + 30 + 2 * P_XStepoffset,
P_Xoffset + 200 + P_XStepoffset, (P_Yoffset
P_Xoffset + 200 + 30 + 2 * P_XStepoffset,
P_Xoffset + 200 + P_XStepoffset, (P_Yoffset
P_Xoffset + 200 + 30 + 2 * P_XStepoffset,
P_Xoffset + 200 + P_XStepoffset, (P_Yoffset
P_Xoffset + 200 + 30 + 2 * P_XStepoffset,
P_Xoffset + 10, (P_Yoffset + 2 + 0 * P_YStepoffset),
P_Xoffset + 10, (P_Yoffset + 2 + 1 *
P_Xoffset + 10, (P_Yoffset + 2 + 2 *
P_Xoffset + 10, (P_Yoffset + 2 + 3 *
P_Xoffset + 10, (P_Yoffset + 2 + 4 *

350, 222, 46, 50, 0, 0, 0 },
1, 218, 26, 55, 0, 0, 0 },
35, 222, 100, 20, 0, 0, 0 },
35, 252, 100, 20, 0, 0, 0 },
345, 74, 140, 40, 0, 0, 0 },
365, 115, 80, 20, 0, 0, 0 },
190, 254, 108, 15, 0, 0, 0 },

```

```

        TEXT_SetText(hItem, text);
    }
elseTEXT_SetText(hItem, "Sensor 2: NC");

hItem = WM_GetDialogItem(CCWin, TX_SENS3);
if(temp->TempSensor[2] > -420)
{
    d1 = temp->TempSensor[2] / 10;
    d2 = (temp->TempSensor[2] - d1 * 10);
    if(d2 < 0) d2 = 0 - d2;
    sprintf(text, "Sensor 3: %d.%d °C", d1,d2);
    TEXT_SetText(hItem, text);
}
elseTEXT_SetText(hItem, "Sensor 3: NC");

hItem = WM_GetDialogItem(CCWin, TX_SENS4);
if(temp->TempSensor[3] > -420)
{
    d1 = temp->TempSensor[3] / 10;
    d2 = (temp->TempSensor[3] - d1 * 10);
    if(d2 < 0) d2 = 0 - d2;
    sprintf(text, "Sensor 4: %d.%d °C", d1,d2);
    TEXT_SetText(hItem, text);
}
elseTEXT_SetText(hItem, "Sensor 4: NC");
hItem = WM_GetDialogItem(CCWin, TX_CAB);
if(temp->Tempkast > -420)
{
    d1 = temp->Tempkast / 10;
    d2 = (temp->Tempkast - d1 * 10);
    if(d2 < 0) d2 = 0 - d2;
    sprintf(text, "Cabinet: %d.%d °C", d1,d2);
    TEXT_SetText(hItem, text);
}
elseTEXT_SetText(hItem, "Cabinet: NC");
}
voidCC_WIN_UTime(void)
{
    WM_HWIN hItem;
    RTC_TimeTypeDef time;
    RTC_DateTypeDef date;
    k_GetTime(&time);
    k_GetDate(&date);
    char text[30];
    // set time
    hItem = WM_GetDialogItem(CCWin, TX_Time);
    if(time.Minutes< 10) sprintf(text, "%d:0%0d", time.Hours, time.Minutes);
    else sprintf(text, "%d:%0d", time.Hours, time.Minutes);
    TEXT_SetText(hItem, text);
    // set date
    hItem = WM_GetDialogItem(CCWin, TX_Date);
    sprintf(text, "%d-%d-%d", date.Date, date.Month, date.Year);
    TEXT_SetText(hItem, text);
}
voidCC_WIN_UIP(char *IP)
{
    WM_HWIN hItem = WM_GetDialogItem(CCWin, TX_IP);
    TEXT_SetText(hItem, IP);
}
voidCC_WIN_UIO(unsignedchar IO)
{
    WM_HWIN hItem;
    if(CC_IO == IO) return;
    CC_IO = IO;

    hItem = WM_GetDialogItem(CCWin, CH1RL);
    if((CC_IO >> 0) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
    hItem = WM_GetDialogItem(CCWin, CH1INP);
    if((CC_IO >> 4) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

    hItem = WM_GetDialogItem(CCWin, CH2RL);
    if((CC_IO >> 1) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
    hItem = WM_GetDialogItem(CCWin, CH2INP);
    if((CC_IO >> 5) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

    hItem = WM_GetDialogItem(CCWin, CH3RL);
    if((CC_IO >> 2) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
    hItem = WM_GetDialogItem(CCWin, CH3INP);
    if((CC_IO >> 6) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
}

```

```

    hItem = WM_GetDialogItem(CCWin, CH4RL);
    if((CC_IO >> 3) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
    hItem = WM_GetDialogItem(CCWin, CH4INP);
    if((CC_IO >> 7) & 1) IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
    else IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
}
void CC_WIN_UMode(unsignedchar mode, short settemp)
{
    WM_HWIN      hItem;
    // set SD and modus
    if(mode != bufm)
    {
        bufm = mode;
        switch(mode)
        {
            case CC_STOP:
                hItem = WM_GetDialogItem(CCWin, SDIMG);
                IMAGE_SetBMP(hItem, IMG_SDReady, sizeof(IMG_SDReady));

                hItem = WM_GetDialogItem(CCWin, SET_MOD);
                TEXT_SetTextColor(hItem, 0x00999999);
                TEXT_SetText(hItem, "Offline");
                TEXT_SetFont(hItem, GUI_FONT_20_1);

            break;
            case CC_MANUAL:
                hItem = WM_GetDialogItem(CCWin, SDIMG);
                IMAGE_SetBMP(hItem, IMG_SDWrite, sizeof(IMG_SDWrite));

                hItem = WM_GetDialogItem(CCWin, SET_MOD);
                TEXT_SetTextColor(hItem, 0x00999999);
                TEXT_SetText(hItem, "Manual");
                TEXT_SetFont(hItem, GUI_FONT_20_1);

            break;
            case CC_PROGRAM:
                hItem = WM_GetDialogItem(CCWin, SDIMG);
                IMAGE_SetBMP(hItem, IMG_SDWrite, sizeof(IMG_SDWrite));

                hItem = WM_GetDialogItem(CCWin, SET_MOD);
                TEXT_SetTextColor(hItem, 0x00999999);
                TEXT_SetText(hItem, "Profile");
                TEXT_SetFont(hItem, GUI_FONT_20_1);

            break;
        }
    }
    if(settemp != bufs)
    {
        bufs = settemp;
        if(settemp > -40) // nc
        {
            short d1, d2;
            char text[20];
            d1 = settemp / 10;
            d2 = (settemp - d1 * 10);
            if(d2 < 0) d2 = 0 - d2;
            sprintf(text, "%d.%d °C", d1,d2);

            hItem = WM_GetDialogItem(CCWin, SET_TEMP);
            TEXT_SetTextColor(hItem, 0x00999999);
            TEXT_SetText(hItem, text);
            TEXT_SetFont(hItem, GUI_FONT_20_1);
        }
        else
        {
            hItem = WM_GetDialogItem(CCWin, SET_TEMP);
            TEXT_SetTextColor(hItem, 0x00999999);
            TEXT_SetText(hItem, "");
            TEXT_SetFont(hItem, GUI_FONT_20_1);
        }
    }
}

/*****************************************
 * Private Functions
 \*****************************************/
static void _cbDialog(WM_MESSAGE * pMsg)
{
    WM_HWIN      hItem;
    int         NCode;
    int         Id;

    switch (pMsg->MsgId) {
    case WM_INIT_DIALOG:
        // Initialization of 'Window'

```

```

hItem = pMsg->hWin;
WINDOW_SetBkColor(hItem, 0x00000000);

//Initialization of 'clock'
hItem = WM_GetDialogItem(pMsg->hWin, CLOCK);
IMAGE_SetBMP(hItem, IMG_Clock, sizeof(IMG_Clock));
// Initialization of 'ipbar'
hItem = WM_GetDialogItem(pMsg->hWin, IPBAR);
IMAGE_SetBMP(hItem, IMG_IPBar, sizeof(IMG_IPBar));

// set buttons
hItem = WM_GetDialogItem(pMsg->hWin, CH1RL);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
hItem = WM_GetDialogItem(pMsg->hWin, CH1INP);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

hItem = WM_GetDialogItem(pMsg->hWin, CH2RL);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
hItem = WM_GetDialogItem(pMsg->hWin, CH2INP);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

hItem = WM_GetDialogItem(pMsg->hWin, CH3RL);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
hItem = WM_GetDialogItem(pMsg->hWin, CH3INP);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

hItem = WM_GetDialogItem(pMsg->hWin, CH4RL);
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));
hItem = WM_GetDialogItem(pMsg->hWin, CH4INP);
IMAGE_SetBMP(hItem, IMG_On, sizeof(IMG_On));
IMAGE_SetBMP(hItem, IMG_Off, sizeof(IMG_Off));

// Initialization of 'sens1 bar'
hItem = WM_GetDialogItem(pMsg->hWin, CH1BAR);
IMAGE_SetBMP(hItem, IMG_SensorBar, sizeof(IMG_SensorBar));
// Initialization of 'sens2 bar'
hItem = WM_GetDialogItem(pMsg->hWin, CH2BAR);
IMAGE_SetBMP(hItem, IMG_SensorBar, sizeof(IMG_SensorBar));
// Initialization of 'sens3 bar'
hItem = WM_GetDialogItem(pMsg->hWin, CH3BAR);
IMAGE_SetBMP(hItem, IMG_SensorBar, sizeof(IMG_SensorBar));
// Initialization of 'sens4 bar'
hItem = WM_GetDialogItem(pMsg->hWin, CH4BAR);
IMAGE_SetBMP(hItem, IMG_SensorBar, sizeof(IMG_SensorBar));

// Initialization of 'sens4 bar'
hItem = WM_GetDialogItem(pMsg->hWin, CABBAR);
IMAGE_SetBMP(hItem, IMG_SensorBar, sizeof(IMG_SensorBar));

// Initialization of 'Sensor 1 text'
hItem = WM_GetDialogItem(pMsg->hWin, TX_SENS1);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "Sensor 1: ");
TEXT_SetFont(hItem, GUI_FONT_24_1);
// Initialization of 'Sensor 2 text'
hItem = WM_GetDialogItem(pMsg->hWin, TX_SENS2);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "Sensor 2: ");
TEXT_SetFont(hItem, GUI_FONT_24_1);
// Initialization of 'Sensor 3 text'
hItem = WM_GetDialogItem(pMsg->hWin, TX_SENS3);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "Sensor 3: ");
TEXT_SetFont(hItem, GUI_FONT_24_1);
// Initialization of 'Sensor 4 text'
hItem = WM_GetDialogItem(pMsg->hWin, TX_SENS4);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "Sensor 4: ");
TEXT_SetFont(hItem, GUI_FONT_24_1);

// Initialization of 'Time'
hItem = WM_GetDialogItem(pMsg->hWin, TX_Time);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "22:11");
TEXT_SetFont(hItem, GUI_FONT_D32);
// Initialization of 'Date'
hItem = WM_GetDialogItem(pMsg->hWin, TX_Date);
TEXT_SetTextColor(hItem, 0x00999999);
TEXT_SetText(hItem, "21-12-15");
TEXT_SetFont(hItem, GUI_FONT_20_1);
// Initialization of 'IP text'
hItem = WM_GetDialogItem(pMsg->hWin, TX_IP);
TEXT_SetTextColor(hItem, 0x003333FF);
TEXT_SetText(hItem, "0.0.0.0");
TEXT_SetFont(hItem, GUI_FONT_16_ASCII);

```

```

// Initialization of 'Cabinet'
hItem = WM_GetDialogItem(pMsg->hWin, TX_CAB);
TEXT_SetTextColor(hItem, 0x00FF5E00);
TEXT_SetText(hItem, "Cabinet: ");
TEXT_SetFont(hItem, GUI_FONT_24_1);

// Initialization of 'SD image'
hItem = WM_GetDialogItem(pMsg->hWin, SDIMG);
IMAGE_SetBMP(hItem, IMG_SDReady, sizeof(IMG_SDReady));

// Initialization of 'Settings'
hItem = WM_GetDialogItem(pMsg->hWin, SET_IMG);
IMAGE_SetBMP(hItem, IMG_Settings, sizeof(IMG_Settings));

hItem = WM_GetDialogItem(pMsg->hWin, SET_MOD);
TEXT_SetTextColor(hItem, 0x00999999);
TEXT_SetText(hItem, "Offline");
TEXT_SetFont(hItem, GUI_FONT_20_1);

hItem = WM_GetDialogItem(pMsg->hWin, SET_TEMP);
TEXT_SetTextColor(hItem, 0x00999999);
TEXT_SetText(hItem, "");
TEXT_SetFont(hItem, GUI_FONT_20_1);

break;
case WM_NOTIFY_PARENT:
    Id    = WM_GetId(pMsg->hWinSrc);
    NCode = pMsg->Data.v;

break;
default:
    WM_DefaultProc(pMsg);
    break;
}
} //*****

```

```

*****\n
* File:          Ethernet.c\n
* Project:       Climate Chamber Control\n
* Unit:          Main Interface\n
* Goal:          To easily make a TCP/IP ethernetconnection\n
*                 it can be init with just Ethernet_Init(..)\n
*                 sending a packet can be done with the SER_OUT structure\n
*                 receive and anser can be done with a call back function\n
* Platform:      STM32F7-Disco\n
* Created:       22-10-2015\n
* Author:        D. KEEKSTRA\n
* Company:       Brunelco\n
*****\n
* Includes\n
\*****\n
#include "Ethernet.h"\n
\*****\n
* Functions Declarations\n
\*****\n
voidTCP_Thread(voidconst * argument);\n
staticvoidNetif_Config(void);\n
voidTCP_Thread(voidconst * argument);\n
\n// TCP server\n
voidTCP_Init(unsignedshort PORT);\n
staticvoidTCP_Close(structtcp_pcb *pcb);\n
\nstaticerr_tTCP_Accept(void *arg, structtcp_pcb *pcb, err_t err);\n
staticerr_tTCP_Receive(void *arg, structtcp_pcb *pcb, structpbuf *p, err_t err);\n
staticerr_tTCP_Poll(void *arg, structtcp_pcb *pcb);\n
staticerr_tTCP_Err(void *arg, err_t err);\n
\nstaticvoidTCP_Close(structtcp_pcb *pcb);\n
\n*****\n
* Globals\n
\*****\n
structnetif gnetif; // network interface structure\n
unsignedchar TCP_State = TCP_IDLE;\n
osThreadId TCP_ThreadId = 0;\n
\nchar MAX_DHCP_TRIES;\n
unsignedshort PORT;\n
\nerr_t err;\n
osTimerId WriteID;\n
\n*****\n
* Public Functions\n
\*****\n
voidEthernet_Init(unsignedshort port, char DHCP_tries)\n{
    PORT = port;\n
    MAX_DHCP_TRIES = DHCP_tries;\n
    //PrintF("Starting Ethernet Module.");\n
    //PrintL("Initialize hardware");\n
    tcpip_init(NULL, NULL);\n
    Netif_Config();\n
    if (netif_is_up(&gnetif))\n    {\n        TCP_State = TCP_LINK_UP;\n    }\n    else\n    {\n        return;\n    }\n    if(TCP_ThreadId == 0)\n    {\n        osThreadDef(TCP, TCP_Thread, osPriorityAboveNormal, 0, 2048);\n        TCP_ThreadId = osThreadCreate (osThread(TCP), NULL);\n    }\n}\nvoidEthernet_DeInit(void)\n{
    TCP_State = TCP_IDLE;\n
    if(TCP_ThreadId != 0)\n    {\n        osThreadTerminate(TCP_ThreadId);\n        TCP_ThreadId = 0;\n    }
}
\n*****\n
* TCP Functions

```

```

\*****
voidTCP_Init(unsignedshort PORT)
{
    SER_OUT.State = SER_DISCONNECTED;
    pcb = tcp_new();
    tcp_bind(pcb, IP_ADDR_ANY, PORT); //server port for incoming connection
    while(1)
    {
        if (SER_OUT.State == SER_DISCONNECTED)
        {
            pcb = tcp_listen(pcb);
            tcp_accept(pcb, TCP_Accept);
        }
        if(SER_OUT.Send&& SER_OUT.State == SER_CONNECTED)
        {
            if(tcp_write(pcb, (void *)SER_OUT.Data, SER_OUT.Length, 1) == ERR_OK)
            {
                tcp_output(pcb);
                SER_OUT.Send = 0;
            }
        }
        osDelay(1);
    }
}
static void TCP_Close(structtcp_pcb *pcb)
{
    tcp_arg(pcb, NULL);
    tcp_sent(pcb, NULL);
    tcp_recv(pcb, NULL);
    tcp_close(pcb);
    SER_OUT.State = SER_DISCONNECTED;
}
static err_t TCP_Accept(void *arg, structtcp_pcb *pcb, err_t err)
{
    LWIP_UNUSED_ARG(arg);
    LWIP_UNUSED_ARG(err);
    tcp_recv(pcb, TCP_Receive);
    tcp_poll(pcb, TCP_Poll, 1); //every 500ms
    tcp_accepted(pcb);
    SER_OUT.State = SER_CONNECTED;
    return ERR_OK;
}
static err_t TCP_Receive(void *arg, structtcp_pcb *pcb, structpbuf *p, err_t err)
{
    SER_OUT.State = SER_BUSY;
    unsignedchar *outbuf;
    unsignedint outlen;
    LWIP_UNUSED_ARG(arg);
    CC_DebugLed_On(DEB2);
    if (err == ERR_OK && p != NULL)
    {
        tcp_recved(pcb, p->tot_len);
        outlen = Callback_Handle_Packet(p->payload, &outbuf, p->tot_len);
        if (tcp_write(pcb, outbuf, outlen, 1) == ERR_OK)
        {
            tcp_output(pcb);
        }
        pbuf_free(p);
    }
    else
    {
        pbuf_free(p);
        TCP_Close(pcb);
    }
    SER_OUT.State = SER_CONNECTED;
    CC_DebugLed_Off(DEB2);
    return ERR_OK;
}
static err_t TCP_Poll(void *arg, structtcp_pcb *pcb)
{
    LWIP_UNUSED_ARG(arg);
    LWIP_UNUSED_ARG(pcb);
    if(SER_OUT.Send&& SER_OUT.State == SER_CONNECTED)
    {
        if(tcp_write(pcb, (void *)SER_OUT.Data, SER_OUT.Length, 1) == ERR_OK)
        {
            tcp_output(pcb);
            SER_OUT.Send = 0;
        }
    }
    return ERR_OK;
}
static err_t TCP_Err(void *arg, err_t err)
{
    LWIP_UNUSED_ARG(arg);
}

```

```

LWIP_UNUSED_ARG(err);
    return ERR_OK;
}
//*****************************************************************************
 * Main Thread
\*****
voidTCP_Thread(voidconst * argument)
{
    while(1)
    {
        switch (TCP_State)
        {
            case TCP_LINK_UP:
            {
                gnetif.ip_addr.addr = 0;
                gnetif.netmask.addr = 0;
                gnetif.gw.addr = 0;
                //PrintF("Starting DHCP");
                dhcp_start(&gnetif);
                TCP_State = TCP_WAIT_FOR_ADDRESS;
            }
            break;
            case TCP_WAIT_FOR_ADDRESS: // wachtenopdhcp
            {
                if (gnetif.ip_addr.addr!=0)
                {
                    dhcp_stop(&gnetif);
                    TCP_State = TCP_START;
                }
                else
                {
                    if (gnetif.dhcp->tries> MAX_DHCP_TRIES)
                    {
                        TCP_State = TCP_ERROR;
                        dhcp_stop(&gnetif);
                        //PrintF("No reply from DHCP Server");
                    }
                }
            }
            break;
            case TCP_START: // dhcpgekregen
            {
                staticuint8_t iptxt[128];
                sprintf((char*)iptxt,
                        "%d.%d.%d.%d",
                        (uint8_t)(gnetif.ip_addr.addr),
                        (uint8_t)((gnetif.ip_addr.addr) >> 8),
                        (uint8_t)((gnetif.ip_addr.addr) >> 16),
                        (uint8_t)((gnetif.ip_addr.addr) >> 24));
                CC_WIN_UIP(iptxt);
                //PrintF((char *)iptxt);
                TCP_State = TCP_PROCESS;
            }
            break;
            case TCP_UDPECHO: // wachtoependress broadcast omtebindenopeenip
            {
                // alshetip wordtgekregen van depcappkanditwordengebruiktomhiermee
                //
            }
            break;
            case TCP_PROCESS: // start server
            {
                TCP_Init(31415);
                TCP_State = TCP_CABLEDISC; // cable is down
            }
            break;
            case TCP_CABLEDISC: // cable down
            {
                while(1) osDelay(10);
            }
            break;
        }
        osDelay(250);
    }
}
//*****************************************************************************
 * HAL Config
\*****
staticvoidNetif_Config(void)
{
    structip_addr ipaddr;
    structip_addr netmask;
    structip_addr gw;
    /* IP address setting */
}

```

```

IP4_ADDR(&ipaddr, IP_ADDR0, IP_ADDR1, IP_ADDR2, IP_ADDR3);
IP4_ADDR(&netmask, NETMASK_ADDR0, NETMASK_ADDR1, NETMASK_ADDR2, NETMASK_ADDR3);
IP4_ADDR(&gw, GW_ADDR0, GW_ADDR1, GW_ADDR2, GW_ADDR3);

/* - netif_add(struct netif *netif, struct ip_addr *ipaddr,
   struct ip_addr *netmask, struct ip_addr *gw,
   void *state, err_t (* init)(struct netif *netif),
   err_t (* input)(struct pbuf *p, struct netif *netif))

Adds your network interface to the netif_list. Allocate a struct
netif and pass a pointer to this structure as the first argument.
Give pointers to cleared ip_addr structures when using DHCP,
or fill them with sane numbers otherwise. The state pointer may be NULL.

The init function pointer must point to a initialization function for
your etherenetnetif interface. The following code illustrates it's use.*/

netif_add(&gnetif, &ipaddr, &netmask, &gw, NULL, &etherenetif_init, &tcpip_input);

/* Registers the default network interface. */
netif_set_default(&gnetif);

if (netif_is_link_up(&gnetif))
{
    /* When the netif is fully configured this function must be called.*/
    netif_set_up(&gnetif);
}
else
{
    /* When the netif link is down this function must be called */
    netif_set_down(&gnetif);
}

/*************\n * Callbacks\n\*****\n
弱 unsignedshort Callback_Handle_Packet(unsignedchar *Inbuf, void **Outbuf, unsignedshort InLen)
{
    *Outbuf = "Ok";
    return 2;
}

```

```

/************************************************************************/
 * File:          Ethernet.h
 * Project:       Climate Chamber Control
 * Unit:         Main Interface
 * Goal:          To easily make a TCP/IP etherenetconnecton
 * Platform:     STM32F7-Disco
 * Created:      22-10-2015
 * Author:        D. Keekstra
 * Company:      Brunelco
 /************************************************************************
#ifndef APPLICATION_USER_MODULES_Ethernet_H_
#define APPLICATION_USER_MODULES_Ethernet_H_
/************************************************************************
 * Includes
/************************************************************************
#include "lwip/netif.h"
#include "DebugWindow.h"
#include "ClimateModule.h"
#include "lwip/opt.h"
#include "lwip/dhcp.h"
#include "lwip/netif.h"
#include "lwip/tcpip.h"
#include "lwip/arch.h"
#include "lwip/tcp.h"
#include "lwip/lwip_timers.h"
#include "etherenetif.h"
#include "lwipopts.h"
#include "string.h"
#include "cmsis_os.h"
#include "stm32746g_discovery.h"
#include "stm32f7xx_hal.h"
#include "stm32f7xx_hal_flash.h"
/************************************************************************
 * Constant declarations
/************************************************************************
/* Static IP ADDRESS */
#define IP_ADDR0           192
#define IP_ADDR1           168
#define IP_ADDR2             0
#define IP_ADDR3            10
/* NETMASK */
#define NETMASK_ADDR0        255
#define NETMASK_ADDR1        255
#define NETMASK_ADDR2        255
#define NETMASK_ADDR3          0
/* Gateway Address */
#define GW_ADDR0           192
#define GW_ADDR1           168
#define GW_ADDR2             0
#define GW_ADDR3             1
/* TCP States */
#define TCP_IDLE              0
#define TCP_INIT               1
#define TCP_LINK_UP            2
#define TCP_WAIT_FOR_ADDRESS      3
#define TCP_ERROR                4
#define TCP_LINK_DOWN            5
#define TCP_UDPECHO              6
#define TCP_PROCESS              7
#define TCP_START                 8
#define TCP_CABLEDISC              9
/* Server States */
#define SER_BUSY                  2
#define SER_CONNECTED              1
#define SER_DISCONNECTED            0
/************************************************************************
 * Function declarations
/************************************************************************
void                     Ethernet_Init(unsignedshort port, char DHCP_tries);
void                     Ethernet_DeInit(void);
unsignedshort   Callback_Handle_Packet(unsignedchar *Inbuf, void **Outbuf, unsignedshort InLen);
/************************************************************************
 * Public Globals
/************************************************************************
volatilestructtcp_pcb *pcb;
structSER_SEND
{
    unsignedcharState;
    unsignedcharSend;
    unsignedchar *Data;
    unsignedcharLength;
};
volatilestructSER_SEND SER_OUT;
/************************************************************************
#endif/* APPLICATION_USER_MODULES_Ethernet_H_ */

```

```

/*
 * File:          Protocol.c
 * Project:       Climate Chamber Control
 * Unit:         Main Interface
 * Goal:        To Handle packets form and to PC Software
 * Platform:     STM32F7-Disco
 * Created:      03-11-2015
 * Author:       D. Keekstra
 * Company:     Brunelco
 */
#include<string.h>
#include<stdlib.h>
#include"ClimateModule.h"
#include"Protocol.h"
#include"Ethernet.h"
/*
 * Function Declarations
 */
voidPC_SendSettings(void);
/*
 * Globals
 */
unsignedchar Packet[100];
/*
 * Public Send Functions
 */
voidPC_SendTemps(Temp *temp, unsignedchar IO)
{
    if(SER_OUT.State == SER_CONNECTED && SER_OUT.Send == 0)
    {
        RTC_TimeTypeDef time;
        RTC_DateTypeDef date;
        k_GetTime(&time);
        k_GetDate(&date);
        // packet
        Packet[0] = 0xF1; // preamble
        Packet[1] = 0xF0; // preamble
        Packet[2] = 0xF1; // function
        Packet[3] = 0x00; // lengt of packet
        Packet[4] = 0x13; // lengt of packet
        // external sensors
        Packet[5] = temp->TempSensor[0] >> 8;
        Packet[6] = temp->TempSensor[0];
        Packet[7] = temp->TempSensor[1] >> 8;
        Packet[8] = temp->TempSensor[1];
        Packet[9] = temp->TempSensor[2] >> 8;
        Packet[10] = temp->TempSensor[2];
        Packet[11] = temp->TempSensor[3] >> 8;
        Packet[12] = temp->TempSensor[3];
        // internal sensor
        Packet[13] = temp->Tempkast>> 8;
        Packet[14] = temp->Tempkast;
        // time and date
        Packet[15] = date.Year;
        Packet[16] = date.Month;
        Packet[17] = date.Date;
        Packet[18] = time.Hours;
        Packet[19] = time.Minutes;
        Packet[20] = time.Seconds;
        // IO
        Packet[21] = IO;
        // set length and start
        SER_OUT.Data = Packet;
        SER_OUT.Length = 22;
        SER_OUT.Send = 1;
    }
}
voidPC_SendAlarm(short temp, char nr)
{
    if(SER_OUT.State == SER_CONNECTED)
    {
        Packet[0] = 0xF1;
        Packet[1] = 0xF0;
        Packet[2] = 0xF0;
        Packet[3] = 0x00;
        Packet[4] = 0x04;
        Packet[5] = nr;
        Packet[6] = 1;
        Packet[7] = temp >> 8;
        Packet[8] = temp;
        SER_OUT.Send = 255;
        SER_OUT.Data = Packet;
    }
}

```

```

        SER_OUT.Length = 9;
    }

} //*****\n * Private Send Functions\n\*****\nvoid PC_SendSettings(void)
{
    /*packet[0] = 0;
    if(SER_OUT.State == SER_CONNECTED)
    {
        if(tcp_write(pcb, (void *), , 1) == ERR_OK)
        {
            tcp_output(pcb);
        }
    }*/
}

} //*****\n * Callback\n\*****\nunsignedshort Callback_Handle_Packet(unsignedchar *Inbuf, void **Outbuf, unsignedshort InLen)
{
    unsignedshort Outlen, length;
    int i = 0;
    // find preamble in data stream
    if(Inbuf[i] == 0xFF && Inbuf[i + 1] == 0xFF)
    {
        length = (Inbuf[i + 3] << 8) + Inbuf[i + 4];
        unsignedchar PC = Inbuf[i + 2];
        switch(PC)
        {
            case PC_ADDSETPOINT:
            {
                if(length == 0x0004)
                {
                    CC_Setpoints[CC_SetpointCount].Length = Inbuf[i + 5] * 60;
                    CC_Setpoints[CC_SetpointCount].Temp = (Inbuf[i + 6] << 8) + Inbuf[i + 7];
                    CC_Setpoints[CC_SetpointCount].Steep = Inbuf[i + 8];
                    CC_SetpointCount++;
                }
            }
            break;
            case PC_SETDATETIME:
            {
                if(length == 0x0005)
                {
                    RTC_DateTypeDef date;
                    date.Year = Inbuf[i + 5];
                    date.Month = Inbuf[i + 6];
                    date.Date = Inbuf[i + 7];
                    date.WeekDay = RTC_WEEKDAY_MONDAY;
                    k_SetDate(&date);
                    RTC_TimeTypeDef time;
                    time.Hours = Inbuf[i + 8];
                    time.Minutes = Inbuf[i + 9];
                    time.Seconds = Inbuf[i + 10];
                    time.TimeFormat = RTC_HOURFORMAT_24;
                    k_SetTime(&time);
                }
            }
            break;
            case PC_GETDATA:
            {
                PC_SendSettings();
                // send log file in pieces
                // send settings
                // send
            }
        }
    }
    break;
    case PC_STARTSTOP:
    {
        if(length == 0x0001)
        {
            CC_State = Inbuf[i + 5];
            CC_Update_Counter = 0;
        }
    }
    break;
    case PC_POLL:
    {
    }
}
break;
}

```

```

case PC_SETMANUAL:
{
    //if(length == 0x0003)
    //{
        CC_Manual.Temp = (Inbuf[i + 5] << 8) + Inbuf[i + 6];
        CC_Manual.Steep = Inbuf[i + 7];
        CC_Manual.Send = 255;
    //}
}

break;
case PC_SETALARM:
{
    if(length == 0x0014)
    {
        CC_Setings.WarnTemp[0] = (Inbuf[i + 5] << 8) + Inbuf[i + 6];
        CC_Setings.WarnTemp[1] = (Inbuf[i + 7] << 8) + Inbuf[i + 8];
        CC_Setings.WarnTemp[2] = (Inbuf[i + 9] << 8) + Inbuf[i + 10];
        CC_Setings.WarnTemp[3] = (Inbuf[i + 11] << 8) + Inbuf[i + 12];
        CC_Setings.AlarmTemp[0] = (Inbuf[i + 13] << 8) + Inbuf[i + 14];
        CC_Setings.AlarmTemp[1] = (Inbuf[i + 15] << 8) + Inbuf[i + 16];
        CC_Setings.AlarmTemp[2] = (Inbuf[i + 17] << 8) + Inbuf[i + 18];
        CC_Setings.AlarmTemp[3] = (Inbuf[i + 19] << 8) + Inbuf[i + 20];
        CC_Setings.AlarmRelayOn = Inbuf[i + 21];
    }
}
break;
case PC_ECHOALARM:
{
}
break;
case PC_CLEARSETPOINTS:
{
    CC_SetpointCount = 0;
}
break;
case PC_RESETRELAY:
{
    if(length == 0x0001)
    {
        CC_WriteOutput(Inbuf[i + 5], 0);
    }
}
break;

}
return Outlen;
}

```

```

/************************************************************************\

* File:          Protocol.h
* Project:       Climate Chamber Control
* Unit:          Main Interface
* Goal:          To easily make a TCP/IP ethernetconnecton
* Platform:      STM32F7-Disco
* Created:       03-11-2015
* Author:        D. Keekstra
* Company:       Brunelco
\************************************************************************/

#ifndef APPLICATION_USER_MODULES_Protocol_H_
#define APPLICATION_USER_MODULES_Protocol_H_


/************************************************************************\

* Includes
\************************************************************************/
#include "DebugWindow.h"
/************************************************************************\

* Constant declarations
\************************************************************************/


/* Incoming Functions */
#define PC_ADDSETPOINT          0
#define PC_SETDATETIME          1
#define PC_GETDATA               2
#define PC_STARTSTOP              3
#define PC_POLL                  4
#define PC_SETMANUAL             5
#define PC_SETALARM               6
#define PC_ECHOALARM              7
#define PC_CLEARSETPONTS          8
#define PC_RESETRELAY             9

/* Outgoing Functions */
#define OUTG_SENDALARM           0xF0
#define OUTG_SENDDPOINT           0xF1
#define OUTG_SENDSTAT              0xF2
#define OUTG_SENDALARMS            0xF3
#define OUTG_ECHOPOLL              0xF4
#define OUTG_SENDMEMB              0xF5
#define OUTG_SENDPROW              0x06

/* CMD for GETDATA */
#define GD_ALARMS                 0
#define GD_STATS                  1
#define GD_MEMD                   2
#define GD_PROW                    3

/* Echo key */
#define ECHOKEY                  "febb9dc570881d413ec088d48a478411"

/************************************************************************\

* Functions Declarations
\************************************************************************/
voidPC_SendTemps(Temp *temp, unsignedchar IO);
voidPC_SendAlarm(short temp, char nr);
/************************************************************************\

* Globals
\************************************************************************/


#endif/* APPLICATION_USER_MODULES_Protocol_H_ */

```

```

/*
 * File:          Storage.c
 * Project:       Climat Control
 * Unit:         Interface
 * Goal:          To safe the data to the SD storage card
 * Platform:     STM32F746
 * Created:      02-01-16
 * Author:        DanielKekstra
 * Company:      Brunelco
 */
#include "Storage.h"
#include "ClimateModule.h"
#include "main.h"
/*
 * Functions Declarations
 */
static void FATfsThread(void const *argument);
void Solve_Error(FRESULT x);
FRESULT open_append (
    FIL* fp,           /* [OUT] File object to create */
    const char* path   /* [IN]  File name to be opened */
);
/*
 * Globals
 */
osThreadId sdid = 0;
unsigned char SD_State = 0xff;
RTC_TimeTypeDef time;
RTC_DateTypeDef date;
char SDPath[4]; /* SD card logical drive path */
volatile struct SD
{
    char *FileName;
    char WriteData[SD_MAX_RW_LEN];
    char *ReadData;
}sd;
/*
 * Public Functions
 */
void Temp2Str(short temp, char *retstr)
{
    char buf[8], i;
    itoa(temp, buf, 10);
    char len = strlen(buf);
    for(i = 0; i < len - 1; i++)
    {
        retstr[i] = buf[i];
    }
    retstr[i] = '.';
    i++;
    retstr[i] = buf[i - 1];
    i++;
    retstr[i] = '\0';
}
void SD_CreateFile(char *FileNameBuf, char program)
{
    char bigbuf[240];
    char buf[60];
    char buftemp [8];
    k_GetTime(&time);
    k_GetDate(&date);
    sprintf(FileNameBuf, "Log %d-%d-%d %d.%d.%d.csv", date.Date, date.Month, date.Year, time.Hours,
    time.Minutes, time.Seconds);
    // write common controls
    if(program) sprintf(buf, "Program;; Profile; ; Date; %d-%d-%d ; ;\r", date.Date, date.Month, date.Year);
    else sprintf(buf, "Program; Manual; ; Date; %d-%d-%d ; ;\r", date.Date, date.Month, date.Year);
    strcpy(bigbuf,buf);
    // write alarm temps
    sprintf(buf, "Sensor1;");
    Temp2Str(CC_Setings.AlarmTemp[0], &buftemp);
    strcat(buf,buftemp);
    strcat(buf, ";Sensor2;");
    Temp2Str(CC_Setings.AlarmTemp[1], &buftemp);
    strcat(buf,buftemp);
    strcat(buf, ";Sensor3;");
    Temp2Str(CC_Setings.AlarmTemp[2], &buftemp);
    strcat(buf,buftemp);
    strcat(buf, ";Sensor4;");
    Temp2Str(CC_Setings.AlarmTemp[3], &buftemp);
    strcat(buf,buftemp);
    strcat(buf, "\r");
    strcat(bigbuf,buf);
}

```

```

// write index
strcat(bigbuf, "Time; Setpoint; RoomTemp; Temp1; Temp2; Temp3; Temp4; IO\r");
SD_WriteText(FileNameBuf, bigbuf);
}
voidSD_WriteTemps(Temp *temp, unsignedchar IO, char* FileName, short SetPoint)
{
    k_GetTime(&time);
    k_GetDate(&date);
    char buf[100];
    buf[0] = '\0';
    char buftemp[8];
    // set time
    sprintf(buf, "%d:%d:%d;", time.Hours, time.Minutes, time.Seconds);
    // set setpoint
    Temp2Str(SetPoint, &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set roomtemp
    Temp2Str(temp->Tempkast, &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set sens1
    Temp2Str(temp->TempSensor[0], &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set sens2
    Temp2Str(temp->TempSensor[1], &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set sens3
    Temp2Str(temp->TempSensor[2], &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set sens4
    Temp2Str(temp->TempSensor[3], &buftemp);
    strcat(buf,buftemp);
    strcat(buf,";");
    // set IO
    itoa(IO, buftemp, 10);
    strcat(buf,buftemp);
    // add CR
    strcat(buf, "\r");
    // Time; Setpoint; RoomTemp; Temp1; Temp2; Temp3; Temp4; IO\r
    // send away
    SD_WriteText(FileName,buf);
}
voidSD_Init(void)
{
    if (sdid == 0)
    {
        osThreadDef(uSDThread, FATfsThread, osPriorityRealtime, 0, 8 * configMINIMAL_STACK_SIZE);
        sdid = osThreadCreate(osThread(uSDThread), NULL);
    }
    SD_State = SD_INIT;
}
voidSD_Deinit(void)
{
    SD_State = SD_DEINIT;
}
voidSD_Format(void)
{
    unsignedchar i = 0;
    if(SD_State < SD_DEINIT)
    {
        while(SD_State != SD_READY)
        {
            if(i++ > SD_TimeOut) return;
            osDelay(1);
        }
        SD_State = SD_FORMAT;
    }
}
voidSD_WriteText(char FileName[], char Text[])
{
    unsignedchar i = 0;
    sd.FileName = FileName;
    strcpy(sd.WriteData, Text);
    if(SD_State < SD_DEINIT)
    {
        while(SD_State != SD_READY)
        {
            if(i++ > SD_TimeOut) return;
            osDelay(1);
        }
    }
}

```

```

        SD_State = SD_WRITE;
    }
}
//*****************************************************************************
/* Threads
\*****
static void FATfsThread(void const *argument)
{
    //unsigned char error;
    HRESULT res;
    FATFS SDFatFs; /* File system object for SD card logical drive */
    FIL MyFile; /* File object */

    uint32_t byteswritten, bytesread;
    while(1)
    {
        switch(SD_State)
        {
            case SD_READY: // do nothing just waiting
            break;
            case SD_FORMAT: // format SD
                if(f_mkfs((TCHAR const*)SDPath, 0, 0) != FR_OK) SD_State = SD_ERROR;
                else SD_State = SD_READY;
            break;
            case SD_WRITE:
                res = f_open(&MyFile, sd.FileName, FA_WRITE | FA_OPEN_ALWAYS);
                if(res != FR_OK)
                {
                    Solve_Error(res);
                    break;
                    SD_State = SD_ERROR;
                }
                int size = (&MyFile)->fsize;
                res = f_lseek(&MyFile, size);
                if(res != FR_OK)
                {
                    SD_State = SD_ERROR;
                }
                res = f_write(&MyFile, sd.WriteData, strlen(sd.WriteData), (void
*)&byteswritten);
                f_close(&MyFile);
                if(res != FR_OK)
                {
                    SD_State = SD_ERROR;
                }
                else SD_State = SD_READY;
            break;

            case SD_READ:
                res = f_open(&MyFile, sd.FileName, FA_READ);
                res = f_read(&MyFile, sd.ReadData, sizeof(sd.ReadData), (UINT*)&bytesread);
                f_close(&MyFile);
                if((bytesread != 0) || (res != FR_OK) || (bytesread != byteswritten)) SD_State =
SD_ERROR;
                else SD_State = SD_READY;
            break;
            case SD_DEINIT:
                FATFS_UnLinkDriver(SDPath);
                SD_State = SD_OFFLINE;
            break;
            case SD_INIT:
                if(FATFS_LinkDriver(&SD_Driver, SDPath) == FR_OK)
                {
                    if(f_mount(&SDFatFs, (TCHAR const*)SDPath, 0) == FR_OK)
                    {
                        SD_State = SD_READY;
                        //SD_Format();
                    }
                    else SD_State = SD_ERROR;
                }
                else SD_State = SD_ERROR;
            break;
            case SD_ERROR:
                if(res == FR_LOCKED)
                {
                    FATFS_UnLinkDriver(SDPath);
                    if(FATFS_LinkDriver(&SD_Driver, SDPath) == FR_OK)
                    {
                        if(f_mount(&SDFatFs, (TCHAR const*)SDPath, 0) == FR_OK)
                        {
                            SD_State = SD_READY;
                            //SD_Format();
                        }
                        else SD_State = SD_ERROR;
                    }
                }
        }
    }
}

```

```

        }
        SD_State = SD_DEINIT;
    break;
    case SD_OFFLINE:
    break;
    case 0xff:
    break;
}
osDelay(1);
}

voidSolve_Error(FRESULT x)
{
    switch(x)
    {
        caseFR_LOCKED:
            FATFS_UnLinkDriver(SDPath);
            SD_State = SD_INIT;
        break;
    }
}

FRESULTopen_append (
FIL* fp,          /* [OUT] File object to create */
constchar* path   /* [IN]  File name to be opened */
)
{
FRESULT fr;

/* Opens an existing file. If not exist, creates a new file. */
fr = f_open(fp, path, FA_WRITE | FA_OPEN_ALWAYS);
if (fr == FR_OK) {
/* Seek to end of the file to append data */
fr = f_lseek(fp, f_size(fp));
if (fr != FR_OK)
    f_close(fp);
}
return fr;
}
/************************************************************************
* File:           Storage.h
* Project:        Climat Control
* Unit:          Interface
* Goal:          To save the data to the SD storage card
* Platform:      STM32F746
* Created:       02-01-16
* Author:         DanielKeeckstra
* Company:       Brunelco
\************************************************************************/
#ifndef APPLICATION_USER_MODULES_STORAGE_H_
#define APPLICATION_USER_MODULES_STORAGE_H_
/************************************************************************
* Includes
\************************************************************************/
/* FatFs includes components */
#include "ff_gen_drv.h"
#include "sd_diskio.h"
#include "ClimateModule.h"
/************************************************************************
* Constant declarations
\************************************************************************/
#define SD_MAX_RW_LEN      255
#define SD_TimeOut         50 // mS
#define SD_READY           0
#define SD_FORMAT          1
#define SD_WRITE           2
#define SD_READ            3
// bad conditions
#define SD_DEINIT          4
#define SD_ERROR           5
#define SD_OFFLINE         6
#define SD_INIT             7
/************************************************************************
* Functions Declarations
\************************************************************************/
voidSD_Init(void);
voidSD_Deinit(void);
voidSD_Format(void);
voidSD_WriteText(char FileName[], char Text[]);
voidSD_WriteTemps(Temp *temp, unsignedchar IO, char* FileName, short SetPoint);
voidSD_CreateFile(char *FileNameBuf, char program);
\************************************************************************/
#endif/* APPLICATION_USER_MODULES_<FILENAME>_H_ */

```

Bijlage 8. Applicatie software

De applicatie software is geschreven in visual studio C#. Hierbij is met een grafische ontwikkel tool de gebruikers interface opgebouwt en met code de functionaliteit er achter. in de applicatie is gebruik gemaakt van de beschikbare windows classes.

Main applicatie code:

```

using System;
using System.IO;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Windows.Forms.DataVisualization;
using System.Windows.Forms.DataVisualization.Charting;
using System.Threading;

namespace ClimateControl
{
public partial class Main : Form
{
    #region globals
    //-- chart lines
    const string set = "Instelling";
    const string cur = "Huidig";

    const string Ain1 = "Temperatuur 1";
    const string Ain2 = "Temperatuur 2";
    const string Ain3 = "Temperatuur 3";
    const string Ain4 = "Temperatuur 4";

    const string Din1 = "Input 1";
    const string Din2 = "Input 2";
    const string Din3 = "Input 3";
    const string Din4 = "Input 4";

    const string Dout1 = "Output 1";
    const string Dout2 = "Output 2";
    const string Dout3 = "Output 3";
    const string Dout4 = "Output 4";

    string[] lines_text = { set, cur,
                           Ain1, Ain2, Ain3, Ain4,
                           Din1, Din2, Din3, Din4,
                           Dout1, Dout2, Dout3, Dout4 };
    Color[] lines_color = {Color.Black, Color.Red,
                          Color.Yellow, Color.Green, Color.Blue, Color.Orange,
                          Color.Yellow, Color.Green, Color.Blue, Color.Orange,
                          Color.Yellow, Color.Green, Color.Blue, Color.Orange };
    int[] lines_thicknes = {3,3,
                           2,2,2,2,
                           1,1,1,1,
                           1,1,1,1 };
    int[] lines_times = {0,0,
                         0,0,0,0,
                         0,0,0,0,
                         0,0,0,0 };
    bool[] lines_enabled = {true,true,
                           true,true,true,true,
                           true,true,true,true };

    // chart zoom and size
    const int Zoom_GridInterval = 12;
    const int m = 60;
    int[] Zoom_Steps = { m, 2 * m, 6 * m, 12 * m, 30 * m, 60 * m, 120 * m, 360 * m, 720 * m,
                        1440 * m };
    int[] Zoom_GridIntv = { 12, 12, 6, 12, 6, 12, 12, 12, 12 };
    int[] Zoom_GridUnit = { 1, 1, 60, 60, 60, 60, 60, 60, 3600 };

    string[] Zoom_GridName = {" Sec", " Sec", " Min", " Min", " Min", " Min", " Min", " hour", " Hour"};
    int Zoom_Zoom = 6;
    int Zoom_Size = 60 * m;
    int Zoom_Start = 0;
    double xVal, yVal;
}

```

```

uint lastsec;

// 
string startupfile;
bool browser_enabled = true;
double [] Current_Temp = newdouble[5];
double ManualTemp = 0;
double ManualSteepness = 0.1;
Point? prevPosition = null;
ToolTip tooltip = newToolTip();

// filer
privatefloat[,] avgval = newfloat[10,5];
volatilebyte cnt = 0;

// modes graphics
constbyte MODE_INIT = 0;
constbyte MODE_SETUP = 1;
constbyte MODE_NOOP = 2;
constbyte MODE_PROGRAM = 4;
constbyte MODE_DRAW = 5;
constbyte MODE_MANUALON = 6;
constbyte MODE_MANUALOFF = 7;
byte Mode = 0;
Ethernet eth = newEthernet();

struct logpoint
{
    publicfloat[] TempSensor;
    publicfloat Tempkast;
    publicbyte IO;
    publicbyte Year;
    publicbyte Month;
    publicbyte Day;
    publicbyte Hours;
    publicbyte Minuts;
    publicbyte seconds;
};

struct point
{
    publicbyte Length;
    publicshort Temp;
    publicbyte Steep;
};

struct settings
{
    publicuint LogPointCount;
    publicfloat[] WarnTemp;
    publicfloat[] AlarmTemp;
    publicbool[] AlarmOn;
    publicbool[] WarnOn;
};

// setting global
settings AlarmSet;
AvgFilter filter;

// keuzes van de vraag data op functie (CC_GetData(byte function));
constbyte CC_GETALARM = 0;
constbyte CC_GETSTATUS = 1;
constbyte CC_GETMEMDAT = 2;
constbyte CC_GETQUE = 3;

// Start stop functie modus
constbyte CC_STOP = 0;
constbyte CC_MANUAL = 1;
constbyte CC_PROGRAM = 2;
volatilebyte CC_Modus = 0;

// point count
int startcount = 0;
int count = 0;
#endregion

#region Basics
public Main(string[] arg = null)
{
    InitializeComponent();
    if (arg != null) startupfile = arg[0];
}
privatevoid Form1_Load(object sender, EventArgs e)
{
    Chart_CreateLines();
    Chart_CreateControls();
    SetSize();
}

```

```

if (startupfile != null)
{
    UpdateMode(MODE_SETUP);
    chart_loadfile(startupfile);
}
else UpdateMode(MODE_INIT);

// startup ethernet
eth.ReceivedData += newEthernet.ReceiveDataHandler(ClientReceive);
eth.Disconnected += newEthernet.DisconnectedHandler(ClientDisconnected);
eth.Connected += newEthernet.ConnectedHandler(ClientConnected);

ClientStart();
Current_Temp[0] = -50;

}

private bool b2bool(byte x, byte bitn)
{
return (x & (1 << bitn)) != 0;
}

private void setGlobals()
{
    AlarmSet.AlarmOn = newbool[4];
    AlarmSet.WarnOn = newbool[4];
    AlarmSet.AlarmTemp = newfloat[4];
    AlarmSet.WarnTemp = newfloat[4];
}

#endregion

#region Chart Graphics
private void Chart_ClearLines()
{
    chart.Series[1].Points.Clear();
    chart.Series[2].Points.Clear();
    chart.Series[3].Points.Clear();
    chart.Series[4].Points.Clear();
    chart.Series[5].Points.Clear();
    chart.Series[6].Points.Clear();
    chart.Series[7].Points.Clear();
    chart.Series[8].Points.Clear();
    chart.Series[9].Points.Clear();
    chart.Series[10].Points.Clear();
    chart.Series[11].Points.Clear();
    chart.Series[12].Points.Clear();
    chart.Series[13].Points.Clear();
}

private void Chart_CreateLines()
{
// chart setup
    chart.Series.Clear();
    chart.ChartAreas[0].BackColor = Color.DarkGray;
    chart.ChartAreas[0].CursorX.AutoScroll = false;//<-----
    chart.ChartAreas[0].AxisX.IsMarginVisible = false;
    chart.ChartAreas[0].AlignmentStyle = AreaAlignmentStyles.None;

// y axis
    chart.ChartAreas[0].AxisY.MajorGrid.LineColor = Color.LightGray;
    chart.ChartAreas[0].AxisY.Title = "Temperature °C";
    chart.ChartAreas[0].AxisY.TitleFont = newFont(FontFamily.GenericSansSerif, 11, FontStyle.Regular);
    chart.ChartAreas[0].AxisY.Minimum = -40;
    chart.ChartAreas[0].AxisY.Maximum = 130;
    chart.ChartAreas[0].AxisY.Interval = 10;
    chart.ChartAreas[0].AxisY.MajorGrid.LineDashStyle = ChartDashStyle.Dot;
    chart.ChartAreas[0].AxisY.ScaleView.Zoomable = false;
    chart.ChartAreas[0].AxisY.ScrollBar.Enabled = false;

// X axis
    chart.ChartAreas[0].AxisX.MajorGrid.LineColor = Color.LightGray;
    chart.ChartAreas[0].AxisX.Title = "Time";
    chart.ChartAreas[0].AxisX.TitleFont = newFont(FontFamily.GenericSansSerif, 11, FontStyle.Regular);
    chart.ChartAreas[0].AxisX.Minimum = 0;
    chart.ChartAreas[0].AxisX.Maximum = 86400;
    chart.ChartAreas[0].AxisX.MajorGrid.LineDashStyle = ChartDashStyle.Dot;
    chart.ChartAreas[0].AxisX.ScaleView.Zoomable = false;
    chart.ChartAreas[0].AxisX.ScaleView.Zoom(Zoom_Start, Zoom_Start + Zoom_Size);
    chart.ChartAreas[0].AxisX.ScrollBar.Enabled = false;
    chart.ChartAreas[0].AxisX.IntervalAutoMode = IntervalAutoMode.FixedCount;

// add lines to chart
for (int i = 0; i < lines_text.Length; i++)
{
    chart.Series.Add(lines_text[i]);
    chart.Series[i].Color = lines_color[i];
    chart.Series[i].ChartType = SeriesChartType.FastLine;
}

```

```

        chart.Series[i].BorderWidth = lines_thinknes[i];
        chart.Series[i].Enabled = lines_enabled[i];

    }

this.MouseWheel += newMouseEventHandler(MouseWheelEvent);
Chart_SetAxisLabels();

StripLine stripline = newStripLine();
stripline.Interval = 0;
stripline.IntervalOffset = 0;
stripline.StripWidth = 1;
stripline.BackColor = Color.Black;
chart.ChartAreas[0].AxisX.StripLines.Add(stripline);
}

private void Chart_UpdateLines()
{
for (int i = 0; i < lines_text.Length; i++)
{
    chart.Series[i].Enabled = lines_enabled[i];
    chart.Series[i].Color = lines_color[i];
    chart.Series[i].BorderWidth = lines_thinknes[i];
}
}

private void Chart_CreateControls()
{
// add controls
const int HeightStep = 25;
const int WidthOffset = 3;
const int HeightOffset = 3;
int height = HeightOffset;

// add combobox
ComboBox CB = newComboBox();
CB.Name = "CB_ChartSet";
CB.Location = newPoint(WidthOffset + 3, height + 4);
CB.DropDownStyle = ComboBoxStyle.DropDownList;
CB.FlatStyle = FlatStyle.Flat;
CB.Items.Add("All");
CB.Items.Add("Setup");
CB.Items.Add("Control only");
CB.Items.Add("Set 1");
CB.Items.Add("Set 2");
CB.Items.Add("Set 3");
CB.Items.Add("Set 4");
//CB.SelectedItem = CB.Items[0];
CB.SelectedValueChanged += newEventHandler(Chart_OnEventCombo); // add event
ChartViewPanel.Controls.Add(CB);
height = height + HeightStep;

// rows
for (int i = 0; i < lines_text.Length; i++)
{
// numeric up down
NumericUpDown NumUD = newNumericUpDown();
NumUD.Name = "NUD_ChartSet" + i.ToString();
NumUD.Location = newPoint(WidthOffset + 3, height + 4);
NumUD.Size = newSize(40, 23);
NumUD.Value = lines_thinknes[i];
NumUD.ValueChanged += newEventHandler(Chart_OnEventThicknes);
ChartViewPanel.Controls.Add(NumUD);

// Checkbox
CheckBox checkbox = newCheckBox();
checkbox.Name = "CB_ChartSet" + i.ToString();
checkbox.Text = lines_text[i];
checkbox.FlatStyle = FlatStyle.Flat;
checkbox.Checked = lines_enabled[i];
checkbox.Location = newPoint(WidthOffset + 76, height + 3);
checkbox.Click += newEventHandler(Chart_OnEventCheckbox); // add event
ChartViewPanel.Controls.Add(checkbox);

// Button
Button button = newButton();
button.Name = "BTN_ChartSet" + i.ToString();
button.Text = "";
button.BackColor = lines_color[i];
button.Size = newSize(23, 23);
button.Location = newPoint(WidthOffset + 50, height + 3);
button.FlatStyle = FlatStyle.Flat;
button.Click += newEventHandler(Chart_OnEventColorPicker); // add event
ChartViewPanel.Controls.Add(button);

// new line
}
}

```

```

        height = height + HeightStep;
    }
    ChartViewPanel.Size = newSize(175, height + 3);

}

privatevoid Chart_OnEventCheckbox(object sender, EventArgs e)
{
    CheckBox checkbox = (CheckBox)sender;
    if (checkbox != null) // kan inprincipe niet gebeuren maar toch
    {
        int i = Convert.ToInt16(checkbox.Name.Replace("CB_ChartSet", ""));
        lines_enabled[i] = checkbox.Checked;
        Chart_UpdateLines();
    }
}
privatevoid Chart_OnEventColorPicker(object sender, EventArgs e)
{
    Button button = (Button)sender;
    if (button != null) // kan inprincipe niet gebeuren maar toch
    {
        ColorDialog dlg = newColorDialog();
        if (dlg.ShowDialog() == DialogResult.OK)
        {
            button.BackColor = dlg.Color;
        }
        int i = Convert.ToInt16(button.Name.Replace("BTN_ChartSet", ""));
        lines_color[i] = dlg.Color;
        Chart_UpdateLines();
    }
}
privatevoid Chart_OnEventThicknes(object sender, EventArgs e)
{
    NumericUpDown NumUD = (NumericUpDown)sender;
    if (NumUD != null) // kan inprincipe niet gebeuren maar toch
    {
        int i = Convert.ToInt16(NumUD.Name.Replace("NUD_ChartSet", ""));
        lines_thicknes[i] = Convert.ToInt16(NumUD.Value);
        Chart_UpdateLines();
    }
}
privatevoid LineView(string selected)
{
    switch (selected)
    {
        case "All":
            lines_enabled[0] = true;
            lines_enabled[1] = true;
            lines_enabled[2] = true;
            lines_enabled[3] = true;
            lines_enabled[4] = true;
            lines_enabled[5] = true;
            lines_enabled[6] = true;
            lines_enabled[7] = true;
            lines_enabled[8] = true;
            lines_enabled[9] = true;
            lines_enabled[10] = true;
            lines_enabled[11] = true;
            lines_enabled[12] = true;
            lines_enabled[13] = true;
            break;
        case "Setup":
            lines_enabled[0] = true;
            lines_enabled[1] = false;
            lines_enabled[2] = false;
            lines_enabled[3] = false;
            lines_enabled[4] = false;
            lines_enabled[5] = false;
            lines_enabled[6] = false;
            lines_enabled[7] = false;
            lines_enabled[8] = false;
            lines_enabled[9] = false;
            lines_enabled[10] = false;
            lines_enabled[11] = false;
            lines_enabled[12] = false;
            lines_enabled[13] = false;
            break;
        case "Control only":
            lines_enabled[0] = true;
            lines_enabled[1] = true;
            lines_enabled[2] = false;
            lines_enabled[3] = false;
            lines_enabled[4] = false;
            lines_enabled[5] = false;
            lines_enabled[6] = false;
    }
}

```

```

        lines_enabled[7] = false;
        lines_enabled[8] = false;
        lines_enabled[9] = false;
        lines_enabled[10] = false;
        lines_enabled[11] = false;
        lines_enabled[12] = false;
        lines_enabled[13] = false;

    break;
    case "Set 1":
        lines_enabled[0] = false;
        lines_enabled[1] = false;
        lines_enabled[2] = true;
        lines_enabled[3] = false;
        lines_enabled[4] = false;
        lines_enabled[5] = false;
        lines_enabled[6] = true;
        lines_enabled[7] = false;
        lines_enabled[8] = false;
        lines_enabled[9] = false;
        lines_enabled[10] = true;
        lines_enabled[11] = false;
        lines_enabled[12] = false;
        lines_enabled[13] = false;

    break;
    case "Set 2":
        lines_enabled[0] = false;
        lines_enabled[1] = false;
        lines_enabled[2] = false;
        lines_enabled[3] = true;
        lines_enabled[4] = false;
        lines_enabled[5] = false;
        lines_enabled[6] = false;
        lines_enabled[7] = true;
        lines_enabled[8] = false;
        lines_enabled[9] = false;
        lines_enabled[10] = false;
        lines_enabled[11] = true;
        lines_enabled[12] = false;
        lines_enabled[13] = false;

    break;
    case "Set 3":
        lines_enabled[0] = false;
        lines_enabled[1] = false;
        lines_enabled[2] = false;
        lines_enabled[3] = false;
        lines_enabled[4] = true;
        lines_enabled[5] = false;
        lines_enabled[6] = false;
        lines_enabled[7] = false;
        lines_enabled[8] = true;
        lines_enabled[9] = false;
        lines_enabled[10] = false;
        lines_enabled[11] = false;
        lines_enabled[12] = true;
        lines_enabled[13] = false;

    break;
    case "Set 4":
        lines_enabled[0] = false;
        lines_enabled[1] = false;
        lines_enabled[2] = false;
        lines_enabled[3] = false;
        lines_enabled[4] = false;
        lines_enabled[5] = true;
        lines_enabled[6] = false;
        lines_enabled[7] = false;
        lines_enabled[8] = false;
        lines_enabled[9] = true;
        lines_enabled[10] = false;
        lines_enabled[11] = false;
        lines_enabled[12] = false;
        lines_enabled[13] = true;

    break;
}
    ChartViewPanel.Controls.Clear();
    Chart_CreateControls();
    Chart_UpdateLines();
}

private void Chart_OnEventCombo(object sender, EventArgs e)
{
    ComboBox CB = (ComboBox)sender;
    string selected = CB.SelectedItem.ToString();
    LineView(selected);
}

private void Chart_ReGrid()
{
}

```

```

//update chart
    chart.ChartAreas[0].AxisX.ScaleView.Zoom(Zoom_Start, Zoom_Start + Zoom_Size);
    chart.ChartAreas[0].AxisX.Interval = Zoom_Size / Zoom_GridInterval;
    chart.ChartAreas[0].AxisX.Minimum = 0;
    chart.ChartAreas[0].AxisX.Maximum = 86400;
    chart.ChartAreas[0].AxisX.MinorGrid.Interval = Zoom_Size / Zoom_GridInterval;
    chart.ChartAreas[0].AxisX.MajorGrid.Interval = Zoom_Size / Zoom_GridInterval;
}
//setup mode
privatevoid Chart_Setup()
{
    UpdateMode(MODE_SETUP);

    Chart_UpdateLines();
    chart.Series[0].Points.Clear();
    if (Current_Temp[0] < -40 || Current_Temp[0] > 130) chart.Series[0].Points.AddXY(0, 20);
    else chart.Series[0].Points.AddXY(0, Current_Temp[0]);
}
privatevoid Chart_SetAxisLabels()
{
// set x axis labeling
    chart.ChartAreas[0].AxisX.LabelStyle.Format = "{0}" + Zoom_GridName[Zoom_Zoom];
    chart.ChartAreas[0].AxisX.CustomLabels.Clear();

    int stepsize = Zoom_Steps[Zoom_Zoom] / Zoom_GridIntv[Zoom_Zoom]; // stapgrote = stapgrote hele veld / grid op het veld
    int stepvalue = stepsize / Zoom_GridUnit[Zoom_Zoom]; // stap waarde = stapgrote / unitgroote
    int offset = Zoom_Start / Zoom_GridUnit[Zoom_Zoom];
    int position = Zoom_Start - stepsize / 2;
    for (int i = 0; i < Zoom_GridIntv[Zoom_Zoom] + 1; i++)
    {
        CustomLabel Xlabel = newCustomLabel();
        Xlabel.FromPosition = position;
        position = position + stepsize;
        Xlabel.ToPosition = position;
        int labelvalue = (i) * stepvalue + offset;
        Xlabel.Text = labelvalue.ToString() + Zoom_GridName[Zoom_Zoom];
        Xlabel.RowIndex = 0;
        Xlabel.LabelMark = LabelMarkStyle.SideMark;
        chart.ChartAreas[0].AxisX.CustomLabels.Add(Xlabel);
    }
    chart.ChartAreas[0].AxisX.Interval = Zoom_Size / Zoom_GridInterval;
}
privatevoid Chart_UpdateCursor()
{
    LBL_Yas.Text = "Y: " + yVal.ToString("##.0", System.Globalization.CultureInfo.InvariantCulture.InvariantCulture) + " °C";
    LBL_Xas.Text = "X: " + Convert.ToInt32(xVal / Zoom_GridUnit[Zoom_Zoom]).ToString() +
    Zoom_GridName[Zoom_Zoom];

    if (Mode == MODE_DRAW)
    {
// delta x and y
        chart.Series[0].Points[chart.Series[0].Points.Count - 1].XValue = Convert.ToInt32(xVal);
        double Xlast = chart.Series[0].Points[chart.Series[0].Points.Count - 2].XValue;
        double Ylast = chart.Series[0].Points[chart.Series[0].Points.Count - 2].YValues[0];
        double dX = xVal - Xlast;
        double dY = yVal - Ylast;
        double change;
        if (dX / 60 == 0) change = 0;
        else change = dY / (dX / 60);

        LBL_dX.Text = "dX: " + Convert.ToInt32(dX / Zoom_GridUnit[Zoom_Zoom]).ToString() +
        Zoom_GridName[Zoom_Zoom];
        if (Control.ModifierKeys == Keys.Shift)
        {
            LBL_dY.Text = "dY: 0 °C";
            change = 0;
            LBL_Yas.Text = "Y: " + Ylast.ToString("#0.0", System.Globalization.CultureInfo.InvariantCulture.InvariantCulture) +
            " °C";
            chart.Series[0].Points[chart.Series[0].Points.Count - 1].YValues[0] = Ylast;
        }
        else
        {
            LBL_dY.Text = "dY: " + dY.ToString("#0.0", System.Globalization.CultureInfo.InvariantCulture.InvariantCulture) +
            " °C";
            chart.Series[0].Points[chart.Series[0].Points.Count - 1].YValues[0] = yVal;
        }
        LBL_Line.Text = "Line: " + change.ToString("#0.0", System.Globalization.CultureInfo.InvariantCulture.InvariantCulture) +
        " °C/Min";
// line color
        if (dX < 0 || change > 2 || change < -2) // check if temperture is not higher then 2degree a minute
        {

            if (dX < 0) LBL_dX.ForeColor = Color.Red;
        }
    }
}

```

```

else LBL_dX.ForeColor = Color.Black;
if (change > 2 || change < -2) LBL_Line.ForeColor = Color.Red;
else LBL_Line.ForeColor = Color.Black;
    chart.Series[0].Color = Color.Red;
}
else
{
    LBL_Line.ForeColor = Color.Black;
    LBL_dX.ForeColor = Color.Black;
    chart.Series[0].Color = Color.Black;
}
if (dX > 15300)
{
    chart.Series[0].Color = Color.Red;
}
}
private void Chart_UpdatePoints()
{
    LB_Points.Items.Clear();
int i = 0;
foreach (DataPoint point in chart.Series[0].Points)
{
    LB_Points.Items.Add(i.ToString() + " Time: " + (point.XValue / 60).ToString("##.0",
System.Globalization.CultureInfo.InvariantCulture) + " Min " + "\t| Temp: " + point.YValues[0].ToString("##.0",
System.Globalization.CultureInfo.InvariantCulture) + " °C");
    i++;
}
}
// Mouse events
private void chart_MouseMove(object sender, MouseEventArgs e)
{
// dY dT displayen
var pos = e.Location;
if (prevPosition.HasValue && pos == prevPosition.Value)
return;
    tooltip.RemoveAll();
    prevPosition = pos;
var results = chart.HitTest(pos.X, pos.Y, false,
ChartElementType.PlottingArea);
if (results[0].ChartElementType == ChartElementType.PlottingArea)
{
// x and y val
xVal = Convert.ToInt32(results[0].ChartArea.AxisX.PixelPositionToValue(pos.X) / 60) * 60;

yVal = results[0].ChartArea.AxisY.PixelPositionToValue(pos.Y);
if (yVal < -40) yVal = -40;
if (yVal > 130) yVal = 130;
    Chart_UpdateCursor();
}
}
private void MouseWheelEvent(object sender, MouseEventArgs e)
{
int change = e.Delta / 1;
int buf = Convert.ToInt32(chart.ChartAreas[0].AxisX.ScaleView.Position) / Zoom_Size;
int Steps = Zoom_Steps[Zoom_Zoom] / 2;
int CurrSteps = Zoom_Start / Steps;
if (Control.ModifierKeys == Keys.Control)
{
if (change > 0 && Zoom_Zoom > 0) Zoom_Zoom = Zoom_Zoom - 1;
elseif (change < 0 && Zoom_Zoom < 9) Zoom_Zoom = Zoom_Zoom + 1;

if (Zoom_Zoom == 9) Zoom_Start = 0;
else Zoom_Start = buf * Zoom_Steps[Zoom_Zoom];
    Zoom_Size = Zoom_Steps[Zoom_Zoom];
}
else
{
if (change > 0 && Zoom_Start < (86400 - Zoom_Steps[Zoom_Zoom])) CurrSteps++;
elseif (change < 0 && Zoom_Start > 0) CurrSteps--;
}
Zoom_Start = CurrSteps * Steps;
Chart_ReGrid();
Chart_UpdateCursor();
Chart_SetAxisLabels();
Chart_UpdateLines();
}
private void chart_Click(object sender, EventArgs e)
{
if (Mode == MODE_DRAW)
{
MouseEventArgs me = (MouseEventArgs)e;
double Xlast = chart.Series[0].Points[chart.Series[0].Points.Count - 1].XValue;
double Ylast = chart.Series[0].Points[chart.Series[0].Points.Count - 1].YValues[0];
}
}

```

```

if (me.Button == System.Windows.Forms.MouseButtons.Right)
{
    chart.Series[0].Points.RemoveAt(chart.Series[0].Points.Count - 1);
}
if (me.Button == System.Windows.Forms.MouseButtons.Left && chart.Series[0].Color == Color.Red)
{
    MessageBox.Show("Error! Line out of range.");
    chart.Series[0].Points.RemoveAt(chart.Series[0].Points.Count - 1);
}
Chart_UpdatePoints();
chart.Series[0].Color = Color.Black;
UpdateMode(MODE_SETUP);
LBL_dX.Text = "";
LBL_dY.Text = "";
LBL_Line.Text = "";
}
}
private void chart_DoubleClick(object sender, EventArgs e)
{
if (Mode == MODE_SETUP)
{
    MouseEventArgs me = (MouseEventArgs)e;
    if (me.Button == System.Windows.Forms.MouseButtons.Right)
    {
        if (chart.Series[0].Points.Count - 1 == 0) MessageBox.Show("Can't delete first point");
        else
        {
            chart.Series[0].Points.RemoveAt(chart.Series[0].Points.Count - 1);
            Chart_UpdateCursor();
            Chart_UpdatePoints();
        }
    }
    if (me.Button == System.Windows.Forms.MouseButtons.Left)
    {
        int Xlast = Convert.ToInt32(chart.Series[0].Points[chart.Series[0].Points.Count - 1].XValue);
        int Ylast = Convert.ToInt32(chart.Series[0].Points[chart.Series[0].Points.Count - 1].YValues[0]);
        chart.Series[0].Points.AddXY(Convert.ToInt32(xVal), yVal);
        UpdateMode(MODE_DRAW);
        Chart_UpdateCursor();
        Chart_UpdatePoints();
    }
}
}
private void chart_MouseLeave(object sender, EventArgs e)
{
}
private void chart_MouseEnter(object sender, EventArgs e)
{
    LeaveItems();
}
private void chart_MouseDown(object sender, MouseEventArgs e)
{
//MessageBox.Show("in");
}
private void chart_MouseUp(object sender, MouseEventArgs e)
{
//MessageBox.Show("los");
}
private void BTN_View_MouseEnter(object sender, EventArgs e)
{
    ChartViewPanel.Visible = true;
}
private void chart_loadfile(string path)
{
// load data
var sw = new StreamReader(path);
string Data = sw.ReadToEnd();
string[] rows = Data.Split('\r');
// seperate data:
string[,] data = new string[rows.Length, 3];
int i = 0, j = 0;
foreach (string row in rows)
{
    string[] cells = row.Split(';');
    i = 0;
    foreach (string cell in cells)
    {
        data[j, i] = cell;
        i++;
    }
    j++;
}
// set data
}

```

```

chart.Series[0].Points.Clear();
if (Current_Temp[0] < -40 || Current_Temp[0] > 130) chart.Series[0].Points.AddXY(0, 20);
else chart.Series[0].Points.AddXY(0, Current_Temp[0]);
// skip header start @ 1
for (int x = 2; x < j; x++)
{
try
{
if (data[x, 1] != "" && data[x, 2] != "")
{
double ydata = Convert.ToDouble(data[x, 2].Replace('.', ','));
double xdata = Convert.ToDouble(data[x, 1].Replace('.', ',')) * 60;
chart.Series[0].Points.AddXY(xdata, ydata);
}
}
catch { }
}
Chart_UpdatePoints();
}
// user events
private void BTN_AddPoint_Click(object sender, EventArgs e)
{
double Xlast = chart.Series[0].Points[chart.Series[0].Points.Count - 1].XValue;
double Ylast = chart.Series[0].Points[chart.Series[0].Points.Count - 1].YValues[0];

double Y = Convert.ToDouble(NUD_Temp.Value);
double dX = Convert.ToDouble(NUD_Time.Value * 60);
double X = Xlast + dX;
double dY = Y - Ylast;
double change;
if (dX / 60 == 0) change = 0;
else change = dY / (dX / 60);
if (change > -2 && change < 2)
{
    chart.Series[0].Points.AddXY(X, Y);
    NUD_Line.Value = 0;
    NUD_Temp.Value = 0;
    NUD_Time.Value = 0;
    Chart_UpdatePoints();
}
else
{
    MessageBox.Show("The line is too steep! (Max 2.0°C/min)");
}
}
private void BTN_DelLast_Click(object sender, EventArgs e)
{
if (chart.Series[0].Points.Count - 1 == 0) MessageBox.Show("Can't delete first point");
else
{
    chart.Series[0].Points.RemoveAt(chart.Series[0].Points.Count - 1);
    Chart_UpdatePoints();
}
}
private void BTN_Save2File_Click(object sender, EventArgs e)
{
SaveFileDialog saveFileDialog1 = new SaveFileDialog();
saveFileDialog1.Filter = "Climat Control Files (*.FCC)|*.FCC";
saveFileDialog1.FilterIndex = 1;
saveFileDialog1.RestoreDirectory = true ;
if (saveFileDialog1.ShowDialog() == DialogResult.OK)
{
string Data = "Index; Min; Temperature\r";
int i = 0;
foreach (DataPoint point in chart.Series[0].Points)
{
    Data += i.ToString() + ";" + (point.XValue / 60).ToString("#0.0",
System.Globalization.CultureInfo.InvariantCulture) + ";" + point.YValues[0].ToString("#0.0",
System.Globalization.CultureInfo.InvariantCulture) + "\r";
    i++;
}
if (!File.Exists(saveFileDialog1.FileName))
{
var file = File.Create(saveFileDialog1.FileName);
file.Close();
}
var sw = new StreamWriter(saveFileDialog1.FileName);
sw.WriteLine(Data);
sw.Close();
}
}
private void BTN_LoadFile_Click(object sender, EventArgs e)

```

```

    {
    OpenFileDialog openFileDialog1 = new OpenFileDialog();
    openFileDialog1.Filter = "Climat Control Files (*.FCC)|*.FCC";
    openFileDialog1.FilterIndex = 1;
    openFileDialog1.RestoreDirectory = true;
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        chart_loadfile(openFileDialog1.FileName);
    }
}
/*private void NUD_Time_ValueChanged(object sender, EventArgs e)
{
    if (NUD_Temp.Value == 0 && NUD_Line.Value != 0)
    {
        NUD_Temp.Value = NUD_Line.Value * NUD_Time.Value;
    }
    else if (NUD_Temp.Value != 0 && NUD_Line.Value == 0)
    {
        NUD_Line.Value = NUD_Temp.Value / NUD_Time.Value;
    }
    else if (NUD_Temp.Value != 0 && NUD_Line.Value != 0)
    {
        NUD_Line.Value = NUD_Temp.Value / NUD_Time.Value;
    }
}
private void NUD_Temp_ValueChanged(object sender, EventArgs e)
{
    if (NUD_Time.Value == 0 && NUD_Line.Value != 0)
    {
        NUD_Time.Value = NUD_Temp.Value / NUD_Line.Value;
    }
    else if (NUD_Time.Value != 0 && NUD_Line.Value == 0)
    {
        NUD_Line.Value = NUD_Temp.Value / NUD_Time.Value;
    }
    else if (NUD_Time.Value != 0 && NUD_Line.Value != 0)
    {
        NUD_Line.Value = NUD_Temp.Value / NUD_Time.Value;
    }
}
private void NUD_Line_ValueChanged(object sender, EventArgs e)
{
    if (NUD_Time.Value == 0 && NUD_Temp.Value != 0)
    {
        NUD_Time.Value = NUD_Temp.Value / NUD_Line.Value;
    }
    else if (NUD_Time.Value != 0 && NUD_Temp.Value == 0)
    {
        NUD_Time.Value = NUD_Temp.Value / NUD_Line.Value;
    }
    else if (NUD_Time.Value != 0 && NUD_Temp.Value != 0)
    {
        NUD_Time.Value = NUD_Temp.Value / NUD_Line.Value;
    }
}
*/
#endregion

#region User Events
private void Form1_SizeChanged(object sender, EventArgs e)
{
    SetSize();
}
private void LeaveItems()
{
    ChartViewPanel.Visible = false;
}
private void BTN_Browser_Click(object sender, EventArgs e)
{
    if (browser_enabled)
    {
        browser_enabled = false;
        BTN_Browser.Text = "^^";
    }
    else
    {
        browser_enabled = true;
        BTN_Browser.Text = "^^";
    }
    SetSize();
}
private void button1_Click(object sender, EventArgs e)
{
    Chart_Setup();
}
private void VBTN_A2_Click(object sender, EventArgs e)

```

```

        {
            VBTN_A2.BackColor = Color.Lime;
        }
    private void VBTN_A1_Click(object sender, EventArgs e)
    {
        VBTN_A1.BackColor = Color.Lime;
    }
    private void VBTN_A3_Click(object sender, EventArgs e)
    {
        VBTN_A3.BackColor = Color.Lime;
    }
    private void VBTN_A4_Click(object sender, EventArgs e)
    {
        VBTN_A4.BackColor = Color.Lime;
    }
    private void BTN_Connect_Click(object sender, EventArgs e)
    {
        ClientStart();
    }
    private void VBTN_OUT1_Click(object sender, EventArgs e)
    {
        CC_ResetRelay(0x00);
    }
    private void VBTN_OUT2_Click(object sender, EventArgs e)
    {
        CC_ResetRelay(0x01);
    }
    private void VBTN_OUT3_Click(object sender, EventArgs e)
    {
        CC_ResetRelay(0x02);
    }
    private void VBTN_OUT4_Click(object sender, EventArgs e)
    {
        CC_ResetRelay(0x03);
    }
    #endregion

    #region From Graphics
    private void UpdateMode(byte mode)
    {
        Mode = mode;
        switch (Mode)
        {
            case MODE_INIT:
                Setup_Panel.Visible = false;
                BTN_Setup.Enabled = true;
                BTN_Manual.Enabled = true;
                BTN_Start.Enabled = false;
                BTN_Stop.Enabled = false;
                Manual_Panel.Visible = false;
                TT_graph.Active = false;
                Alarm_Panel.Visible = false;
                Alarm_Panel.Enabled = true;
                Current_Panel.Visible = false;
                LineView("ALL");
                SetSize();
                break;
            case MODE_DRAW:
                Setup_Panel.Visible = true;
                BTN_Setup.Enabled = true;
                BTN_Manual.Enabled = true;
                BTN_Start.Enabled = false;
                BTN_Stop.Enabled = false;
                Manual_Panel.Visible = false;
                TT_graph.Active = true;
                Alarm_Panel.Visible = true;
                Alarm_Panel.Enabled = true;
                Current_Panel.Visible = false;
                SetSize();
                break;
            case MODE_SETUP:
                Setup_Panel.Visible = true;
                BTN_Manual.Enabled = true;
                BTN_Setup.Enabled = true;
                BTN_Start.Enabled = true;
                BTN_Stop.Enabled = false;
                Manual_Panel.Visible = false;
                TT_graph.Active = true;
                Alarm_Panel.Visible = true;
                Alarm_Panel.Enabled = true;
                Current_Panel.Visible = false;
                LineView("Setup");
                SetSize();
                break;
            case MODE_MANUALON:

```

```

        BTN_Start.Enabled = false;
        BTN_Stop.Enabled = true;
        BTN_Setup.Enabled = false;
        BTN_Manual.Enabled = false;
        Setup_Panel.Visible = false;
        Manual_Panel.Visible = true;
        TT_graph.Active = false;
        Alarm_Panel.Visible = true;
        Alarm_Panel.Enabled = false;
        Current_Panel.Visible = true;
        LineView("All");
        SetSize();

break;
case MODE_MANUALOFF:
        BTN_Start.Enabled = true;
        BTN_Stop.Enabled = false;
        BTN_Setup.Enabled = true;
        BTN_Manual.Enabled = true;
        Setup_Panel.Visible = false;
        Manual_Panel.Visible = true;
        TT_graph.Active = false;
        Alarm_Panel.Visible = true;
        Alarm_Panel.Enabled = true;
        Current_Panel.Visible = false;
        LineView("All");
        SetSize();

break;
case MODE_PROGRAM:
        BTN_Start.Enabled = false;
        BTN_Setup.Enabled = false;
        BTN_Manual.Enabled = false;
        BTN_Stop.Enabled = true;
        Setup_Panel.Visible = false;
        Manual_Panel.Visible = false;
        TT_graph.Active = false;
        Alarm_Panel.Visible = false;
        Alarm_Panel.Enabled = true;
        Current_Panel.Visible = true;
        LineView("All");
        SetSize();

break;
}

private void SetSize()
{
try
{
    int h = this.Height - 40;
    int w = this.Width - 16;
    int ch = h;
    int cw = w;

    chart.Location = newPoint(0, 0);
    Browser.Location = newPoint(0, h - 75);
    Browser.Size = newSize(w, 75);

    if (browser_enabled)
    {
        ch -= 75;
        Browser.Visible = true;
        BTN_Browser.Location = newPoint(w / 2 - 30, h - 95);
    }
    else
    {
        Browser.Visible = false;
        BTN_Browser.Location = newPoint(w / 2 - 30, h - 20);
    }

    if (Mode == MODE_MANUALON || Mode == MODE_PROGRAM)
    {
        cw -= 100;
    }

    if (Setup_Panel.Visible == true || Manual_Panel.Visible == true) cw -= 200;
    Setup_Panel.Location = newPoint(w - 200, 0);
    Setup_Panel.Size = newSize(200, ch - 180);
    Manual_Panel.Location = newPoint(w - 200, 0);
    Manual_Panel.Size = newSize(200, ch - 180);
    Alarm_Panel.Size = newSize(200, 180);
    Alarm_Panel.Location = newPoint(w - 200, ch - 180);

    Current_Panel.Location = newPoint(cw, 0);
    Current_Panel.Size = newSize(100, ch);
}
}

```

```

P_Cursor.Location = newPoint(cw - 100 , ch - P_Cursor.Size.Height);
chart.Size = newSize(cw , ch);
ChartViewPanel.Location = newPoint(cw - ChartViewPanel.Width, 0);
BTN_View.Location = newPoint(cw - BTN_View.Width, 0);

// in panel
    LB_Points.Size = newSize(188, Setup_Panel.Size.Height - 180);
    LB_Points.Location = newPoint(5, 180);
}
catch { }
// Browser

}

privatevoid ShowAlarm(string text)
{
if (IsFormOpen(typeof (Alarm)))
{
Alarm alarm = newAlarm();
alarm.Show();
alarm.setForm1(this);
}
}

publicbool IsFormOpen(Type formType)
{
foreach (Form form inApplication.OpenForms)
if (form.GetType().Name == form.Name)
returntrue;
returnfalse;
}

// closing
privatevoid Main_FormClosing(object sender, FormClosingEventArgs e)
{
ICON.Visible = false;
if (e.CloseReason == CloseReason.UserClosing && (Mode == MODE_MANUALON || Mode == MODE_PROGRAM))
{
ICON.Visible = true;
this.Hide();
e.Cancel = true;
}
else eth.Clientstop();
}

privatevoid ICON_Click(object sender, EventArgs e)
{
this.Show();
ICON.Visible = false;
}

privatevoid Main_FormClosed(object sender, FormClosedEventArgs e)
{
ICON.Visible = false;
}

#endregion

#region Run
privatevoid RunManual()
{
chart.Series[0].Points.Clear();
ManualTemp = Convert.ToDouble(Man_Temp.Value);
ManualSteepness = Convert.ToDouble(Man_Steep.Value);
int time = 0;
double DeltaTemp = Current_Temp[0] - ManualTemp;
if (DeltaTemp != 0) time = Convert.ToInt32((DeltaTemp) / ManualSteepness);
chart.Series[0].Points.AddXY(0, Current_Temp[0]);
chart.Series[0].Points.AddXY(time, ManualTemp);
chart.Series[0].Points.AddXY(60*60*24, ManualTemp);
Chart_ClearLines();
count = 0;
startcount = 0;
settings x = GetSettings();
CC_SetAlarms(x);
Thread.Sleep(300);
CC_SetManual(Convert.ToInt16(ManualTemp * 10), Convert.ToByte(ManualSteepness * 10));
Thread.Sleep(300);
CC_StartStop(CC_MANUAL);
}
privatevoid RunProgram()
{
byte dX;
double dY;
pb.Visible = true;
pb.Maximum = chart.Series[0].Points.Count + 2;
pb.Minimum = 0;
pb.Value = 0;
this.Validate();
int i = 0;
count = 0;
}

```

```

        startcount = 0;
        Chart_ClearLines();
        CC_ClearSetpoints();
    for (i = 1; i < chart.Series[0].Points.Count; i++)
    {
        pb.Value = i;
    this.Validate();
    point setpoint = newpoint();
    dX = (byte)((chart.Series[0].Points[i].XValue - chart.Series[0].Points[i - 1].XValue) / 60);
    dY = chart.Series[0].Points[i].YValues[0] - chart.Series[0].Points[i - 1].YValues[0];
    setpoint.Length = dX;

    setpoint.Temp = (short)(chart.Series[0].Points[i].YValues[0] * 10);
double steep = (((double)dY / (dX)) * 10);
Math.Abs(steep);
    setpoint.Steep = (byte)steep;
    CC_AddSetPoint(setpoint);
Thread.Sleep(350);
    lastsec = (uint)dX * 60;
}
    CC_SetAlarms(GetSettings());
    pb.Value = i++;
this.Validate();
Thread.Sleep(350);
bool x = CC_StartStop(CC_PROGRAM);
    pb.Value = i++;
this.Validate();
Thread.Sleep(20);
    pb.Visible = false;
}
privatevoid Stop()
{
    CC_StartStop(CC_STOP);
}

// events
privatevoid BTN_Start_Click(object sender, EventArgs e)
{
    cnt = 0;
if (Mode == MODE_MANUALOFF)
{
    UpdateMode(MODE_MANUALON);
    ManualTemp = Convert.ToDouble(Man_Temp.Value);
    RunManual();
}
else
{
    UpdateMode(MODE_PROGRAM);
    RunProgram();
}
}
privatevoid BTN_Manual_Click(object sender, EventArgs e)
{
    UpdateMode(MODE_MANUALOFF);
}
privatevoid BTN_Stop_Click(object sender, EventArgs e)
{
    Stop();
if (Mode == MODE_MANUALON) UpdateMode(MODE_MANUALOFF);
else UpdateMode(MODE_INIT);
}
privatevoid BTN_ManualSet_Click(object sender, EventArgs e)
{
    ManualTemp = Convert.ToDouble(Man_Temp.Value);
    ManualSteepness = Convert.ToDouble(Man_Steep.Value);
    CC_SetManual(Convert.ToInt16(ManualTemp * 10), Convert.ToByte(ManualSteepness * 10));

int time = 0;
double DeltaTemp = Current_Temp[0] - ManualTemp;
if (DeltaTemp != 0) time = Convert.ToInt32((DeltaTemp) / ManualSteepness);

chart.Series[0].Points.Clear();
chart.Series[0].Points.AddXY(0, Current_Temp[0]);
chart.Series[0].Points.AddXY(time, ManualTemp);
chart.Series[0].Points.AddXY(60 * 60 * 24, ManualTemp);
}

// ethernet functionality
privatevoid Chart_AddDatapoint(logpoint data)
{
constdouble MAX_IO = 100;
constdouble MIN_IO = 0;
    count = (data.Hours * 3600 + data.Minuts * 60 + data.seconds) - startcount;
// Cabinet
}

```

```

chart.Series[1].Points.AddXY(count, data.Tempkast);
if (130 < data.Tempkast || data.Tempkast < -40)
{
    LB_Temp0.Text = "NC";
    Current_Temp[0] = 0;
}
else
{
    LB_Temp0.Text = data.Tempkast.ToString() + " °C";
    Current_Temp[0] = data.Tempkast;
}

// set 1
chart.Series[2].Points.AddXY(count, data.TempSensor[0]);
Current_Temp[1] = data.TempSensor[0];
if (130 < data.TempSensor[0] || data.TempSensor[0] < -40) LB_Temp1.Text = "NC";
else LB_Temp1.Text = data.TempSensor[0].ToString("0.0") + " °C";
if ((data.IO & 0x01) == 0)
{
    VBTN_IN1.BackColor = Color.Lime;
    chart.Series[6].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_IN1.BackColor = Color.Red;
    chart.Series[6].Points.AddXY(count, MAX_IO);
}
if ((data.IO & 0x10) == 0)
{
    VBTN_OUT1.BackColor = Color.Lime;
    chart.Series[10].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_OUT1.BackColor = Color.Red;
    chart.Series[10].Points.AddXY(count, MAX_IO);
}

// set 2
chart.Series[3].Points.AddXY(count, data.TempSensor[1]);
Current_Temp[2] = data.TempSensor[1];
if (130 < data.TempSensor[1] || data.TempSensor[1] < -40) LB_Temp2.Text = "NC";
else LB_Temp2.Text = data.TempSensor[1].ToString("0.0") + " °C";
if ((data.IO & 0x02) == 0)
{
    VBTN_IN2.BackColor = Color.Lime;
    chart.Series[7].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_IN2.BackColor = Color.Red;
    chart.Series[7].Points.AddXY(count, MAX_IO);
}
if ((data.IO & 0x20) == 0)
{
    VBTN_OUT2.BackColor = Color.Lime;
    chart.Series[11].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_OUT2.BackColor = Color.Red;
    chart.Series[11].Points.AddXY(count, MAX_IO);
}

// set 3
chart.Series[4].Points.AddXY(count, data.TempSensor[2]);
Current_Temp[3] = data.TempSensor[2];
if (130 < data.TempSensor[2] || data.TempSensor[2] < -40) LB_Temp3.Text = "NC";
else LB_Temp3.Text = data.TempSensor[2].ToString("0.0") + " °C";
if ((data.IO & 0x04) == 0)
{
    VBTN_IN3.BackColor = Color.Lime;
    chart.Series[8].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_IN3.BackColor = Color.Red;
    chart.Series[8].Points.AddXY(count, MAX_IO);
}
if ((data.IO & 0x40) == 0)
{
    VBTN_OUT3.BackColor = Color.Lime;
    chart.Series[12].Points.AddXY(count, MIN_IO);
}

```

```

else
{
    VBTN_OUT3.BackColor = Color.Red;
    chart.Series[12].Points.AddXY(count, MAX_IO);
}

// set 4
chart.Series[5].Points.AddXY(count, data.TempSensor[3]);
Current_Temp[3] = data.TempSensor[3];
if (130 < data.TempSensor[3] || data.TempSensor[3] < -40) LB_Temp4.Text = "NC";
else LB_Temp4.Text = data.TempSensor[3].ToString("0.0") + " °C";
if ((data.IO & 0x08) == 0)
{
    VBTN_IN4.BackColor = Color.Lime;
    chart.Series[9].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_IN4.BackColor = Color.Red;
    chart.Series[9].Points.AddXY(count, MAX_IO);
}
if ((data.IO & 0x80) == 0)
{
    VBTN_OUT4.BackColor = Color.Lime;
    chart.Series[13].Points.AddXY(count, MIN_IO);
}
else
{
    VBTN_OUT4.BackColor = Color.Red;
    chart.Series[13].Points.AddXY(count, MAX_IO);
}
chart.ChartAreas[0].AxisX.StripLines[0].IntervalOffset = count;

// user input
private settings GetSettings()
{
settings SET = new settings();
SET.AlarmTemp = newfloat[4];
SET.WarnTemp = newfloat[4];
SET.WarnOn = newbool[4];
SET.AlarmOn = newbool[4];

SET.WarnTemp[0] = (float)NUD_Alarm1.Value;
SET.WarnTemp[1] = (float)NUD_Alarm2.Value;
SET.WarnTemp[2] = (float)NUD_Alarm3.Value;
SET.WarnTemp[3] = (float)NUD_Alarm4.Value;

SET.AlarmTemp[0] = (float)NUD_Alarm1.Value;
SET.AlarmTemp[1] = (float)NUD_Alarm2.Value;
SET.AlarmTemp[2] = (float)NUD_Alarm3.Value;
SET.AlarmTemp[3] = (float)NUD_Alarm4.Value;

SET.WarnOn[0] = CB_A1.Checked;
SET.WarnOn[1] = CB_A2.Checked;
SET.WarnOn[2] = CB_A3.Checked;
SET.WarnOn[3] = CB_A4.Checked;

SET.AlarmOn[0] = CB_RL1.Checked;
SET.AlarmOn[1] = CB_RL2.Checked;
SET.AlarmOn[2] = CB_RL3.Checked;
SET.AlarmOn[3] = CB_RL4.Checked;
return SET;
}
#endif

#region Ethernet
// receiving
public void ClientStart()
{
this.Enabled = false;
ipbx ipbx = newipbx();
// Show testDialog as a modal dialog and determine if DialogResult = OK.
if (ipbx.ShowDialog(this) == DialogResult.OK)
{
    eth.Clientstart(ipbx.IPAddress, 31415);
}
ipbx.Dispose();
this.Enabled = true;
}

private void blink()
{
try

```

```

    {
        this.Invoke((MethodInvoker)delegate
        {
            VBTN_Data.BackColor = Color.Navy;
        });
        Thread.Sleep(200);
        this.Invoke((MethodInvoker)delegate
        {
            VBTN_Data.BackColor = Color.LightGray;
        });
    }
    catch { }
}
public float AvgFilter10(float val, byte ch)
{
    for (int i = 0; i < 9; i++)
    {
        avgval[i,ch] = avgval[i + 1,ch];
    }
    avgval[9,ch] = val;
    if (cnt < 50)
    {
        cnt++;
    }
    return val;
}

    val = 0;
for (int i = 0; i < 10; i++)
{
    val += avgval[i,ch];
}
val = val / 10;
return val;
}

private void ClientReceive(object sender, ReceiveEventArgs e)
{
// blink for incomming thread
Thread newThread = new Thread(() => blink());
newThread.Start();
const ushort Preamble = 0xF1F0;
// find preamble in stream
ushort search = (ushort)((e.Data[0] << 8) + e.Data[1]);
int i = 0 ;
if (search == Preamble)
{
    ushort Length = (ushort)((e.Data[i + 3] << 8) + e.Data[i + 4]);
    switch(e.Data[i + 2]) // check function
    {
        case 0xF0: // alarm
        if (Length == 0x0004)
        {
            byte channel = e.Data[i + 5];
            float temp = ((short)((e.Data[i + 7] << 8) + e.Data[i + 8])) / 10;
            string text = "Sensor: " + e.Data[i + 5].ToString() + ". Has reacht: " + temp.ToString();
            this.Invoke((MethodInvoker)delegate
            {
                ShowAlarm(text);
            });
            switch (channel)
            {
                case 0:
                    VBTN_A1.BackColor = Color.Red;
                    break;
                case 1:
                    VBTN_A2.BackColor = Color.Red;
                    break;
                case 2:
                    VBTN_A3.BackColor = Color.Red;
                    break;
                case 3:
                    VBTN_A4.BackColor = Color.Red;
                    break;
            }
            CC_EchoAlarm();
        }
        break;
        case 0xF1: // Stuur Data punt
        if (Length == 0x0013)
        {
            logpoint data = new logpoint();
            data.TempSensor = new float[4];

```

```

        data.TempSensor[0] = ((short)((e.Data[i + 5] << 8) + e.Data[i + 6])) / 10.0f;
        data.TempSensor[1] = ((short)((e.Data[i + 7] << 8) + e.Data[i + 8])) / 10.0f;
        data.TempSensor[2] = ((short)((e.Data[i + 9] << 8) + e.Data[i + 10])) / 10.0f;
        data.TempSensor[3] = ((short)((e.Data[i + 11] << 8) + e.Data[i + 12])) / 10.0f;
        data.Tempkast = ((short)((e.Data[i + 13] << 8) + e.Data[i + 14])) / 10.0f;
        data.Year = e.Data[i + 15];
        data.Month = e.Data[i + 16];
        data.Day = e.Data[i + 17];
        data.Hours = e.Data[i + 18];
        data.Minuts = e.Data[i + 19];
        data.seconds = e.Data[i + 20];
        data.IO = e.Data[i + 21];

    // filter
    if (startcount == 0)
    {
        startcount = (data.Hours * 3600 + data.Minuts * 60 + data.seconds);
    }
    this.Invoke((MethodInvoker)delegate
    {
        for (byte j = 0; j < 4; j++)
        {
            data.TempSensor[j] = AvgFilter10(data.TempSensor[j], j);
        }
        data.Tempkast = AvgFilter10(data.Tempkast, 4);
        Chart_AddDatapoint(data);
    });
}
break;
case 0xF2: // Stuur status
{
    if (Length == 0x0001)
    {
        CC_Modus = e.Data[i + 5];
    }
}
break;
case 0xF3: // Stuur alarm waarden
{
    if (Length == 0x0013)
    {
        AlarmSet.WarnTemp[0] = ((ushort)(e.Data[i + 5] << 8) + (e.Data[i + 6])) / 10;
        AlarmSet.WarnTemp[1] = ((ushort)(e.Data[i + 7] << 8) + (e.Data[i + 8])) / 10;
        AlarmSet.WarnTemp[2] = ((ushort)(e.Data[i + 9] << 8) + (e.Data[i + 10])) / 10;
        AlarmSet.WarnTemp[3] = ((ushort)(e.Data[i + 11] << 8) + (e.Data[i + 12])) / 10;

        AlarmSet.AlarmTemp[0] = ((ushort)(e.Data[i + 13] << 8) + (e.Data[i + 14])) / 10;
        AlarmSet.AlarmTemp[1] = ((ushort)(e.Data[i + 15] << 8) + (e.Data[i + 16])) / 10;
        AlarmSet.AlarmTemp[2] = ((ushort)(e.Data[i + 17] << 8) + (e.Data[i + 18])) / 10;
        AlarmSet.AlarmTemp[3] = ((ushort)(e.Data[i + 19] << 8) + (e.Data[i + 20])) / 10;

        AlarmSet.WarnOn[0] = b2bool(e.Data[i + 20], 0);
        AlarmSet.WarnOn[1] = b2bool(e.Data[i + 20], 1);
        AlarmSet.WarnOn[2] = b2bool(e.Data[i + 20], 2);
        AlarmSet.WarnOn[3] = b2bool(e.Data[i + 20], 3);

        AlarmSet.AlarmOn[0] = b2bool(e.Data[i + 20], 4);
        AlarmSet.AlarmOn[1] = b2bool(e.Data[i + 20], 5);
        AlarmSet.AlarmOn[2] = b2bool(e.Data[i + 20], 6);
        AlarmSet.AlarmOn[3] = b2bool(e.Data[i + 20], 7);
    }
}
break;
case 0xF4: // poll echo (moet UDP worden)
    CC_EchoAlarm();
break;
case 0xF5: // Stuur geheugen block

break;
case 0xF6: // Stuur Programma rij

break;
}
}

private void ClientDisconnected(object sender)
{
try
{
    this.Invoke((MethodInvoker)delegate
    {
        BTN_SaveLog.Visible = false;
        BTN_Connect.Visible = true;
        this.Text = "Climate Control | Disconnected!";
    });
}
}


```

```

        });
    }
    catch { }
}
privatevoid ClientConnected(object sender)
{
this.Invoke((MethodInvoker)delegate
{
    BTN_SaveLog.Visible = true;
    BTN_Connect.Visible = false;
this.Text = "Climate Control | Connected";
});
CC_SetTimeDate();
}
// sending
privatebool CC_AddSetPoint(point Set)
{
byte [] buf = newbyte[9];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x00; // functie voeg setpoint toe
buf[3] = 0x00; // lengte data
buf[4] = 0x04;
buf[5] = Set.Length;
buf[6] = (byte)(Set.Temp >> 8);
buf[7] = (byte)(Set.Temp);
buf[8] = Set.Steep;
return eth.Send(buf, 9);
}
privatebool CC_ClearSetpoints()
{
byte[] buf = newbyte[5];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x08; // functie verwijder setpoints
buf[3] = 0x00; // lengte data
buf[4] = 0x00;
return eth.Send(buf, 5);
}
privatebool CC_SetTimeDate()
{
byte[] buf = newbyte[11];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x01; // functie voeg setpoint toe
buf[3] = 0x00; // lengte data
buf[4] = 0x05;
buf[5] = (byte)(DateTime.Now.Year - 2000);
buf[6] = (byte)(DateTime.Now.Month);
buf[7] = (byte)(DateTime.Now.Day);
buf[8] = (byte)(DateTime.Now.Hour);
buf[9] = (byte)(DateTime.Now.Minute);
buf[10] = (byte)(DateTime.Now.Second);
return eth.Send(buf, 11);
}
privatebool CC_GetData(byte function)
{
byte[] buf = newbyte[6];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x02; // functie vraag data op
buf[3] = 0x00; // lengte data
buf[4] = 0x01;
buf[5] = function;
return eth.Send(buf, 6);
}
privatebool CC_StartStop(byte Modus)
{
byte[] buf = newbyte[6];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x03; // functie start of stop cc
buf[3] = 0x00; // lengte data
buf[4] = 0x01;
buf[5] = Modus;
return eth.Send(buf, 6);
}
// poll interface moet UDP worden vanwege broadcast
privatebool CC_SetManual(short Temp, byte Steep)
{
byte[] buf = newbyte[8];
buf[0] = 0xff; // preamble
buf[1] = 0xff;
buf[2] = 0x05; // functie voeg set manuel
buf[3] = 0x00; // lengte data
}

```

```

        buf[4] = 0x01;
        buf[5] = (byte)(Temp >> 8);
        buf[6] = (byte)(Temp);
        buf[7] = Steep;
    return eth.Send(buf, 8);
}
private bool CC_SetAlarms(settings set)
{
    byte[] buf = new byte[22];
    buf[0] = 0xff; // preamble
    buf[1] = 0xff;
    buf[2] = 0x06; // functie set alarmen
    buf[3] = 0x00; // lengte data
    buf[4] = 0x14;

    // warning temps
    ushort temp;
    temp = (ushort)(set.WarnTemp[0] * 10);
    buf[5] = (byte)(temp >> 8);
    buf[6] = (byte)(temp);
    temp = (ushort)(set.WarnTemp[1] * 10);
    buf[7] = (byte)(temp >> 8);
    buf[8] = (byte)(temp);
    temp = (ushort)(set.WarnTemp[2] * 10);
    buf[9] = (byte)(temp >> 8);
    buf[10] = (byte)(temp);
    temp = (ushort)(set.WarnTemp[3] * 10);
    buf[11] = (byte)(temp >> 8);
    buf[12] = (byte)(temp);

    // alarm temps
    temp = (ushort)(set.AlarmTemp[0] * 10);
    buf[13] = (byte)(temp >> 8);
    buf[14] = (byte)(temp);
    temp = (ushort)(set.AlarmTemp[1] * 10);
    buf[15] = (byte)(temp >> 8);
    buf[16] = (byte)(temp);
    temp = (ushort)(set.AlarmTemp[2] * 10);
    buf[17] = (byte)(temp >> 8);
    buf[18] = (byte)(temp);
    temp = (ushort)(set.AlarmTemp[3] * 10);
    buf[19] = (byte)(temp >> 8);
    buf[20] = (byte)(temp);

    // on off
    buf[21] = (byte)(Convert.ToByte(set.WarnOn[0])
        + (Convert.ToByte(set.WarnOn[1]) << 1)
        + (Convert.ToByte(set.WarnOn[2]) << 2)
        + (Convert.ToByte(set.WarnOn[3]) << 3)
        + (Convert.ToByte(set.AlarmOn[0]) << 4)
        + (Convert.ToByte(set.AlarmOn[1]) << 5)
        + (Convert.ToByte(set.AlarmOn[2]) << 6)
        + (Convert.ToByte(set.AlarmOn[3]) << 7));
}
return eth.Send(buf, 22);
}
private bool CC_EchoAlarm()
{
    byte[] buf = new byte[5];
    buf[0] = 0xff; // preamble
    buf[1] = 0xff;
    buf[2] = 0x08; // functie antwoord op het gegenereerde alarm
    buf[3] = 0x00; // lengte data
    buf[4] = 0x00;
return eth.Send(buf, 5);
}
private bool CC_ResetRelay(byte CH)
{
    byte[] buf = new byte[6];
    buf[0] = 0xff; // preamble
    buf[1] = 0xff;
    buf[2] = 0x09; // functie antwoord op het gegenereerde alarm
    buf[3] = 0x00; // lengte data
    buf[4] = 0x01;
    buf[5] = CH;
return eth.Send(buf, 6);
}
#endif

private void BTN_SaveLog_Click(object sender, EventArgs e)
{
    SaveFileDialog saveFileDialog1 = new SaveFileDialog();
    saveFileDialog1.Filter = "Comma-separated values (*.csv)|*.csv";
    saveFileDialog1.FilterIndex = 1;
    saveFileDialog1.RestoreDirectory = true;
}

```


Ethernet class

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Net;
using System.Net.Sockets;
using System.IO;
using System.Threading;

namespace ClimateControl
{
    public class ReceiveEventArgs
    {
        public ReceiveEventArgs(byte[] s, int len)
        {
            Data = s;
            Length = len;
        }
        public byte[] Data { get; private set; }
        public int Length { get; private set; }
    }
    class Ethernet
    {
        // events
        public event ReceiveDataHandler ReceivedData;
        public delegate void ReceiveDataHandler(Ethernet m, ReceiveEventArgs e);

        public event DisconnectedHandler Disconnected;
        public delegate void DisconnectedHandler(Ethernet m);
        public event ConnectedHandler Connected;
        public delegate void ConnectedHandler(Ethernet m);

        // ethernet
        private struct ETH
        {
            public bool Run;
            public bool Send;
            public byte[] Data;
            public int Length;
        }

        ETH Eth;
        bool first = true;

        public bool Clientstart(string ip, ushort port)
        {
            if (!Eth.Run)
            {
                Thread newThread = new Thread(() => ClientThread(ip, port));
                newThread.Start();
                return true;
            }
            else
                return false;
        }

        public void Clientstop()
        {
            Eth.Run = false;
        }

        public bool Send(byte[] Data, int len)
        {
            if (Eth.Run && !Eth.Send)
            {
                Eth.Data = Data;
                Eth.Length = len;
                Eth.Send = true;
            }
            return false;
        }

        private void ClientThread(string ip, ushort port)
        {
            try
            {
                TcpClient tcpClient = new TcpClient();
                tcpClient.Connect(ip, port);

                if (tcpClient != null && tcpClient.Connected)
                {
                    NetworkStream stream = tcpClient.GetStream();
                    byte[] data;
                    // Write done
                }
            }
        }
    }
}

```

```

        Eth.Run = true;
    while (tcpClient != null && tcpClient.Connected)
    {
        // send
        if (Eth.Send)
        {
            stream.Write(Eth.Data, 0, Eth.Data.Length);
            stream.Flush();
            Eth.Send = false;
        }

        // receive
        if (stream.DataAvailable && stream.CanRead)
        {
            // Buffer to store the response bytes.
            data = newByte[720];
            // Read the first batch of the TcpServer response bytes.
            int bytes = stream.Read(data, 0, data.Length);
            // callback the data
            if (ReceivedData != null)
            {
                ReceivedData(this, newReceiveEventArgs(data, bytes));
            }
        }

        // check connection
        if (!tcpClient.Connected || !Eth.Run)
        {
            Eth.Run = false;
            tcpClient.Close();
        }
        // raise event
        if (Disconnected != null)
        {
            Disconnected(this);
        }
        return;
    }
    Thread.Sleep(1); // dont rush
    // raise event
    if (Connected != null && first)
    {
        Connected(this);
        first = false;
    }
    else
    {
        // raise event
        if (Disconnected != null)
        {
            Disconnected(this);
        }
        return;
    }
    catch (Exception ex)
    {
        Console.WriteLine(ex.Message);
    }
    if (Disconnected != null)
    {
        Disconnected(this);
    }
}
}

```