EcoDome – Intelligent Greenhouse Control System

Authors:
Hans V. Rasmussen
Frederik S. Gjødsbøl

Supervisor:
Søren Top

In association with VedvarendeEnergi-Sønderborg
and University of Southern Denmark

Author Note

Funded by Green Makerspace at VedvarendeEnergi-Sønderborg,

Møllestedgård, Vestermark 7 A-B, 6400 Sønderborg

Acknowledgement

We would like to thank Green Makerspace at VedvarendeEnergi-Sønderborg for giving us the opportunity to do this project, and to Mads Clausen Instituttet at University of Southern Denmark for providing us with equipment for making the necessary hardware. To Jerome Jouffroy, lecturer at SDU, for taking the time to discuss possible control strategies useful for our project, and thanks Elisa Skytte Eggert and Erik Henneberg for giving advice regarding electronic components. A special thanks to our supervisor Søren Top for supporting us troughout the project, and always being ready to give helpful advice.

Frederik S. Gjødsbøl Hans V. Rasmussen

Abstract

For the 6th semester bachelor project of the Mechatronics BSc in Engineering at University of Southern Denmark (SDU) a control system was developed for the Green Makerspace EcoDome, an experimental greenhouse which aims at extending the seasons and keeping the temperature within a set limit. This project focuses on control theory and the corresponding software, as the EcoDome already exists, and uses prognoses from local weather forecasting services as well as active heat storages.

To prepare the EcoDome for the prototype small changes were made to the hardware and mechanics, including installing flex tubes to increase the efficiency of the heat storages.

During the project a working prototype was developed which fulfilled the overall requirements of the project. The prototype is capable of regulating the environment in the EcoDome using air circulation and stone beds, however, the currently installed actuators are not powerful enough to keep the environment within the set limitations during the extreme weather experienced in May of 2018 in Denmark.

Keywords: SDU, Green Makerspace, EcoDome, control system.

Contents

Problem Formulation

Background

The Mechatronic Bachelor of science in engineering of the 6th semester in the Mads Clausen Institute of the University of Southern Denmark has focus on the bachelor project. The aim of the bachelor project is to demonstrate the student's qualified skills in expressing, analysing, and processing problems within a limited course-relevant and engineering-specific subject. The bachelor project must document the student's engineering-specific competencies attained during their work within this subject. The final project should include an independent delimitation, solution, discussion, and written report of a practical problem within the subject area.

Within these requirements an opportunity was found at VedvarendeEnergi-Sønderborg that required the understanding of Advanced Programming, Control Engineering, and Thermodynamics to make a device capable of using weather forecasting to intelligently manage a passive heat storage. Similar systems already exist, but none of these offer a solution for the normal hobbyist or worker and are normally specialized for a single building. Based on this, making an easily accessible and easy to setup solution would allow for cheap control solutions for greenhouses and other environments where a better energy efficiency is desired.

VedvarendeEnergi-Sønderborg is working on developing new methods, and improving current ones, for collecting water, creating energy, and generally elevating the living standards of people, with renewable resources in mind. As part of their vision of a better tomorrow for people living in suboptimal conditions, they are building a solution that is capable of assisting small-

scale food production by the means of an “EcoDome”. This EcoDome takes in water and energy in the form of electricity and uses these resources to extend the seasons and keep the plants alive.

For this purpose, a system capable of controlling the environment within the EcoDome with only minimal power consumption would mean that smaller solar panels would be needed, and thereby making such solutions more viable for people.

Existing systems

Currently there only exist a few temperature controlling solutions for the private user that wants weather forecasting for efficiency. One such solution would be the Tado° Smart Thermostat V3, which not only uses weather forecasting, but also calculates the heat flow through the house. The thermostat can control both air temperature and water temperature.

Just like the Tado° Smart Thermostat there are other solutions that aim to make housing more efficient, but none of these are open source or have the capability to control a customized passive heat storage. Furthermore, since they were made for housing, they don’t support airflow/watering or other agricultural features.

Problem description

As part of Green Makerlabs EcoDome it would be of interest to implement a solution that is not only capable of controlling the environment in the now, but also by looking at the imminent future. This solution should contain functionality to control passive heat storages in an intelligent way. Furthermore, a solution that is flexible and easy to implement in other projects would be desired, as this would allow for easier implementation in possible already existing setups.

Objective

The goal is to create a control system as seen many times before to control heat and other variables but should also be able to analyse the forecast of the local weather and adjust the heating and other variables accordingly. For weather forecasting the system should use a number of weather stations and forecasting services for rough estimation, and local sensors to measure the current state of the environment to be regulated. The system should be capable of using active as well as passive energy sources/storages to regulate the environment to be controlled.

Risk assessment

What could delay the project or prevent it from being completed, and how could this be prevented:

VedvarendeEnergi-Sønderborg: Probably the largest hazard to the project would be if the EcoDome was removed or otherwise destroyed, making it difficult to test the system in the desired environment. Another hazard from VedvarendeEnergi-Sønderborg side would be if the contact persons for the project were unavailable for a longer period, withholding important information. Furthermore, there is currently some unease among various members of VedvarendeEnergi-Sønderborg which could threaten the project further down the line. All these problems could cause major delays or maybe even the termination of the project, therefore it is important to keep up good communication and make sure that no misunderstandings happen. In the unlucky case that fatal problems with VedvarendeEnergi-Sønderborg were to happen, the project should be able to continue, although with limited resources.

Supervisor, Søren Top: The main hazard towards the project coming from the supervisor would be if he was unavailable for a longer period, be it due to sickness or vacation. Such absence would not be a big problem on its own, although if a different problem were to arise it

could prove much more difficult to resolve than under normal circumstances. Such a scenario could be prevented by giving a heads up if something were to happen. In the case that contact with the supervisor is lost the project will continue as normal. In this case it would be wise to find another professor at SDU to give advice until the contact with the supervisor is established again.

The development team, Hans Rasmussen & Frederik Gjødsbøl: The cause of delay with the largest impact on the project is if any or all members of the development team are unavailable to each other. One way to make sure this does not happen is to always be available on the chosen platform for communication, and that all data always is available on a shared data storage. Another problem could be eventual mutual disagreements, this would be another reason to keep communicating. There is a possibility that lack of information or a mutual disagreement might happen, but this should be negligible. Though in the case that it does happen, the supervisor or VedvarendeEnergi-Sønderborg should be involved.

Wrong/no weather forecast: It is possible that a weather forecast is wrong, or that none are available, at a certain time, resulting in a possible delayed test. This is not a huge hazard on its own, but it is present, and these delays could accumulate.

Time plan

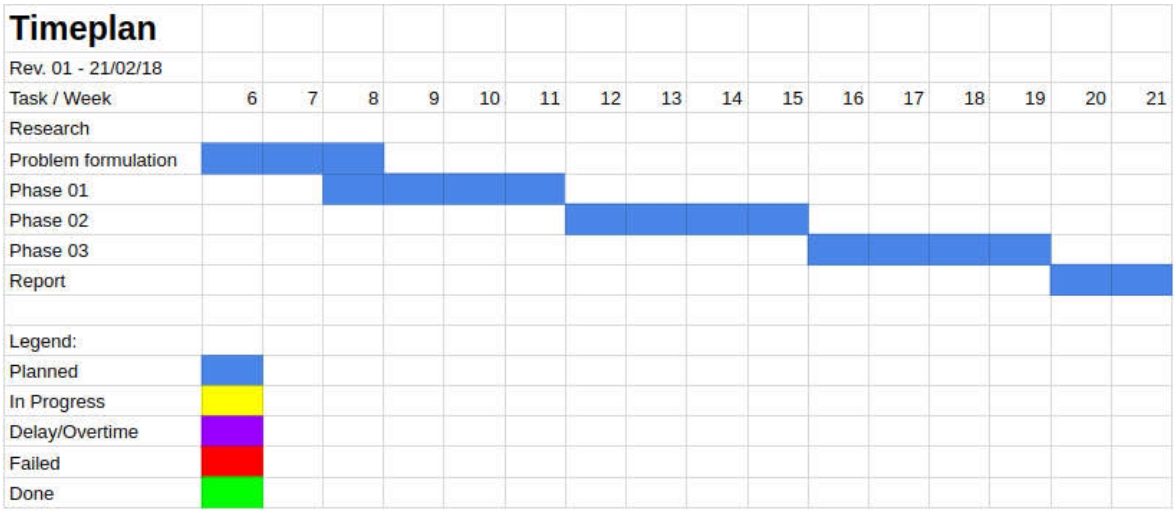

Figure 1Time plan (1st revision)

The time plan was made to show each phase, with the intention that the 3rd phase should act as a sort of buffer. The third phase would only be made if the core prototype was finished in its allocated time, although it was clear from the start that this was rather unlikely as delay always happen in this phase.

EcoDome pick-up state

Figure 2: Picture of the EcoDome.

The EcoDome was nearly complete when this project started. The greenhouse was built as part of an internship in spring 2016. There are wooden planting boxes along the walls of the greenhouse as well as four plastic 1000-liter tanks in the center.

It contained stone beds along the wall, located underneath the planting boxes, for passive heat storage, as well as various sensors and actuators. The sensors measure the temperature and humidity inside the EcoDome, and the actuators include a piston for opening and closing a window located at the top of the dome, a 14-inch fan for drawing air into the EcoDome, and 8-inch fans for drawing air into the stone beds. At the end of one of

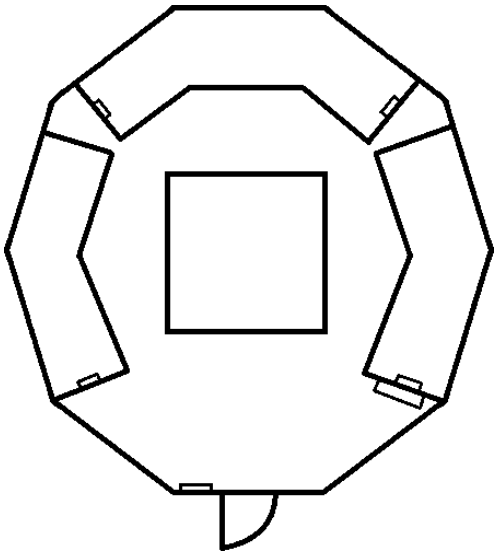

Figure 3 Top view of the EcoDome, with fans and control box, not to scale.

the planting beds was a box that would house the control system. The various components were either not connected, or they would need to be turned on manually.

Control box

The control box was a mess, there were two Arduino Uno's controlling both temperature and humidity. Most connections were soldered, and these connections were isolated by hot glue instead of shrink flex. Additionally, isolation tape had been used on top of the hot glue. Often many connections were isolated by one big glob of glue and tape, making it impossible to disconnect a single connection without disconnecting the neighboring pins too. Many parts were not in use or only partially connected.

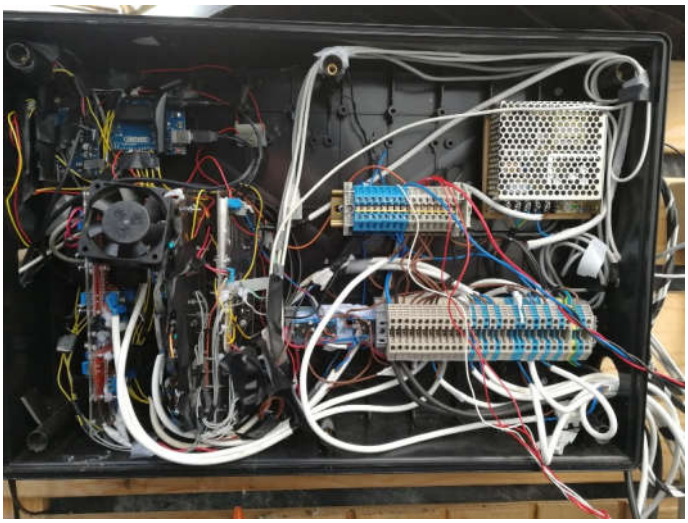

Figure 4 Control box before maintenance.

Besides the 12-volt connection from the batter that was placed outside the EcoDome and which was powered by solar power, an auxiliary power supply had been installed and wired into the 12-volt rail with the help of a relay to change between battery and auxiliary. Although the auxiliary was connected, the relay was in an exceptionally bad state, legs broken off and some connections nowhere to be found.

The actuator was currently not connected to the window, hanging from its support. Inside the control box it was assigned to two relays that were not in operational state either.

The main fan was connected to an L298N motor controller, but with its amperage draw of at least 5 amperes (the L298N is rated for 2A cont. 3A burst) the motor controller ran rather hot, to the point where a separate computer fan was installed to cool down the motor controller. The stone bed fans were connected two to one input, resulting in an amperage draw close to the limit of the motor controller.

The circuit for controlling the water valves were made rather spacious, to the point where 4 outputs filled as much as the 6 motor controllers. The temperature and humidity sensors were also mounted to use as much space as possible, covered with an extra layer of isolation tape.

Actuators

Stone bed fans

The EcoDome has three planting boxes with stone beds underneath along the edge. The two stone beds at either side each have a fan at one end while the one in the middle has a fan at either end. The fans were connected two and two; the ones at the sides together, and the ones in the middle together. The fans would draw in air from the floor in an attempt to heat up the stone beds.

Figure 5: Stone bed fan before the modification.

The fans are 8-inch radiator fans, their center has a diameter of 4.5 inches leaving an area of $0.0235\ m^2$ for air intake. The air flow through the fans has been measured to be $v = 9\frac{m}{s}$. This means that they can be expected to draw at least $12\frac{m^3}{min}$ into the stone beds. However, just a few centimeters behind the fan is a wooden board that leaves only a small gap for the air to enter the stone beds, limiting the airflow significantly, so this could possibly be much higher.

Window

The actuator in the window is an LA 22.50 100-24 VDC linear actuator from Linak, it normally requires 24 volts, but can be run with 12 volts with the possibility that it gets stuck when fully closed, requiring 24 volts to get it moving again.

Figure 6: Window with actuator and sensor.

Fan

The large fan beside the door is a 14-inch radiator fan, its center has a diameter of 4.5 inches leaving an area of $0.09\ m^2$ for air intake. The air flow

Figure 7 Main fan.

through the fan has been measured to be $v = 14.2\frac{m}{s}$. This means that it can be expected to draw at least $78\frac{m^3}{min}$ into the EcoDome.

Stone beds

As mentioned in the report from the first project about the EcoDome, the planting boxes were built as high beds to protect the plants from mice and to allow some space underneath the plants. This space was then taken up by the stone beds. The report from this previous project states that the pressure in the stone beds changes by $35\ Pa$ when subjected to an airflow.

Perl hose

The EcoDome uses perl hoses to water the planting boxes along the wall, two in each box running parallel to each other. These hoses are connected to a garden hose that runs along the wall, using only one water pump to supply the perl hoses.

To keep the control system for this project simple the hoses were left out in the beginning, but with the possibility of adding them later if time would allow it.

Analysis

Before the building of the prototype can start, it is necessary to refine the problem formulation to give a better structure of how the project should be handled. To do this the analysis will classify the requirements by using the MoSCoW model, at least 3 project phases using the V-model, and a risk assessment. After this, the calculations and research of the project will be presented.

MoSCoW

Requirements:

The following requirements must be met during this project:

- Applied weather forecasting.
- Control systems for passive heat storage.
- Easy to implement; Parts must be readily available

Must have:

- Local weather forecasting
  - Included solution for downloading and preprocessing incoming data (from custom nodes or weather forecasting services)
  - Temperature control system including passive heat storage
- Program for calculating passive heat storage capacity
  - Uses weather forecasting to manage passive heat storage intelligently
- Local Hardware interaction
  - Sensors
    - Inside temp.

        - Stone bed temp
    - Actuators
        - 2 on/off relays
        - 1 CW/CCW/off relays

Should have:

- A number of general purpose PID-tuned control systems
- IoT - Core capable of communicating with other cores (and nodes?) through internet
- outside weather observation station
- Security related considerations

Could have:

- Humidity control system
- Airflow control system (for keeping animals and plants alive)
- Possible design ideas/recommendations for passive heat storage systems

Won't have:

- Requires internet connection to be present

Product Risk assessment

Risks: cause and effect, prevention, recovery (table)

Short circuit - Caused by humidity. System might shut down, probably burnt components. Could be prevented by proper insulation (rare/catastrophic)

Power loss - Assuming solar panels: Caused by too little sunshine during the day, system shuts down, possible data loss. Could be prevented by back-up generator (possible/marginal)

Component failure - Caused by usage/wrong installation. Can result in system being unable to continue operation. Can be prevented by using high-grade components, sending out error logs would be advised (possible/Critical)

Internet connection lost - Caused by user failure, servers going offline, component failure. Results in prognoses failing to download. Can be prevented by backup connection, the device can continue work without prognosis input, although rather unreliable. (almost certain/marginal)

Extreme weather - Systems may shut down, and vegetation might die. Effects can be prevented by using weather forecasting to completely open or close ventilation before crisis strikes (heat wave, storm, flood). (unlikely/critical, NOTE: assuming environment to be controlled is sufficiently tough, the impact-level can be lowered to Marginal or Negligible)

Table 1 Risk fatality

	Negligible	Marginal	Critical	Catastrophic
Almost Certain	High	High	Extreme	Extreme
Likely	Moderate	High	High	Extreme
Possible	Low	Moderate	High	Extreme
Unlikely	Low	Low	Moderate	Extreme
Rare	Low	Low	Moderate	High

Development stages

The project will include three development phases, represented by separate V-models. The first phase will be for making a proof of concept that only includes basic functionality, such as downloading weather prognosis and putting out data for controlling other devices/modules.

The second phase will include the “must haves” that the project will include and will probably take the most time. This is also the phase that is expected to be finished, further phases are optional and only done if time allows to do so.

The third phase will include the should have and other optional features, such as moisture controller or customized nodes for better results.

Proof of Concept

Phase one consists of a proof of concept and should be quick to do. The prototype built in phase one will only be able to do very basic functionality, this phase is to show and make sure that the project is indeed possible to make.

To be able to create a device that is capable of using weather forecasting to control an environment, it is necessary to show that one indeed can download the prognosis for the next few hours. The device should also be able to make a signal that can be used by other devices.

This phase should also show to what extend one can estimate an already existing passive heat storage without disassembling it.

Stage 1: Pre-analysis

The pre-analysis consists of the problem formulation.

Stage 2: Analysis

The analysis of the Proof of concept should discuss topics regarding where to get the prognoses from, and how to calculate the passive heat storages capacity.

Stage 3: Design

The design stage should discuss topics regarding the device and what programming language that will be used in the proof of concept. These decisions might move onto the core prototype.

Stage 4: Implementation

One would ideally make 2 pieces of code that use files/semaphores to communicate with each other. The programs should be capable of downloading and analyzing the data.

Find the heat capacity of a passive heat storage.

Stage 5: Integration tests

The integration tests will make sure that the individual modules/programs work as expected.

Stage 6: Acceptance test

Combine all the modules and run a test to see whether the proof of concept can be used to prove that this project indeed is possible to make.

Stage 7: Deploy

The deploy phase is this report.

Core Prototype

This phase might be the most difficult one, as it creates the core prototype for the finished solution, therefore no further development can be done without finishing it. The core prototype will include the result from phase 01 as well as the “Must haves” from the MoSCoW stated in the problem analysis.

Stage 1: Pre-analysis

The pre-analysis consists of the problem formulation.

Stage 2: Analysis

Problem specifications, what exactly should the core prototype include?

What calculations are to be taken into consideration when making the core prototype?

Stage 3: Design

The platform will be the one decided upon in the analysis, the same goes for the programming language as this will build the base for every further work. These decisions might be inherited from the proof of concept.

Stage 4: Implementation

The core prototype will consist of at least 3 parts: a solution to download and analyses the weather data, a control system for controlling actuators, and a solution to acquire the actual local weather.

Stage 5: Integration tests

The integration tests of the core prototype might inherit the work from the prrof of concept, given that those modules already have entered a functional and suficient state of compatibility. The rest of the tests will simply check if the expected behavior is met.

Stage 6: Acceptance test

The acceptance tests will determine how much of a difference the individual actuators present when controlled by the finished control system, as well as a test to make sure that the whole control system with all actuators and sensors works reliably.

Stage 7: Deploy

The deploy phase is this report.

Additional features

The third phase will only happen if the second phase goes faster than anticipated, as, although the time plan will be made to fit the third phase too, delays are bound to happen.

The third phase consists of all “should have” and “nice to haves”. Although this phase realistically seen would require a lot of time, it will only be assigned to the spare time that also is to be used as a buffer in case the second phase gets delayed.

Stage 1: Pre-analysis

The pre-analysis consists of the problem formulation.

Stage 2: Analysis

What additional functionality would be meaningful for the project, and would such features be possible?

Stage 3: Design

If there are any choices or design problematics, these are to be discussed.

Stage 4: Implementation

The different features are to be split into modules or integrated into existing ones

Stage 5: Integration tests

If any changes were made to existing modules, extensive tests need to be made to make sure that no functionality has been lost, new modules also need to undergo testing.

Stage 6: Acceptance test

The device needs to still fulfill the basic functionality from phase 2, extra functionality should be working without interference.

Stage 7: Deploy

The deploy phase is this report.

Passive heat storage

There are different ways of collecting and storing heat without having to run a generator that will consume a lot of electricity. These mainly utilize the heat from the sun, and by far the best is a tank filled with water, followed by a mound or collection of stones or rocks, and at the bottom of the list are wood and dirt.

Passive heat storage is sometimes called sensible heat storage. Sensible heat refers to a temperature change in a body while other variables, such as volume or pressure, remain unchanged. Its counterpart, latent heat, relates to the phase change of a body, i.e. melting and boiling points, where the temperature remains unchanged. For heat storage in a greenhouse latent heat shouldn't need to be taken into consideration, not even if the heat is stored in water tanks unless the greenhouse is located in a very hot area.

Water vs stone

Water has a specific heat of $4.187\,\frac{kJ}{kg \cdot K}$ while stone lies at $0.84\,\frac{kJ}{kg \cdot K}$ making water almost five times better than stone at storing heat. Stone, however, can reach much higher temperatures, and is known to be used for heat storage at $600\,^{o}C$[1]. This means that a stone bed will be able to

[1] https://ing.dk/artikel/nyt-energilager-skal-opsamle-groen-energi-varme-sten-189135

store the heat longer than a water tank but the heat transfer to and from the greenhouse will be much slower.

Though water might seem to be the better choice due to its significantly greater heat capacity, stone does have a few advantages. One is that it can reach much higher temperatures, giving it a heat storage potential of a greater ratio than that of the specific heat, though this might not be relevant for a hobbyist greenhouse like the EcoDome. Another advantage is in the way the heat is distributed throughout the greenhouse. In a stone bed it would suffice to just blow air through to move the heat, whereas the water needs to run through pipes and heat up the surrounding air along the way.

In the EcoDome

The EcoDome uses stone beds with various stones such as limestone, flint, granite, and others, for passive heat storage, situated underneath the soil.

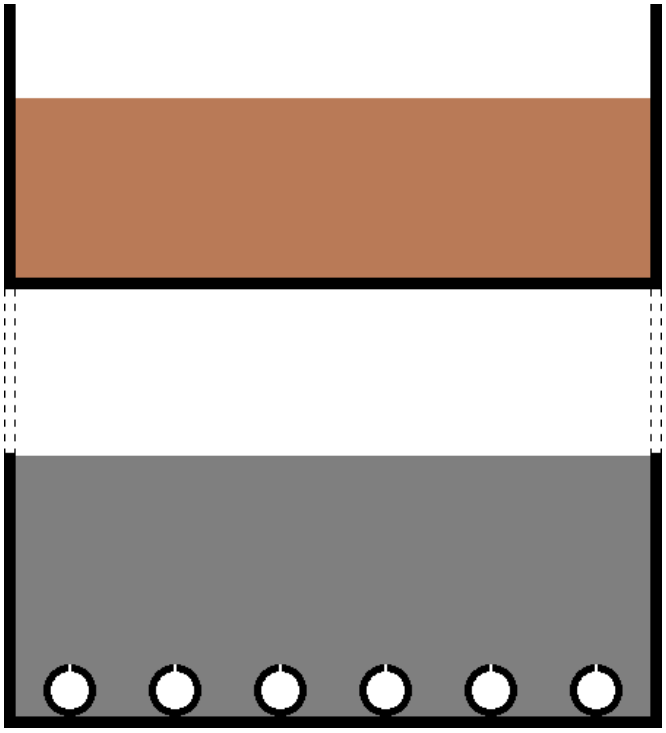

Figure 8: Cross-section view of stone bed and soil, not to scale.

The simplest model for heat storage in the stone beds to analyse would be a static one, meaning there is no air flow.

$$Q = m \cdot c \cdot \Delta T$$

where m is the total mass of the stones, c is the specific heat capacity of the stone-mix, and ΔT is the change in temperature.

This presents a good estimate of how much heat the stone beds are capable of storing. Next step would then be to have an air-flow through the stone beds transferring heat to the stones through convection:

$$\dot{Q} = h_c \cdot A_s \cdot (T_s - T_{ambient})$$

where h_c is the convective heat transfer coefficient of air. Expecting a slow air flow through the stone beds, it can be approximated to $h_c = 10 \frac{W}{m^2 \cdot K}$[2]. Because of what is visible of the entrance to the stone beds, the air can be assumed to flow into a set of parallel tubes, as that seems like the best reason for this setup. The tubes have holes along their periphery. Through these holes, the air flows into the stone beds and exits them at the top, which is open to the environment. This means that the heat transfer to the stone beds can be divided into two parts; first, air flows through the tubes and into the stone beds until a temperature barrier is formed just above the tubes, with the air below the barrier having a uniform temperature close to the one of the air being drawn into the tubes, then the temperature barrier starts moving upwards with a uniform surface area through the stone beds. The first part is expected to happen fairly quickly, and can therefore be ignored, while the heat transfer for the second part can be calculated as heat convection.

---

[2] https://www.engineersedge.com/heat_transfer/convective_heat_transfer_coefficients__13378.htm

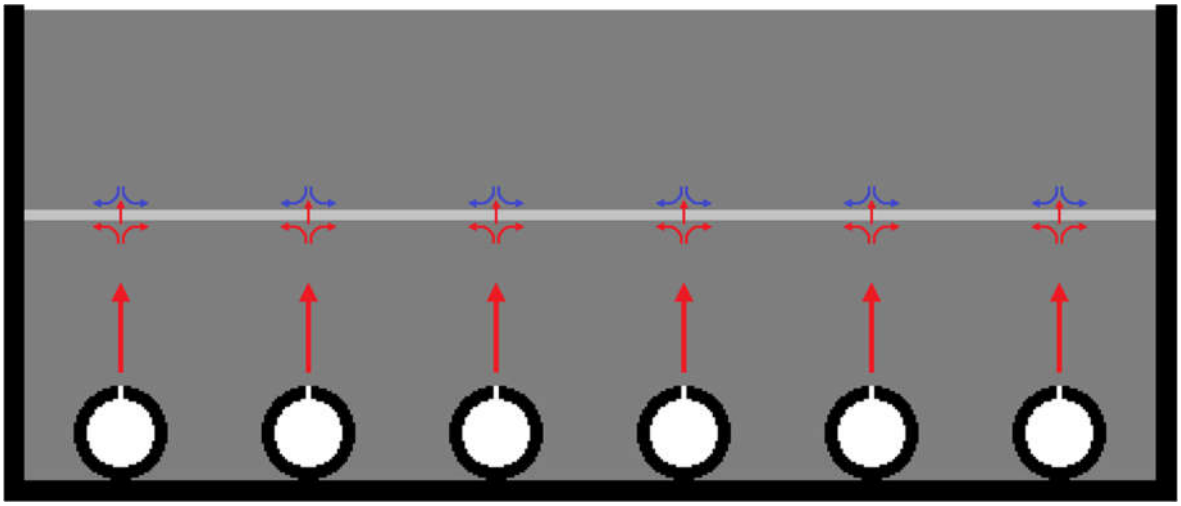

Figure 9: Cross-section view of heat moving through stone bed, not to scale.

Currently the fans are drawing in air from down near the floor, at a height with the stone beds where the temperature is relatively low compared to higher up in the EcoDome, and close to the outside temperature. This results in the air around the stone beds simply being circulated without much of a chance of changing the temperature, relying heavily on the sun heating up the EcoDome to be able to store any heat in the stone beds. It is possible to increase the heat stored in the stone beds by drawing in air from higher up in the EcoDome where the air is warmer. An easy way to do this could be to install some tubes that run from the fans in the stone beds and up near

Figure 10 Flex tube that pulls down hot air and into the stone bed.

the ceiling of the EcoDome. This could potentially increase the temperature of the air flowing through the stone beds by $10 - 15\ ^{o}C$, which would increase the heat storage significantly.

The layout of the planting boxes can be seen below, with a surface area of $A_{mid} = 5.55\ m^2$ for the middle box, and $A_{side} = 3.11\ m^2$ for the boxes at either side, totalling at $A_{total} = 11.77\ m^2$; this does not include the tanks in the center as they do not contain a stone bed. With the stone bed having a height of $h = 0.4\ m$ their volumes are then; $V_{mid} = 2.22\ m^3$, $V_{side} = 1.24\ m^3$, and $V_{total} = 4.7\ m^3$.

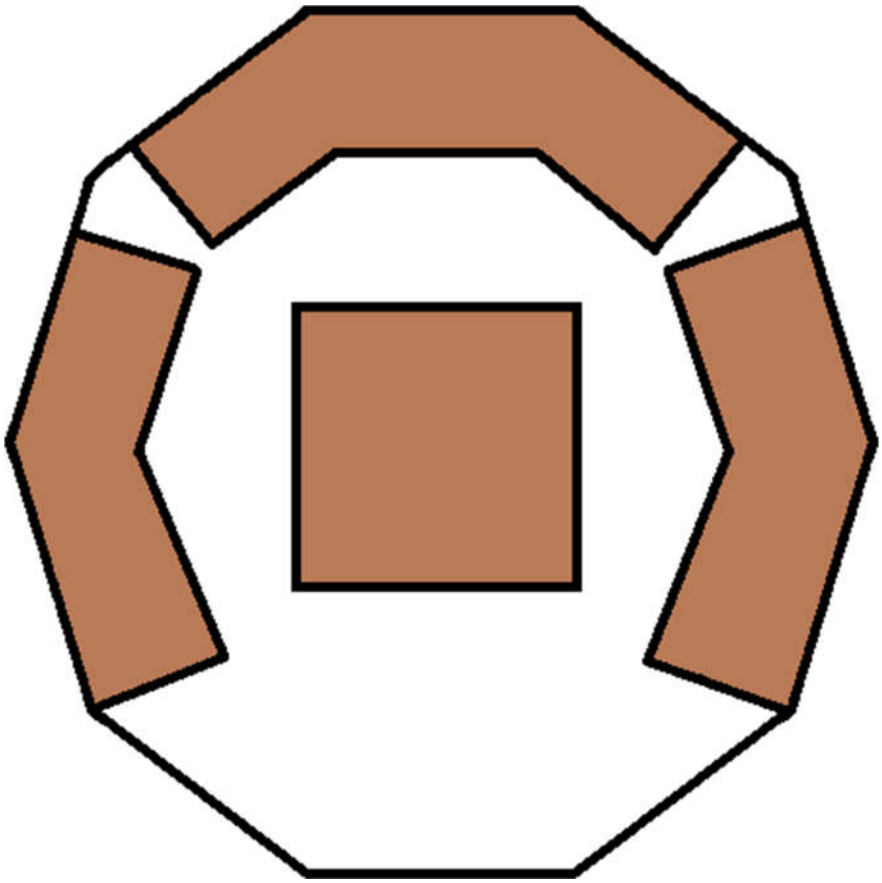

Figure 11: Top view of planting boxes, center box doesn't have a stone bed, not to scale.

Ten stones of varying size and kind were picked out and measured to estimate an average mass and volume for a single stone in the stone bed. These values were then used to calculate an average density of $\rho_{avg} = 292.17\ \frac{kg}{m^3}$, which was used to estimate the total mass of the stone

bed; $m_{total} \approx \frac{2}{3} V_{total} \cdot \rho_{avg} = 917\ kg$. Since the stones cannot be packed tightly enough to fill the entirety of the boxes they are estimated to only take up ⅔ of the space. With the various kinds of stone being present in the stone bed its specific heat has been estimated at $c_p = 0.84 \frac{kJ}{kg \cdot K}$.

Heat transfer between EcoDome and outside

The walls of the EcoDome are made from low density polyethylene, which has a thermal conductivity of $k = 0.33 \frac{W}{m \cdot K}$. The heat transfer through these walls can then be calculated, for thermal conductivity:

$$\dot{Q} = k \cdot A \cdot \frac{(T_{in} - T_{out})}{\Delta x}$$

For a temperature difference estimated to be somewhere between $5\ ^oC$ and $10\ ^oC$, and the walls having a thickness of $\Delta x = 0.0001524\ m$, the heat transfer from the EcoDome to the outside is then between $\dot{Q} = 552.5\ kW$ and $\dot{Q} = 1105\ kW$.

But heat is also transferred the other way as the sun hits the surface of the EcoDome, through thermal radiation:

$$\dot{Q} = A \cdot e \cdot \sigma \cdot (T_{sun}^4 - T_{out}^4)$$

where e is the emissivity of the EcoDome, and σ is the Stefan-Boltzmann constant. The surface area getting hit by the sun's rays does, however, also change throughout the day. In the morning and the evening, the sun hits the EcoDome from the side, essentially only hitting half of the dome, which is only a quarter of a sphere, whereas at noon the sun hits the entire dome from above, now half a sphere.

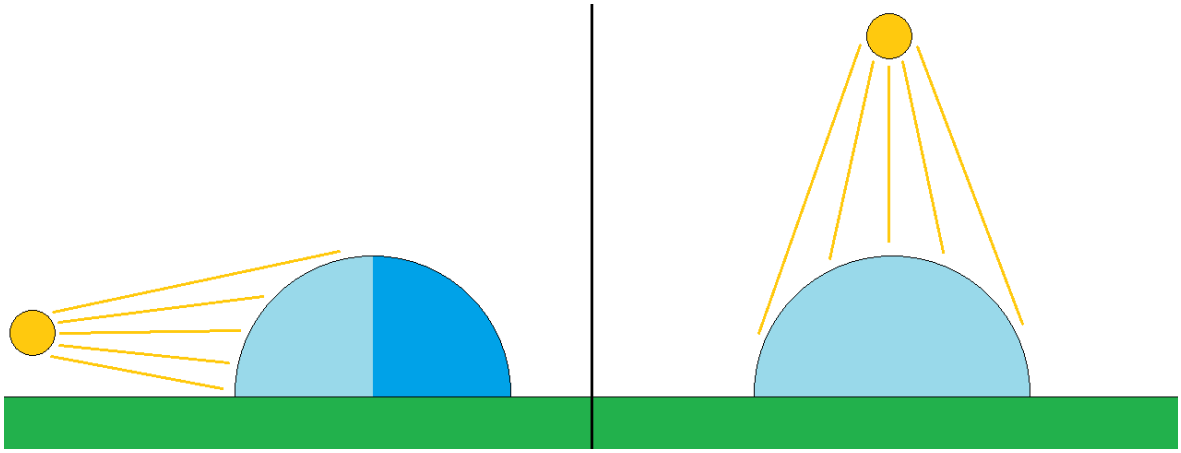

Figure 12: Sun hitting the EcoDome from the side, sun hitting the EcoDome from the top.

This means that throughout the day the area of the EcoDome that gets hit by the sun changes from $A = \pi \cdot r^2$ to $A = 2 \cdot \pi \cdot r^2$. Since the EcoDome is surrounded by trees to the south and the west and by buildings to the north, it does not catch much of the evening sun, but it does get the morning sun, which quickly raises the temperature inside.

Weather forecasting

Before starting work on the proof of concept, it is required to decide on which weather forecasting system to use. As Dansk Meteorologisk Institut is working on a new service called Beta,[3] it would be wise to consider this, as DMI provides some of the most accurate weather forecasts of Denmark. Other options include yr.no,[4] which is a Norwegian weather forecasting service, already incorporating some of the features that DMI's Beta is aiming for.

DMI's Beta is a project under construction, aiming to supply weather forecasts free of charge for use in projects regarding the weather. As the site is still under construction, only the

[3] https://www.dmi.dk/nyheder/arkiv/nyheder-2013/3/velkommen-til-betadmidk/
[4] https://www.yr.no/

basic weather forecasting for humans is available at the time of the EcoDome project. By E-Mail exchanges with DMI it was confirmed that the weather forecasts for bots and projects is not available yet, and that a subscription to the current weather forecasts will cost between 2000 to 4000 DKK per year. Contrary to DMI's Beta, yr already has set up a site that delivers weather forecasts free of charges. Because of the evident price difference it can be argued, that even though the forecasts of yr might be of lesser accuracy, it still would be a better option to choose yr's system that is stable and has little chance of undergoing major changes during the EcoDome project duration.

Design

Since the proof of concept does not need to control any environment, but simply show the output of some calculations, it is not necessary to run it on a minicomputer or microcontroller. Therefore, the proof of concept will be made to run on any Linux computer.

The handling of prognoses will be made so that the program can handle multiple prognoses at once, although this likely will not be implemented in the prototype, it gives the option to expand in the "Additional features" phase.

The proof of concept will consist of the following modules:

1. First it is necessary to download one or multiple sets of weather forecasts, this is done by a module that pre-processes the data and saves it to files that are easily accessible. The first module should be as simple as possible.
2. Second is the prognosis analysis module, which takes in the data from the first module, and then makes one variable to be used by the control system(s).
3. Third is the control system, that takes in the variable from the second module and controls the actuators, in the case of the proof of concept it will simply show the variable being passed from the second module and will not be given its own function nor class.

Programming language

Since this project is aimed partly for hobbyists, it is necessary to make design choices to make the reconfiguration of certain parts of the program that handle often changing tasks an easy task. To achieve this the, the program that handles the downloading and preprocessing of the data

will be made in Python. This is because Python is largely viewed as the go-to programming language for hobbyists and other non-professional folk.

The rest of the program will be made in C++ for Linux.

Program settings

Since the program for the EcoDome also needs to be easy to implement, it also needs to be user friendly to a limited degree. To make it easier to change the program to act according to the user's requirements, a configuration file and configuration parser will be implemented. The configuration file will control some constants that do not require further meddling with the code itself. These variables include the url for the prognosis downloader, the number of prognoses to be used in calculations, and the min, max, and desired temperatures. The configuration file holds potential for future improvement.

Controlling the EcoDome

The EcoDome has two separate methods for regulating the temperature; a fan and a window for drawing air into and out of the greenhouse, and a series of stone beds for storing heat for later use.

Various temperature sensors inside and outside the EcoDome feed the controller with data that is then used to calculate a controller output with a PI-controller. Based on this output, one or both methods will be used in the system.

The controlling itself uses a temperature sensor placed at the same height as the plants, while another sensor near the window measures the temperature of the air that is drawn into the stone beds, and a sensor on top of the stone beds measures the ambient temperature of the air

surrounding the stone beds. This temperature difference is used in the calculations for the heat storage in the stone beds. The outside temperature is measured and logged along with the rest, it is also used as a reference in calculating the Tref in the prognosis analyzer, as well as the main control system.

Control strategy

The general idea behind the control system is fairly simple. It can be divided into two parts; one part controls the environment in the EcoDome through sensors and actuators, while another part does predictive control, which helps to regulate the output of the first part.

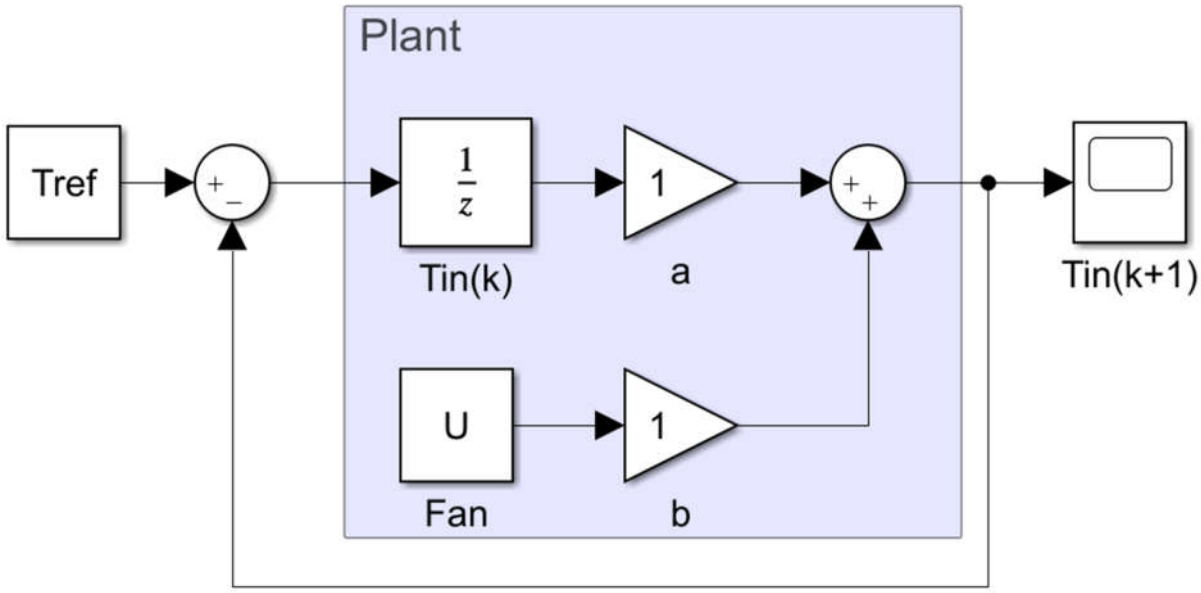

Figure 13: Basic control system.

The general idea for the control system itself is very simple; the temperature inside the EcoDome, measured by a sensor placed near the plants, is compared to a reference temperature. This difference in temperatures, or error, then determines how much the plant will try to correct

the error. The system's behavior can be expressed with the following state-space representation (S.S.rep.):

$$\begin{cases} x(t+1) = a \cdot x(t) + b \cdot u(t) \\ y(t) = x(t) \end{cases}$$

where a and b are constants that decide the aggressiveness of the system response.

Predictive control is generally rather complicated, so in this controller it is kept very simple. The predictive controller compares the temperature from a weather forecast looking 6 to 12 hours into the future to an optimal temperature set by the user. Whenever the temperature goes a certain amount above or below the predefined optimal temperature, a reference temperature is changed accordingly so that the system can start heating up or cooling down the EcoDome in advance in an attempt to keep the inside temperature within a certain range.

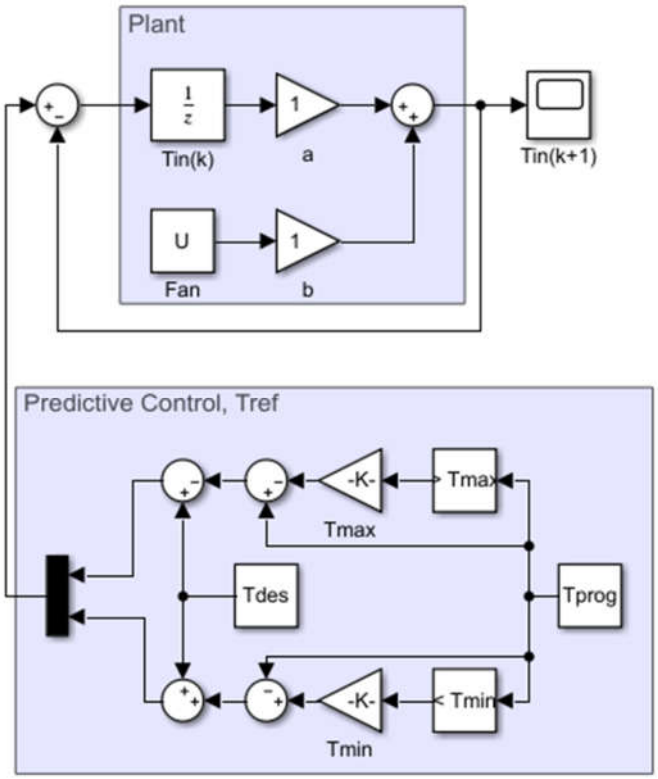

Figure 14: Predictive control regulating Tref.

Figure 15 shows the general idea of how Tref is regulated by looking at the temperature in the near future. Here, the temperature from the prognosis is just compared to the maximum and minimum temperatures of the optimal range; the one implemented in the actual controller is, and will still become, more complicated, taking various things into account such as the temperature difference between the inside and outside, and how this relates to the different cases.

PI-controller

The PI-controller is one of the most commonly used controllers in feedback control systems. It outputs a voltage, used to control some actuator in the system's plant, which is determined by summing up a proportional term and an integral term:

$$u(t) = K \cdot (r - y(t)) + \frac{K_E}{T_s} \cdot \int_0^t (r - y(\tau))d\tau$$

where K is the proportional gain, also written K_P, and $\frac{K_E}{T_s}$ is the integral gain, which can be written K_I. Sometimes, a derivative term is added to the equation, but since it is used for slowing down the system response to decrease overshoot, and the EcoDome system already runs slowly, due to changing temperatures, its gain, K_D, has been set to 0.

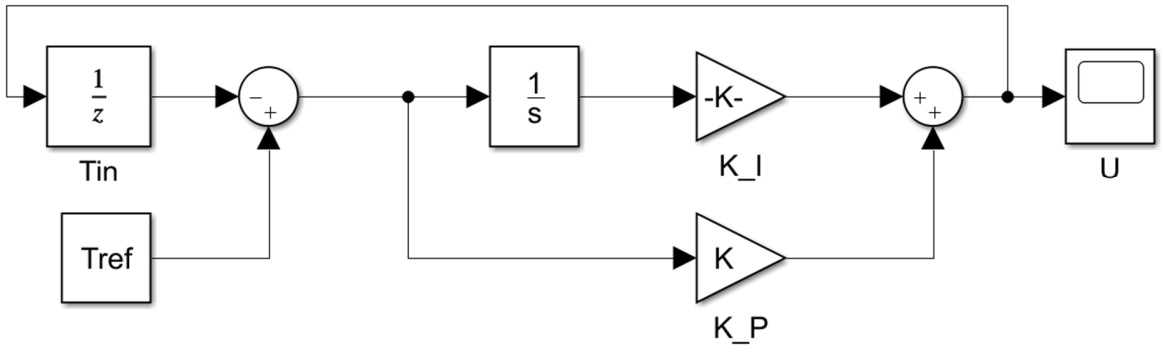

Figure 16: PI-controller.

The proportional term plays a rather obvious, but perhaps also the most important, part in any control system; the greater the error, the difference between the desired output and the actual output, the more the controller tries to compensate. The proportional gain determines how aggressive this compensation is.

The integral term works as a sort of memory; even if the error doesn't change through a series of iterations, possibly due to it being so small that the output from the controller has no effect on the plant, the integral term slowly accumulates it until it becomes significant enough for the controller to start correcting this.

As mentioned earlier, the controller outputs a voltage that regulates an actuator in the plant. However, since the EcoDome regulates the inside temperature, which changes slowly over time, it makes more sense to have the fans that work to regulate the temperature running at full speed all the time while they are needed, that is while the error is different from 0. Instead, the controller output works more like a reference; while $u(t)$ is within a certain range, only a specific part of the plant is working. This also means that the PI-controller won't need as much tuning as you would normally do.

The values for the gains K and K_E were found in Matlab with the acker-function, which takes a matrix A, a matrix B, and some arbitrary eigenvalues, here set to 0.6 and 0.2. The matrices A and B represent the system's S.S.rep.

$$\begin{cases} x(t+1) = a \cdot x(t) + b \cdot u(t) \\ y(t) = x(t) \end{cases}$$

where A relates to the changing temperature, $x(t)$, the output from the plant, which is multiplied by the constant a, and B relates to the plant's input, $u(t)$, which is multiplied by the constant b.

To simplify how the system responds, a and b have both been set to 1. This gives the gains $K = 1.2$ and $K_E = 0.32$.

Initially, the integral term was only concerned with one iteration, but it was later changed to a moving window taking in the ten most recent iterations. This increases $u(t)$ while also evening out the rate at which it changes. Normally, the integral term would accumulate the error from the system starts running, and then help slowly correcting it, but since the fans in the EcoDome are either on or off this won't change much in the way of how the system runs.

Implementation

The implementation will discuss the creation process and setup of the proof of Concept and Core prototype, as well as the programming environment that is common for both.

EcoDome Hardware

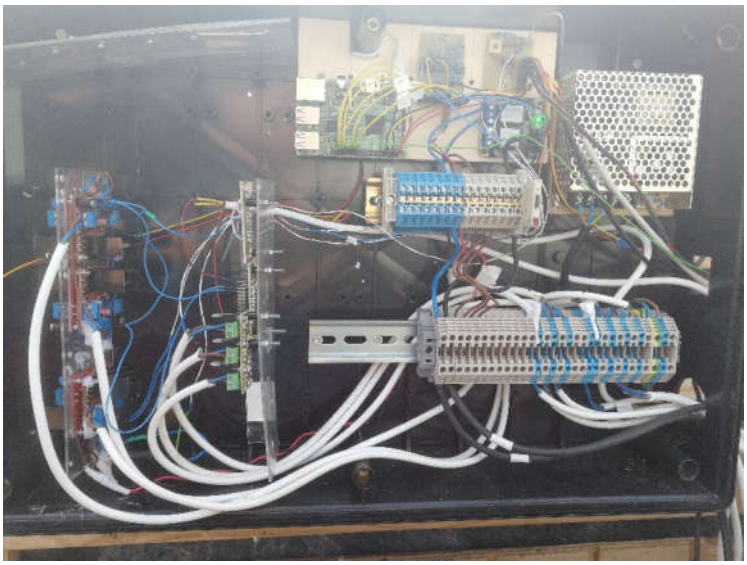

Figure 17 Control box after maintenance.

The first modification to the hardware was the removal and cleaning of the control box. All connections to the box where labeled before removal, followed by everything in the box being removed. After this the reusable parts, including the motor controllers and connection rails were built into the box again. Voltage dividers and op amps were implemented for good measure when using a 3.3v controller with a 5v system. A wooden plate was used to install the Raspberry Pi minicomputer that was meant to control the system, together with housing some newly installed relays and connectors for sensors.

The old humidity and temperature sensors were removed and replaced by DS18B20 temperature sensors using the one wire protocol, meaning that only one pin had to be sacrificed for all temperature sensors combined.

The window actuator was connected to an L298N motor controller and connected to the Raspberry Pi with capability to move both forward and backward. During the re-installation of

the window actuator with the window, a button was placed in the window and connected to the Raspberry Pi to make it possible to stop the window before it fully closes, removing the danger of the window actuator getting stuck. The stone bed fans remained connected the same way as before, but the connection to the Raspberry Pi was bridged to only use one pin for all 4 fans, since they only needed to turn one direction.

The main fan was connected to one of the relays, taking the load off the motor controllers. This removed the possibility of controlling it with a PWM signal.

The flex hoses were installed by making a wooden plate of 1x20x20 centimeter with a 15-centimeter diameter hole in the middle, the flex hose was then pulled through the hole and stapled to the backside. The diameter of the fan is 4.5 centimeter larger than that of the hose, resulting in the plate with the hose being mounted 3 centimeters away from the fan to allow airflow. To measure the temperature of the air flowing into the stone beds a small hole was cut in one of the flex tubes through which the sensor was put.

Figure 18: Flex tube mounted on board, with temperature sensor inside.

Proof of Concept

Both the download and preprocessing are done in a python script. This script can be called by the system, meaning that a system() call is to be used in the code. This method is not optimal, and should generally be avoided because of its slow nature, but is great as a placeholder or for tasks that easily can become complicated with traditional methods.

Downloading and preprocessing prognoses

The python script itself is rather simple. It takes in a path where to save the downloaded files, and a URL pointing to an xml page on yr.no, to find this page one can simply go to yr.no's page, search for a city and replace the last part of the URL that includes the '.html' with 'forecast.xml', which means that:

"https://www.yr.no/place/Denmark/South_Denmark/S%C3%B8nderborg/statistics.html"

Would look like:

"https://www.yr.no/place/Denmark/South_Denmark/S%C3%B8nderborg/forecast.xml"

The python script consists mainly of 4 parts; The first part is an argument parser, that takes in the strings passed when starting the script and saves them for later use. The second part uses the arguments to download and do the first round of processing to the .xml data, meaning that it converts it from html to a string. The third part of the script filters out the useful data, getting rid of all the brackets and other xml-related structure. The fourth and last part filters further so that it is only the actual value without name or anything which is then saved to the files.

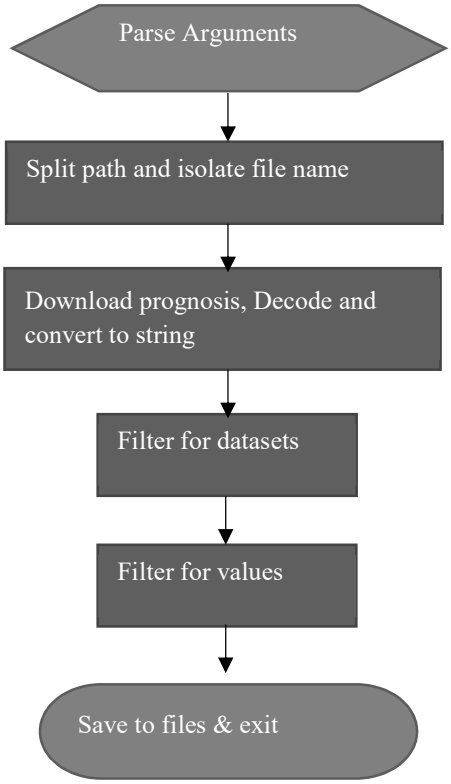

Figure 19 Flowchart describing the python prognosis downloading script.

Loading and processing prognoses

The C++ program uses a configuration file and the python script to process the data. The program can be split into four parts. First the program runs the DataDown python script, downloading and overwriting the files containing the prognosis data. Next the program reads the files and loads them into its memory, being a vector of structures that each contain one set of data. Third the program runs the prognosis analyzer, with the prognosis data as input. In this case the analyzer only calculates the mean of a number of datasets that are specified within the

configuration file. At last the output from the prognosis analyzer is passed to the control system. In the proof of concept cout is used to print it to the screen, acting as a placeholder for the actual control system.

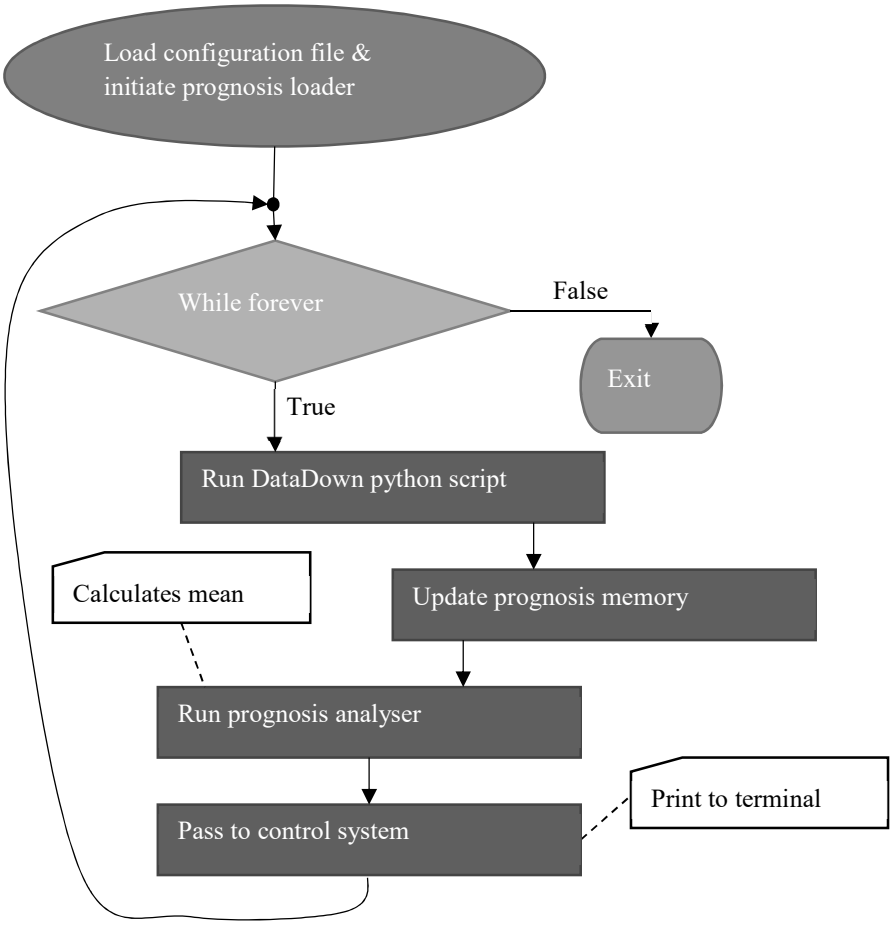

Figure 20 Flowchart of proof of concept

In this program, the datasets represent the data from one timestamp. The data from yr.no includes a timestamp in the ISO 8601 format, as well as wind direction in degrees, wind speed in meter/second, temperature in degrees Celsius, and pressure in mPa. All this info is then saved into a structure, and then into a vector.

Configuration Parser

To achieve more customizability in the core program, which is made in C++, the Libconfig[5] configuration file parser is used. Libconfig is a lightweight config parser that uses a file format that is more compact than XML and more flexible than INI, it is also type aware.

```
# Configuration file for EcoSoft

version = "1.0"

data =
{
    progdata =
    (
        {
            # The config file should have at least 1 dataset.
            # More datasets can be added by simply increasing the number after
'dataset'
            dataset1 =
"https://www.yr.no/place/Denmark/South_Denmark/S%C3%B8nderborg/forecast.xml";
        }
    );

    # the prognoses from yr.no include 38 prognoses with 6-hour increments,
    # meaning that setting prog_number = 2, the prognosis will look 0 to 6
hours into the future, =3 will be 6 to 12 hours and so on.
    # prog_dat descripes how many prognosis increments should be used for
calculations:
    prog_number = 3;

}
```

Figure 21 Code snippet showing the configuration file v1.

[5] https://hyperrealm.github.io/libconfig/

Regarding parsing of the configuration file, a class is made that opens the configuration file in the constructor and has functions to load each variable or variable group. This class is declared and used by the main function, where the variables also are distributed to the respective classes and threads that might use them.

Core Prototype

Whereas the Proof of Concept was a demo to make sure that the core intention of this project indeed is possible, the core prototype builds upon the proof of concept to form the first working device of this project.

Modifications to proof of concept

Although the prognosis downloader does not see much change, the prognosis analyzer only had a placebo calculation that calculated the mean value of the temperatures of different datasets till now. This has been changed, the constructor now takes in the maximum, minimum, and desired temperatures, and the prognosis analyzing function takes in the dataset vector, the inside temperature, the outside temperature, and the prognosis number.

The analysis itself checks whether the outside temperature, according to the prognosis of the corresponding prognosis taken into consideration, will become higher than the maximum allowed temperature for inside minus the difference between temperatures inside and outside. This difference is calculated for the temperature as it is at the moment of the iteration. Similarly, the analyzer also tests if the prognosis temperature will become lower than the minimum allowed temperature for inside plus the difference between temperatures inside and outside. If the inside temperature is estimated to rise above the maximum, then Tref is set equal to the desired

temperature minus the maximum temperature plus the difference between the inside and outside temperature, plus the outside temperature from the prognosis. The same process, but opposite happens if the inside temperature is expected to fall below the minimum allowed temperature.

```
float panalyse(vector< prognosis_data_structure > _progin_v, float _Tin, float
_Tout, int _pn)
{
    if (_progin_v[_pn-1].temperature > (Tmax-(_Tin - _Tout)))
    {
        //Tref = Tdes - ((_Tin - _Tout) + (_progin_v[_pn-1].temperature -
(Tmax-(_Tin - _Tout))));
        Tref = Tdes - ((Tmax - (_Tin - _Tout)) + _progin_v[_pn-
1].temperature);
        if(Tref < (Tmin + 0.5))
        {
            Tref = Tmin + 1;
        }
    }
    else if (_progin_v[_pn-1].temperature < (Tmin + (_Tin - _Tout)))
    {
        Tref = Tdes + ((Tmin + (_Tin - _Tout)) - _progin_v[_pn-
1].temperature);
        if(Tref > (Tmax - 0.5))
        {
            Tref = Tmax - 1;
        }
    }
    else
    {
        Tref = Tdes;
    }
    return Tref;
}
```

Figure 22 Code snippet showing the prognosis analyses function

Runtime user interface and Logging data

Since forcefully exiting the program can cause actuators to continue running and might cause damage to the window and/or overheating/undercooling, it was necessary to implement a

way to safely shutting down the program.

```
void InternalThreadEntry()
{
    while(tercon->pos())
    {
        /*...*/
    }
    digitalWrite(RELAY_1_P1, LOW);
    digitalWrite(L298N_STONE, LOW);
    window.WaitForInternalThreadToExit();
}
```

Figure 24 Code snippet showing the terminal controller controlling the exiting of the program.

Because of this the terminal_controller thread was created. Since the terminal_controller is a thread, it can run alongside the normal process of the main controller and other tasks without causing interference, this also causes it to respond very fast. Since the thread takes control of the terminal, it is preferred to use its write function instead of cout or printf to avoid fragmented terminal output.

Terminal logger

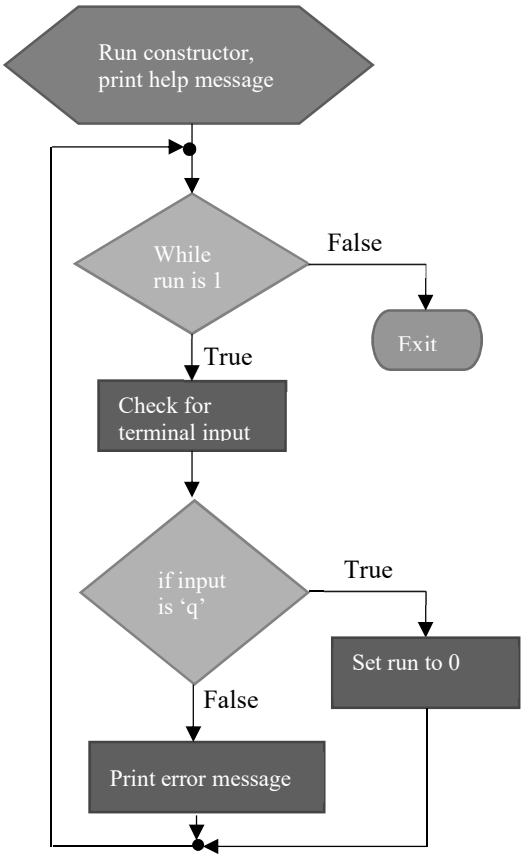

Figure 23 Flowchart showing the terminal controllers behavior for the quit command, more commands might be present in the future.

Besides handling terminal output, the terminal_controller also has a ‘help’ message together with a few commands, including the ‘q’ or ‘quit’ command, that sets the global program operation state variable to false, meaning that all

'while' syntaxes being reliable on this variable will stop at next iteration, giving the program possibility to run code before it exits.

```
pi@raspberrypi:~/programming/EcoDome_Main_dev $ ./Eco_Soft
Found 1 dataset(s).

--Help message--
Valid commands are:
        'help' 'h' '?'          - Shows Help message
        'q' 'quit' 'exit'       - Exits this program by stopping all processes and actuators

> q

Program shutting down on next iteration...
pi@raspberrypi:~/programming/EcoDome_Main_dev $ ./Eco_Soft
Found 1 dataset(s).

--Help message--
Valid commands are:
        'help' 'h' '?'          - Shows Help message
        'q' 'quit' 'exit'       - Exits this program by stopping all processes and actuators

>
```

Figure 25 Screenshot of the terminal, showing the interaction between user and program.

Besides being able to interface with the program, it is also necessary to have a convenient way to save the different variables in the program to a log file for later testing and analysis. This is accomplished by the LOGGER object. The constructor takes in a vector of strings that contain the names of the variables that are to later be saved. It checks if the 'logs' directory exists inside the working directory and creates it if not, it then checks if other log files already exist and makes a new numbered log file with the lowest number not already in use. When the log file is created and opened, it prints some basic data including the current data and time, and the time step of the controller. After this the logger prints the headers of the variables.

```
Test started at 2018-05-22.16:37:09
Time step is 10
TimeStamp    in soil      in window
16:37:15     30.687000    38.500000
16:37:25     30.687000    38.500000
16:37:35     30.687000    38.500000
```

Figure 26 Snippet from the logfile, showing the setup and 3 measurements.

After the constructor has finished, one can use the 'update' function to give it a string vector of all the variables that then are printed to the log file in an orderly manner. The logger

also gives a timestamp of the current time in hours, minutes and seconds that can be used as the x-axis when plotting.

Actuator interfacing, sensors, and main controller

Controlling the actuators comes down to making a class for the L298N and a thread for the window. The relays can be controlled with one line of code, so making a class or function is unnecessary.

The L298N class uses the constructor to prepare the pins, additionally it has functions for controlling the direction and duty cycle of a motor, including high impedance and off. The class is made so that the PWM only is enabled if an enable pin is defined during object creation.

The window is slightly harder to manage since it requires the program to constantly check whether the window has reached its closed state. Therefore, the window uses a threaded class instead of a conventional class. The window thread requires the two direction pins for the L298N and an additional input pin for the sensor in the window. The thread has two functions: open and close. When one of these functions is called, a to_do variable is flipped to true, which causes the internal thread to work.

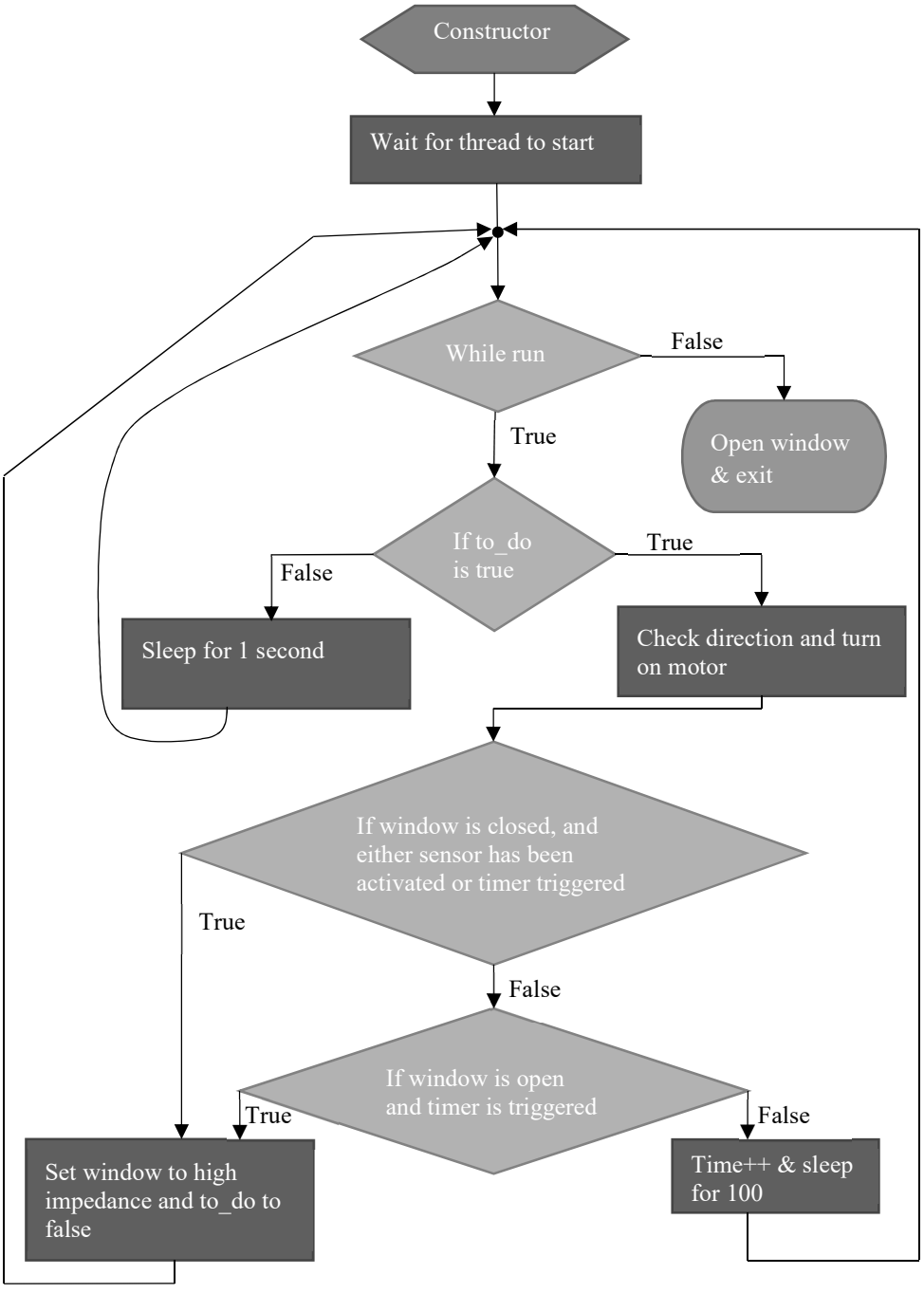

Figure 27 Flowchart about the window thread

The thread first sets the direction in which the window should move, it then checks whether the conditions for stopping have been satisfied. When closing the window, the

conditions for stopping are that either the sensor has been triggered, or that the set time has elapsed, whereas it for the opening of the window simply checks if the time has elapsed. If the stopping conditions are fulfilled, the thread sets the window to high impedance mode and sets the to_do variable to false. If the conditions have not been fulfilled it increments a timer and waits for 100 milliseconds. When the window is not moving it checks whether new commands have arrived every second. When the program is exiting, the window thread fully opens the window, as to prevent overheating of the EcoDome while the program is offline.

The most important functions that the main control system harbors include the controller function that is the PI controller, the plant function that decides on what to do with the actuators, and the internal thread which takes care of running the different functions in addition to administering the prognosis download, preprocessing and analysis. The main control system and the sensor reading class have been implemented as a threaded class, this is a result of both the reading of the sensors, as well as the downloading of the prognoses being rather slow, and as a result can easily exceed the timestep duration when the timestep is low. The sensor and controller class use

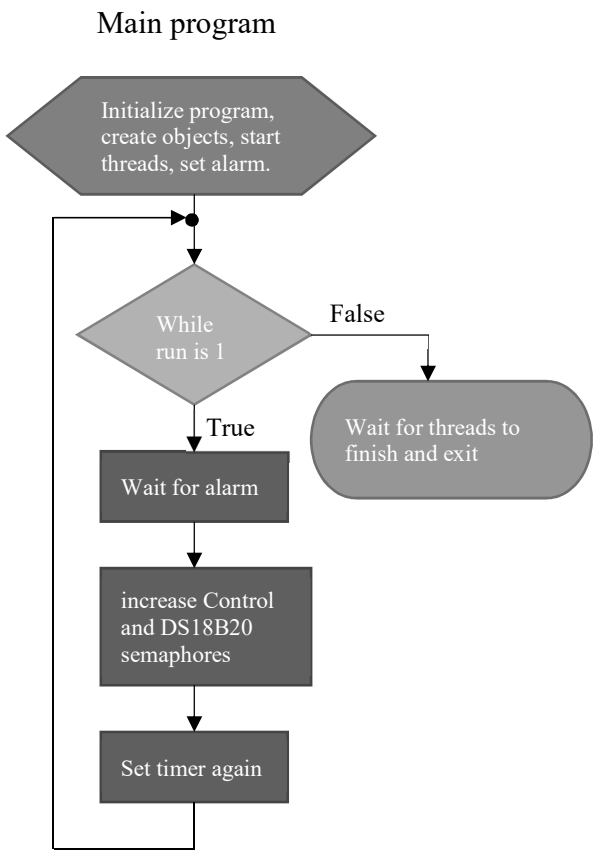

Figure 28 The main thread of the prototype

semaphores to coordinate, resulting in the alarm triggering both the sensor and control class, after which they start their slow work. When the sensor class has finished acquiring the new data it tells the control class so by using a shared semaphore. When the control class starts work as per result of the alarm from the main, it starts downloading and preprocessing the prognoses. After finishing the download and preprocessing the control class waits for the sensor class to finish, it then asks for the newest measurement data and passes it on to the prognosis analyzer, thereafter it calls the PI controller and plant. The plant was implemented so that the main circulation is limited when it comes to heating up the EcoDome, as the outside temperature needs to be hotter than inside. This restriction does not apply when cooling down, as any circulation is better than none in this case. Regarding the stone beds, the plant checks whether Tref is higher or lower than Tdesired or if u has reached a certain threshold. If any of these is true, then it is checked if the stone bed is hotter or colder than the current inside temperature, and if it is deemed that the stone beds can make a positive influence the stone beds are turned on.

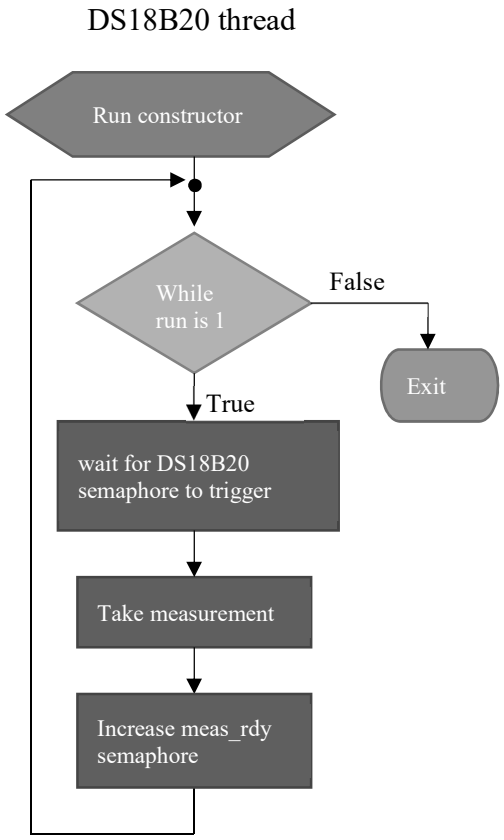

Figure 29 The sensor measurement thread of the main program.

The main control system constructor takes the same input as the prognosis analyzer does with an addition of two semaphores, one for starting next iteration and one for knowing when the sensor class is ready, and object pointers for the terminal controller and the sensor class.

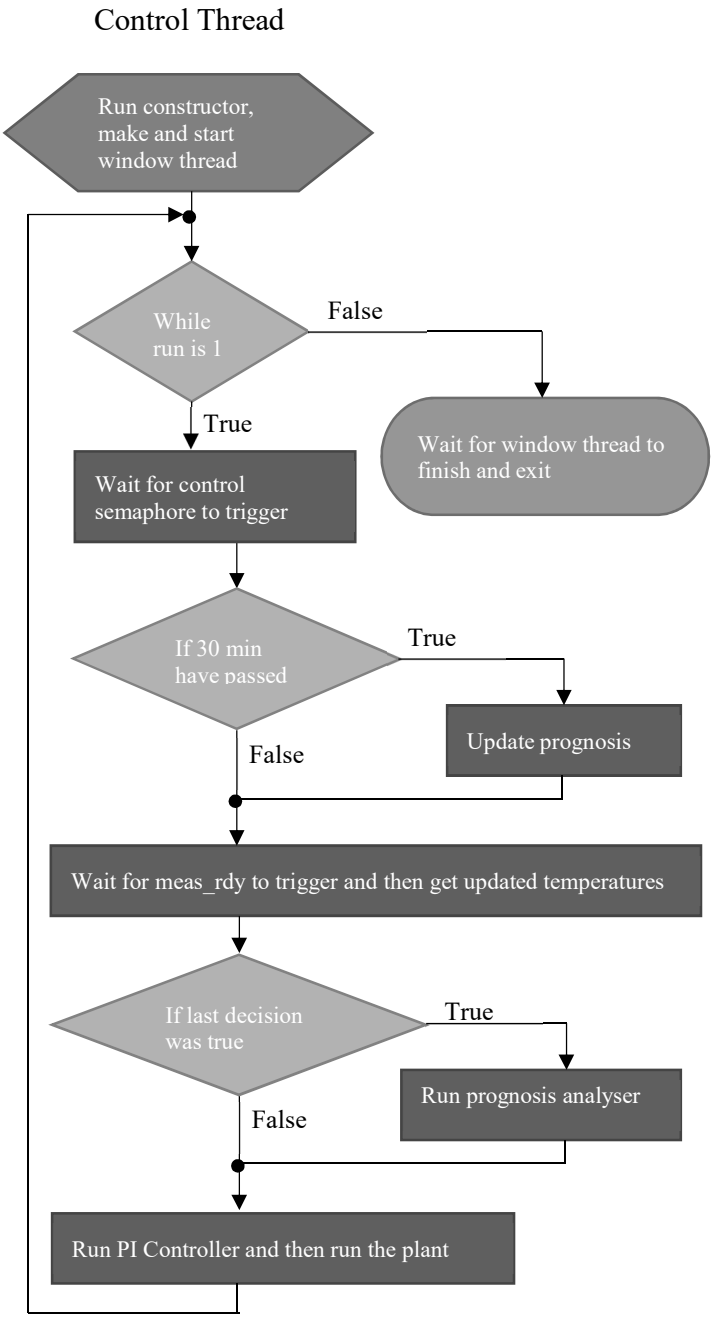

Figure 30 Controller thread of the main program

Integration tests

The integration tests are small scale tests to make sure that the individual modules discussed in the implementation fulfill their purpose. The execution of these tests varies from sample code being run to a complete module being fed dummy or live data.

Downloading and preprocessing

During the creation of the DataDown python script one small scale test was made, followed by a live test of the whole module.

The first test was a separate python script that tested how argument parsing works. This script processed the arguments and printed the contents of arguments that met certain requirements. The argument parser was required to identify and load the path to where the data should be saved, and the URL from which to gather the prognosis datasets.

```
# This program will test the functionality of how to use arguments
import argparse

# Create object to contain arguments
parser = argparse.ArgumentParser(description='Downloads weather prognosis from
yr.no')

# What to look for, nargs is how many arguments after the -- are arguments of
this type
parser.add_argument('-dn', nargs=2, help='Name as which the data should be
saved, can be a path')
parser.add_argument('-wl', nargs=1, help='Address of the webpage to download
data from')

# Prepare args
args = parser.parse_args()

# Work with the args
print(args)
print(args.dn)
```

Figure 31 Code snippet showing the argument parsing experiment for the python script.

```
hans@DX64 ~/Py_arg_parser $ python3 main.py -dn /down/path /down/path2 -wl www.test
Namespace(dn=['/down/path', '/down/path2'], wl=['www.test'])
['/down/path', '/down/path2']
hans@DX64 ~/Py_arg_parser $
```

Figure 32 Terminal output of running the python argument parsing script.

As seen above, the argument parser looks for the predefined keywords '-dn' and '-wl' and saves the number of arguments defined by 'nargs' to a list. In this test two arguments were allowed for the path, this was only to grasp a better understanding of the argument parsing, the final program only accepts one path and one URL.

Besides the argument parsing, the rest of the script only includes formatting of strings and saving these strings to files. The module test was made by passing a directory and url with live data to the script.

```
hans@DX64 ~/DataDown_test $ python3 DataDown.py -dn ./Datasets/progdat -wl https://www.yr.no/place/Denmark/South_Den
mark/S%C3%B8nderborg/forecast.xml
hans@DX64 ~/DataDown_test $
```

Figure 33 Command to be used with the python prognosis downloading script.

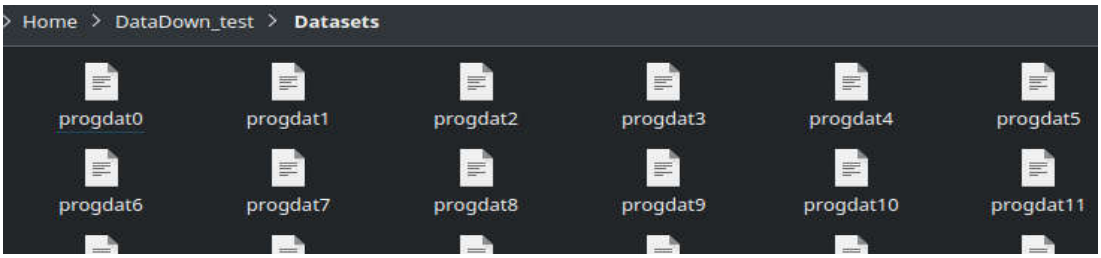

Figure 34 Snippet showing the structure in which the prognosis is saved.

As can be seen from the images above, given a path and URL, the script creates the directory and saves the data into this directory as a file per prognosis iteration, where each iteration has a 6-hour increment. The data contained in each file is saved without any unit descriptions, meaning that one needs to know what number corresponds to what value. This can be found out by either looking at the .xml site that is downloaded, or by enabling the debugging section in the script. For full python script see appendix.

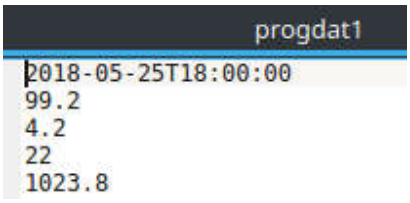

Figure 35 Content of one dataset/file.

Loading and processing prognosis

The loading and analyzing/processing of the prognoses is done by 3 sub-modules working together. The first module passes the command to the python script, downloading the data. The second module first checks whether the directory and files exist, and then loads the existing files into local memory. The third module does the processing itself, in this case it is set to only calculate the mean value. In this test one can see how, after receiving the path and URL from the main function, downloads the datasets. After the datasets have been downloaded there is no guarantee that the python script did not encounter any errors, because of this the prognosis loader starts by checking the file structure, making sure that all files exist. The prognosis downloader receives the path from

```
Running command:
python3 ./DataDown.py -wl https://www.yr.no/place/De
nmark/South_Denmark/S%C3%B8nderborg/forecast.xml -dn
 ./data/dataset1/wdat


Checking Directory...
        './data/dataset1/'      [PASSED]
Checking Files...
        './data/dataset1/wdat0' [PASSED]
        './data/dataset1/wdat1' [PASSED]
        './data/dataset1/wdat2' [PASSED]
        './data/dataset1/wdat3' [PASSED]
        './data/dataset1/wdat4' [PASSED]
        './data/dataset1/wdat5' [PASSED]
        './data/dataset1/wdat6' [PASSED]
        './data/dataset1/wdat7' [PASSED]
        './data/dataset1/wdat8' [PASSED]
        './data/dataset1/wdat9' [PASSED]
        './data/dataset1/wdat10'        [PASSED]
        './data/dataset1/wdat11'        [PASSED]
        './data/dataset1/wdat12'        [PASSED]
        './data/dataset1/wdat13'        [PASSED]
        './data/dataset1/wdat14'        [PASSED]
        './data/dataset1/wdat15'        [PASSED]
        './data/dataset1/wdat16'        [PASSED]
        './data/dataset1/wdat17'        [PASSED]
        './data/dataset1/wdat18'        [PASSED]
        './data/dataset1/wdat19'        [PASSED]
        './data/dataset1/wdat20'        [PASSED]
        './data/dataset1/wdat21'        [PASSED]
        './data/dataset1/wdat22'        [PASSED]
        './data/dataset1/wdat23'        [PASSED]

Data integrity test     [PASSED]


tmp 0 is 23
tmp 1 is 22
tmp 2 is 17
tmp 3 is 14
tmp 4 is 24
tmp 5 is 25
tmp 6 is 17
tmp 7 is 15
tmp 8 is 20
tmp 9 is 22
tmp 10 is 17
tmp 11 is 17
tmp 12 is 22
tmp 13 is 21
tmp 14 is 18
tmp 15 is 19
tmp 16 is 23
tmp 17 is 22
tmp 18 is 18
tmp 19 is 19
tmp 20 is 23
tmp 21 is 21
tmp 22 is 18
tmp 23 is 18
```

Figure 36 Output of the prognosis loader and analyzer.

main the same way the prognosis downloader function that called the python script did. After the data has been loaded and returned to the main function, it is passed to the analyzer which outputs the contents of the data used and gives us the mean value just as hoped. One can clearly see the change in the mean value if one comments out the prognosis downloader function and changes the temperatures within the files manually.

Scenario tests

When all the different modules have combined, they should fulfill the requirements stated in the MoSCoW, together with some more specific requirements unique to the individual tests.

The first test to be made is to check check the prrof of concept, after this it is important to test how EcoDome behaves without any meddling, meaning that all actuators are turned off, and only the sensors are reading and logging the data continuously. This test will be used as a reference for the other tests.

After the first test, the rest of these tests would be:

- How does the system (EcoDome) behave with only the main circulation on?
- How does the system behave using only the stone beds (without modifications)?
- How does the system behave using only the stone beds (with modifications)?
- How does the system behave with all actuators on with no restrictions?

Unless otherwise stated, the analysis process consists of the preparation, the measurement and result, and the analysis, which are executed in the following way:

The preparation consists of the placement of sensors and commenting out parts of the code to restrict the actuators. The code used is the main build, meaning that even if the actuators are disabled, it will still run the controller and prognosis analyzer.

The sensor setup consists of the standard setup, meaning that there is one sensor outside, two sensors inside, one located at soil level (in this case 1 meter above the ground) and one at the window (3 meter above the ground), 2 sensors in the rightmost bed, where one is close to the fan and one is at the far end, and finally one sensor that is located at the intake of the stone bed fan.

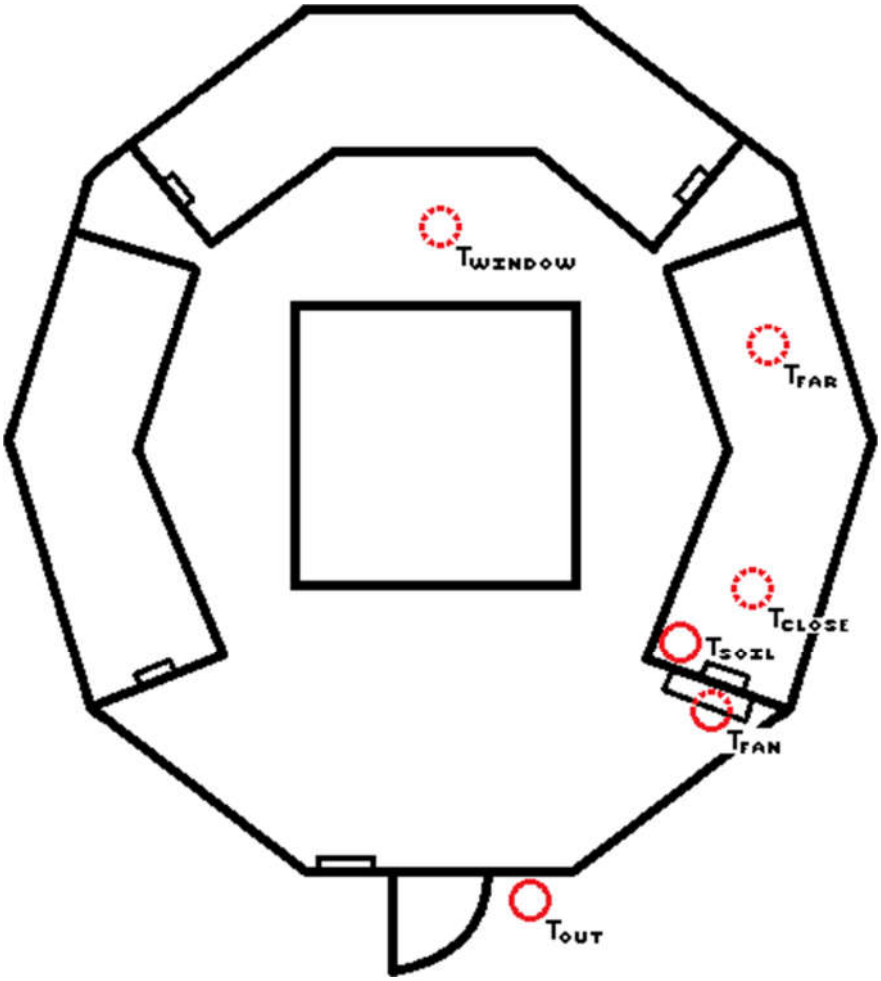

Figure 37: Sensor placement in the EcoDome.

The input for the program is:

- Prognosis number: 3
- Maximum Temperature: 30 degrees Celsius
- Minimum Temperature: 18.5 degrees Celsius
- desired Temperature: 22.5 degrees Celsius

The measurement consists of running the experiment, normally for a duration of 48 hours.

The analysis consists of the extraction of the log file, followed by running it through a MATLAB script. The script starts by calling two functions; the first imports the time and date from the first column as a string array, the second imports the data from the rest of the columns. Then, it creates two subplots with predefined positions and sizes, shown in the same window; the first is a graph of the temperatures the various sensors have measured and the Tref calculated by the prognosis analyzer, second plot shows how the controller output, $u(t)$, changes with the inside temperature, and whether the fans were on and the window open. The latter are just booleans, and, since $u(t)$ usually lies between $+600$ and -600, the data for the stone bed fans is multiplied by $+200$ and the main fan and window by -200 to make it easier to see on the graph when they are on.

```
clc;
clear;
close all;

%% Import data
% Import time as string array
%logdata_time = importTime('C:\Users\frede\Documents\Bachelor\Measurements\28-
05-2018\logdata0.txt');
% Import time as string array
logdata_time = importDatetime('C:\Users\frede\Documents\Bachelor\Measure-
ments\28-05-2018\logdata0.txt');
% Import data as matrix
logdata = importData('C:\Users\frede\Documents\Bachelor\Measurements\28-05-
2018\logdata0.txt');

%% Data analysis starts here
% Number of rows
N = size(logdata_time,1);
% Make time step
Ts = zeros(N,1);
for i = 1:N
    Ts(i,1) = i;
end

hold;
time = datenum(logdata_time, 'yyyy-mm-dd HH:MM:SS');   % Convert data to correct
format
x_axis = time;  % Can be replaced with Ts
pos1 = [0.1 0.4 0.8 0.5];   % Specify position of subplot
subplot('Position',pos1);   % Create subplot from specified position
plot(x_axis,logdata(:,1:6),x_axis,logdata(:,8));    % Plot data
datetick('x','yyyy-mm-dd HH:MM:SS');   % Set x-axis to year-month-day
hours:minutes:seconds
axis([x_axis(1,1) x_axis(N,1) -inf inf]);   %set axes start and stop positions
grid on;
xlabel("Time");
ylabel("Temperature [C]");
legend('Tsoil','Twindow','Tstone fan','Tout','Tstone close','Tstone
far','Tref');

pos2 = [0.1 0.1 0.8 0.2];
subplot('Position',pos2);
plot(x_axis,logdata(:,7),x_axis,logdata(:,9)*200,x_axis,logdata(:,10)*(-200));
datetick('x','yyyy-mm-dd HH:MM:SS');
axis([x_axis(1,1) x_axis(N,1) -inf inf]);
grid on;
legend('U','F_S_b I/O','F_M I/O');
```

Figure 38 The MATLAB script used to analyze the logdata.

Test 01 - Proof of concept

The first test to be made is whether the proof of concept fulfills its role, giving us insight into the difficulty of this project.

Goal:

The goal of the proof of concept was to show that it indeed is possible to make a device that can use weather forecasting to give meaningful output for further use.

Setup:

This setup does not use the default setup. The program containing the prognosis loader and processing class, and the prognosis downloading script are placed alongside each other on a regular Linux computer. The prognosis analyzer is set to calculate the mean value of the first 24 datasets that have been gathered. The main function returns the output from the prognosis analyzer to the terminal and repeats the download to analysis process.

Hypothesis:

Since yr.no presents a prognosis dataset for each 6 hours, starting at 00:00, it can be expected that the temperature values from the prognosis will all move one slot closer to the present, the current value will be discarded and a new will appear furthest away from the present. The mean value calculated by the proof of concept will change.

Results:

From Figure 39 it can be seen, that when the prognosis is updated on yr.no's site, there occur a change in the first 5 datasets, and that the mean value goes from 19.9167 degrees Celsius to 20.1667 degrees Celsius.

Analysis:

Although it was expected that all datasets would move a place further towards the present, this did not happen. Furthermore, the change did not happen at 12:00, but 5 minutes and 40 seconds later. Since this is the way that yr.no seems to push out their prognoses, there is nothing much for us to do than adapt.

Conclusion:

Although the result was unexpected and did not match with the hypothesis, it should be no problem to adapt the program.

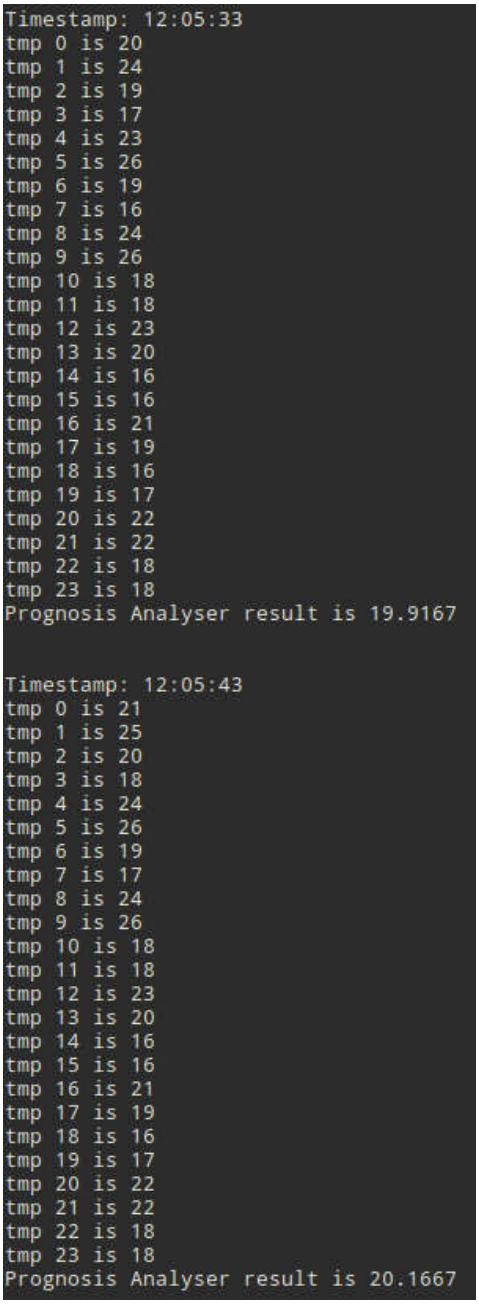

Figure 39 Terminal snippet showing the change from the prognosis, and the coresponding reaction from the program.

Test 02 - Actuators off

This test aims to serve as a point of reference for future use. During this test all actuators will be turned off, meaning that only the sensors will be working. The program will only log all data gathered to the log file for analysis.

Goal:

The goal is to make a curve that can be used in later tests as reference, and to give insight to the current state of the EcoDome.

Setup:

Window is open, all actuators are off. Temperature sensor locations are 1 outside, 1 inside at 1 m above floor (same height as plants), 2 inside one of the stone beds.

Hypothesis:

The outside temperature will have a rather slow rise time, with maximum around noon, thereafter it will start falling again. The inside temperature will roughly look like a square-wave, the temperature will be high as long as the sun shines, and low when it is dark. The stone bed will probably show a sine curve, roughly following the inside temperature with a phase shift of 90 degrees.

Results:

The biggest difference between hypothesis and result are that the outside temperature has a spike at the evening due to the sensor being hit by thermal radiation from the sun. Furthermore, the inside temperature is way more unstable than anticipated with large spikes during the day.

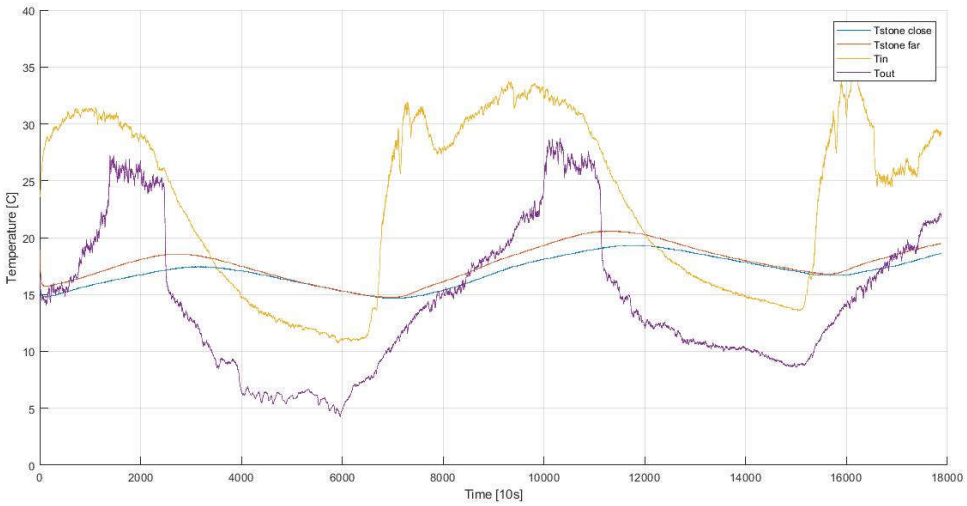

Figure 40: Graph of logged temperatures. Big version in Appendix A: Figure 1

Analysis:

The spike at the evening can be explained by the placement of the sensor, as the sun started shining on it in the late noon. The spikes on the inside temperature can be explained by people opening the door, or because of clouds blocking the sun.

Conclusion:

Besides the outside temperature, the measurements could prove to be rather useful.

Test 03 - Main circulation

This test aims to show that the main fan and window work as expected and have no conflicts with neither the code nor the electronic or mechanic setup.

Goal:

The goal is to find how much of a difference the main fan and window make. Additionally, this test also aims to test the reliability of this part of the system, making sure that window and fan work together, and that the window does not get stuck in any way.

Setup:

The sensors are located at default locations, the stone bed fans have been disabled in the code.

```
void plant(void)
{
    lock_guard <mutex> u_lock(u_mutex);

    // main fan & window
    if((u > 20) && (tm.T_outmean > tm.T_inside))
    {
        digitalWrite(RELAY_1_P1, HIGH);
        window.open();
        mainFAN = true;
    }
    else if((u < -20) || (tm.T_inside > Tmax))
    {
        digitalWrite(RELAY_1_P1, HIGH);
        window.open();
        mainFAN = true;
    }
    else
    {
        digitalWrite(RELAY_1_P1, LOW);
        window.close();
        mainFAN = false;
    }
    // stone bed - commented out
}
```

Figure 41 Code snippet of the plant function, used to control the actuators, note that the stone beds have been commented out.

The observations whether the window and fan work together as they should will be made qualitatively, as the program can only turn the actuators on or off, and it would require additional sensors on the actuators to decide whether they are running or not. Therefore, the observations will be made by a human who looks at the fan and window to make sure they both start and stop at the same time, and that the window opens and closes fully. The time over which the test was run, was 24 hours.

Hypothesis:

The inside temperature will stay roughly around the optimal temperature, following the Tref. During daytime the sun might become more than the actuators can handle, meaning that the fan and window will run continuously. During nighttime the temperature might fall below the optimal temperature, meaning that the fan and window should close to save heat.
The stone bed sensors should simply follow a sine wave since they are not enabled.

Results:

As seen by Figure 42, the main circulation cannot keep the EcoDome cool during day, at least not by itself. While this might be the case, one can also see that during evening hours the temperature is held stable at ~30 degrees Celsius for about 2 hours, and during the morning to the 29th it can be seen that the same happens for 30 minutes.

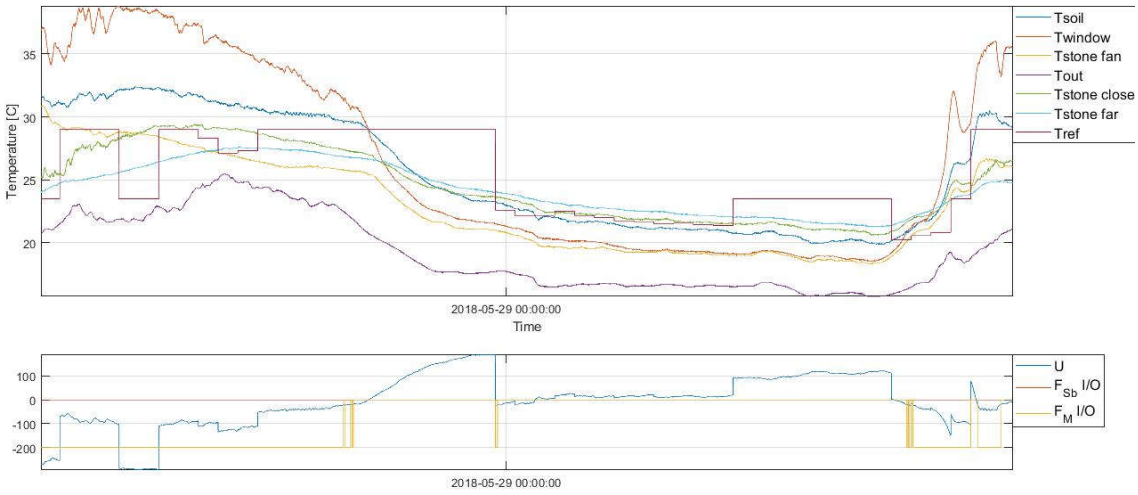

Figure 42 Graph showing the results of running the EcoDome with only main circulation. Big version in Appendix A: Figure 2

Analysis:

Although this might be disappointing, it was to be expected. The main circulation is simply too small compared to the giant heater that is the EcoDome. At least it can clearly be seen that the program is doing a difference and is capable at stabilizing the temperature for a short amount of time during morning and evening.

Conclusion:

The main circulation works, even though the actuators are having a hard time keeping up with the EcoDome's production of heat during sunny days.

Test 04 - Stone bed (without modification)

Like the main circulation test, this test aims to prove that the stone beds work reliably.

Goal:

The goal, here, is to determine how well the stone beds work. To function properly as a core part of the EcoDome control system they must be able to collect heat throughout the day and release it at night to keep the temperature in the EcoDome from diving below a specified threshold.

Setup:

The stone bed fans are all enabled, the main fan and window are set to only open if the inside temperature reaches a critical point (30 degrees Celsius) as not to kill the plants inside.

- The stone beds are circulating the cold air from the ground.
- Measure time is only ~24 hours.

```
void plant(void)
{
    lock_guard <mutex> u_lock(u_mutex);
    // main fan & window
    // Safety
    if(tm.T_inside > 30)
    {
        digitalWrite(RELAY_1_P1, HIGH);
        window.open();
        mainFAN = true;
    }
    else
    {
        digitalWrite(RELAY_1_P1, LOW);
        window.close();
        mainFAN = false;
    }

    // stonebed
    if(((r > Tdes) || (u < -5)) && (tm.T_stonemean < tm.T_inside))
    {
        digitalWrite(L298N_STONE, HIGH);
        stoneFAN = true;
    }
    else if(((r < Tdes) || (u > 5)) && (tm.T_stonemean > tm.T_inside))
    {
        digitalWrite(L298N_STONE, HIGH);
        stoneFAN = true;
    }
    else
    {
        digitalWrite(L298N_STONE, LOW);
        stoneFAN = false;
    }
}
```

Figure 43 Code snippet of the plant function, note that the main cirucaltion has been exchanged with a simple safety against overheating.

Hypothesis:

The difference in the temperatures between ‘stone close’ and ‘stone mid’ in the stone bed will be higher than without the fans being on, furthermore the maximum and minimum temperatures will be slightly more extreme. If the inside temperature rises above 30 degrees Celsius, the main fan and window will open, causing the inside temperature to be more stable. The inside temperature is expected to lie within a certain range, specified by a fixed low and high optimal temperature.

Results:

As the graph shows, the safety that is the main fan and window works exceedingly well, keeping the inside temperature within the desired range. However, the contribution from the stone bed is less than ideal. The temperature never rises much during the day, and as a result the stone bed doesn’t have much heat to release at night.

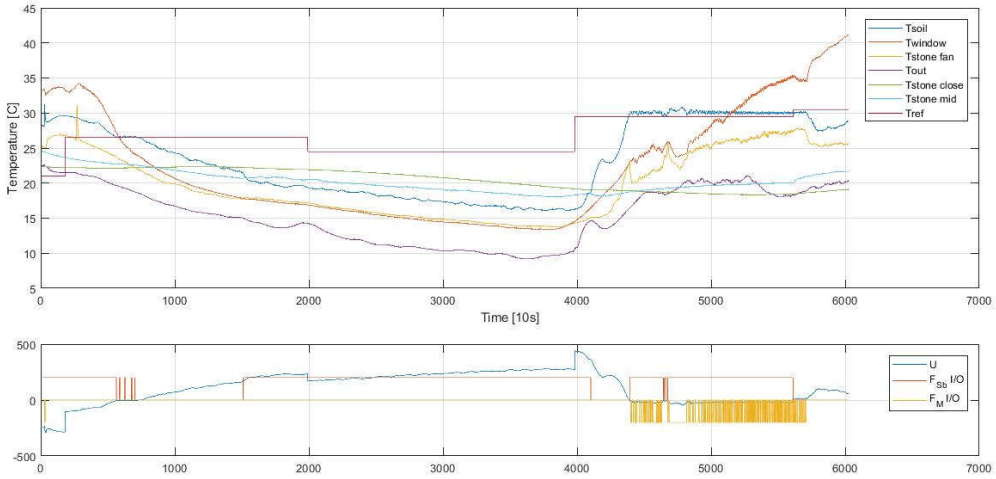

Figure 44: Graph showing performance of stone beds. Big version in Appendix A: Figure 3

Analysis:

The results of this test are rather disappointing, the stone bed is meant to work together with the prognosis analyzer and form the core of the project. Since the stone bed does not show satisfying results, it is necessary to somehow enhance it, even if it is only slightly.

The reason behind the stone bed showing so underwhelming results, might be because it circulates the cold air from the bottom of the greenhouse, instead of sucking it in from the top where there is a greater potential for heat transfer of any significance due to the warmer air.

Conclusion:

This test showed that the stone beds are faulty in their current state, and that it is important to make the changes that the stone beds produce more extreme. The stone beds do not need to work as seamlessly as an actually working stone bed, but they must show that the prototype indeed does affect them.

Test 05 - Stone bed (with modification)

Since the failure of the first stone bed test, this test aims to prove that the improvements to the stone beds make enough of a difference to let the project continue without further problems in this area.

Goal:

The goal for this test is to show that the improvements made to the stone beds indeed maximize its effectiveness to a useable level. This means that the stone bed temperatures should

not show a sinus curve like in test 04 but be influenced by the current temperature at the top layers as well as the status of the fans.

Setup:

Same as the ‘test 04’ but with the introduction of flex tubes to suck the air from the top of the EcoDome down to the stone beds. Also, the safety that is the window and main fan were changed so that the window opens at 30 degrees Celsius, and the fan starts at 32 degrees Celsius.

Hypothesis:

The temperature sucked in from the flex tubes should be much higher than the air from the bottom. The differences in temperature regarding sensor placement and maximum/minimum values should also be more extreme. Additionally, the mean temperature in the stone beds should slowly rise as days pass by (this might not be clear from the measurements).

Results:

As one can see from the graph, the stone beds behave largely different now. Although the temperature at the far end of the stone bed is rather smooth, it does not resemble a sinus curve, meaning that the stone bed reacts to the temperature more aggressively. When looking at the temperature close to the fan, one can even see some instabilities. Looking at the difference between the inside and outside temperature, it is clear that after soaking in the heat during the 25th the stone bed is actually capable of keeping the inside temperature 3 degrees Celsius hotter compared to the outside temperature during the night to the 26th than the night before.

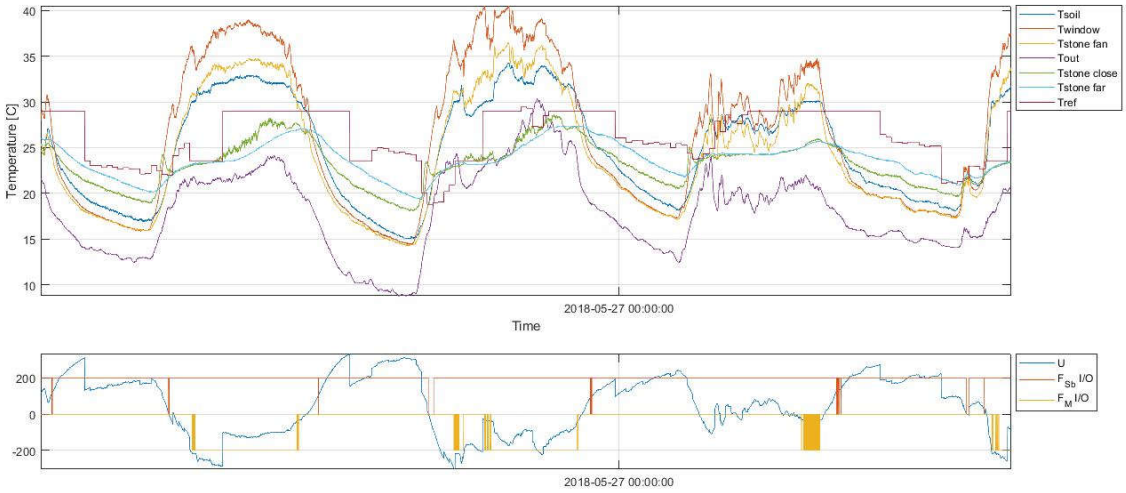

Figure 45: Graph showing improvement in stone beds. Big version in Appendix A: Figure 4

Analysis:

Installing flex tubes seems to have improved the performance of the stone beds, although not by much. It can keep the temperatures about 3 degrees Celsius higher during night, and should have the opposite effect during the day, but this is still not sufficient to keep the EcoDome inside the maximum temperatures specified.

Conclusion:

These results satisfy the requirements for the stone beds to be useful to the project, allowing the prognosis analyzer to affect the temperature of the EcoDome.

Test 06 - All actuators on

This is the final test to make sure that everything that was tested individually before acts nicely together, complimenting each other and keeping the environment inside the EcoDome at acceptable temperatures, defined as the maximum, minimum and desired temperatures.

Goal:

The goal is to understand how the EcoDome behaves when everything is running, and to make sure that everything works when combined. The test also aims to give an understanding of how effective the program is at controlling an enclosed environment.

Setup:

This setup uses the default setup.

Hypothesis:

During night the inside temperature of the EcoDome should be clearly higher than the outside temperature. During day the inside temperature should vary depending on Tref, but should never exceed the maximum temperature, as long as the actuators are capable of handling the load.

Tref should be high during daytime to save the heat for night, and low during night time to release the heat so it can cool the environment during daytime. Tref should never exceed the maximum or minimum temperature specified in the configuration file.

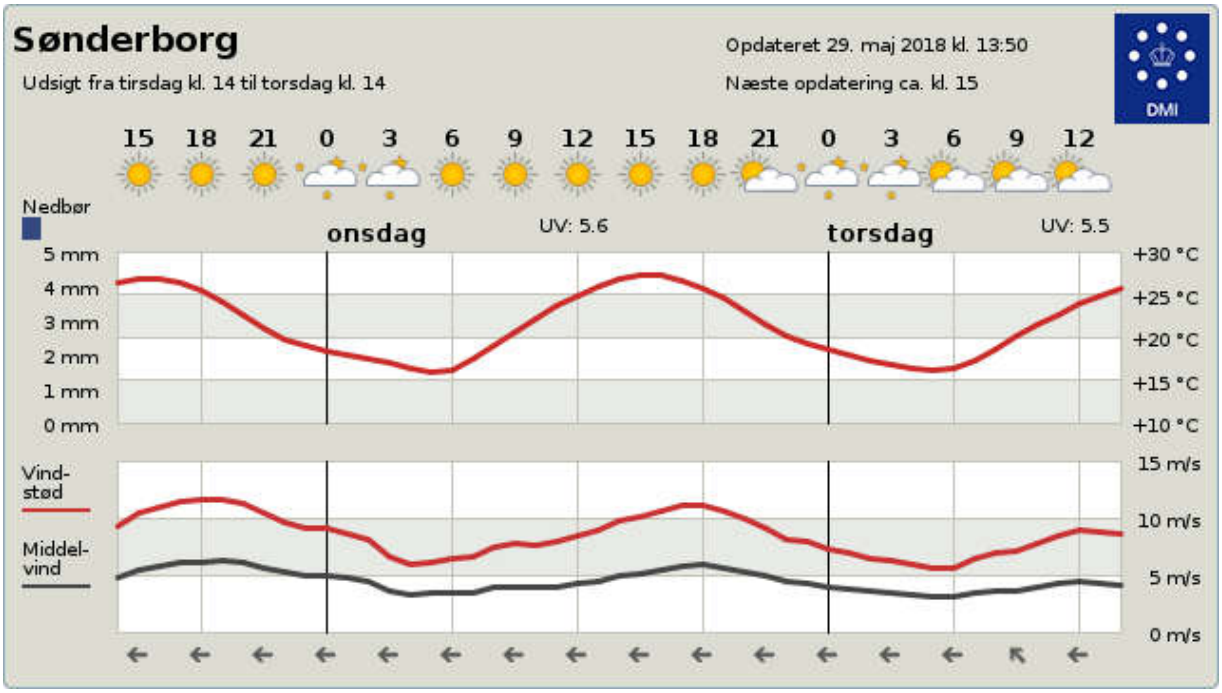

Figure 46: DMI weather forecast for reference.

It is also expected that Tout resembles the graph proposed by DMI for the results to be deemed reliable.

Results:

As seen on the graph the inside of the EcoDome is $5\ °C$ warmer than outside during the night and warmer still during the day. looking 6-12 hours into the future Tref increases at day to prepare for the night. Tout also follows the graph presented by DMI to same degree.

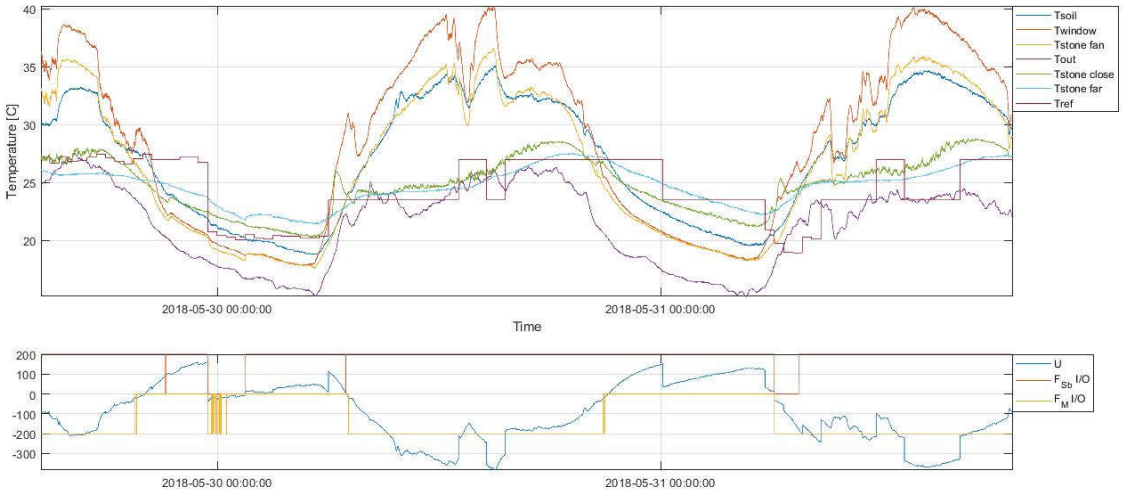

Figure 47: All actuators on, Tout follows DMI. Big version in Appendix A: Figure 7

Analysis:

As Figure 46 and Figure 47 show Wednesday the 30th was very warm day. The inside temperature goes above the specified maximum and stays somewhere close the temperature of the air being drawn into the stone beds which, when comparing to the 29th and 31st, is higher than usual, but the control system still tries to regulate this according to Tref.

Conclusion:

The control system turns on the various actuators as intended, but the ones currently installed are not powerful enough to influence the inside temperature once it goes much above $30\ °C$.

Discussion of results

As test 04 shows, the stone beds have very little influence on how the temperature changes inside the EcoDome. The temperature in the stone beds only changes by a few degrees, and when air is flowing through them they are again only a few degrees warmer than with no airflow. While active they rarely reach 23 ^{o}C, and during the day they have even been noted to be colder than the air flowing into them. This last part could be because of the sun hitting the sensor by the fan during the day, as it does read lower temperatures than the ones in the stone bed at night, where the temperature drops down to around 18 ^{o}C just before dawn.

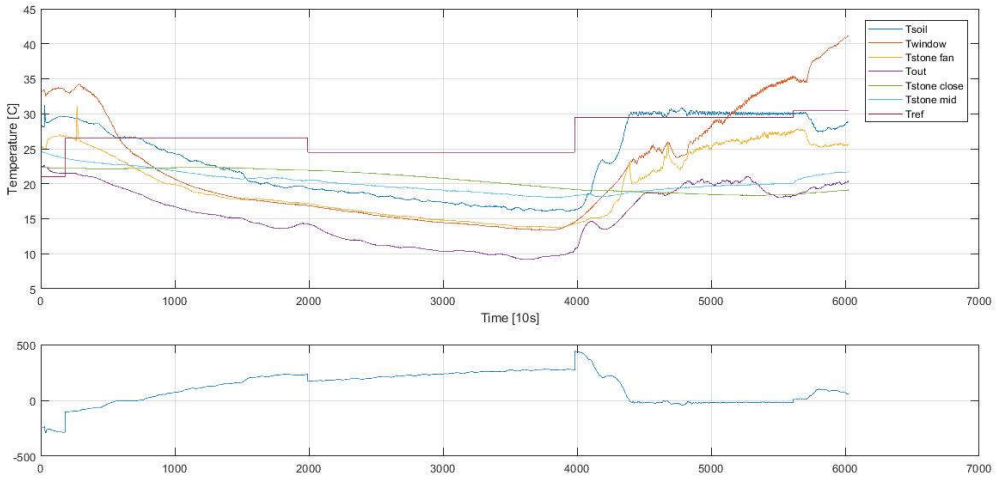

Figure 48: Stone bed temperature dropping to 18 °C *over night. Big version in Appendix A: Figure 3*

But it is also reasonable to assume that, as the air is blown straight into a wooden board only a few centimeters past the fan with just a small gap that leads into the stone bed itself, the airflow through the stone beds simply isn't sufficient to regulate their temperature in any meaningful way. In addition to the limited airflow the fans draw in the air from down near the floor where the temperature is usually only some 20 odd degrees. This relatively cold air flows

through the stone beds and comes out at the top where it mixes with the air surrounding the stone beds, which is also some 20 odd degrees, before it gets drawn back into the stone beds.

In an attempt to compensate for the low temperature and airflow it was decided upon to install flex tubes that would draw air from the top of the EcoDome, which is usually $10 - 15\ ^{o}C$ warmer than down by the fans. This has helped to raise the temperature of the stone beds by about $5\ ^{o}C$ during the day, while at night the temperature still drops to about $18\ ^{o}C$. This means that installing the flex tubes has increased the temperature difference from $5\ ^{o}C$ to almost $10\ ^{o}C$, practically doubling the heat storage.

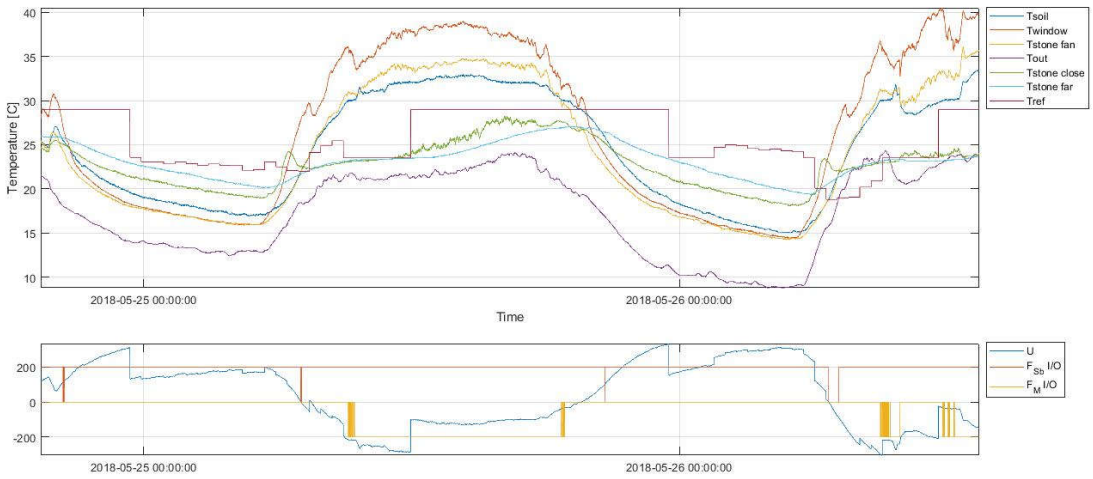

Figure 49: Graph showing how stone bed temperature has increased after installing flex tubes. Big version in Appendix A: Figure 5

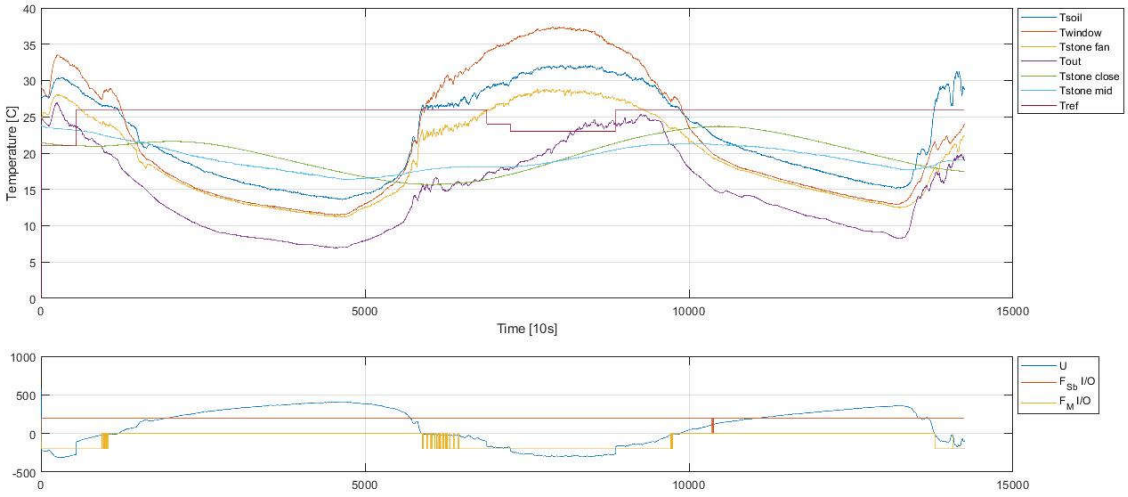

Figure 50 Regulating inside temperature. Big version in Appendix A: Figure 6

As can be seen on the above graph, the blue curve representing Tsoil at one point hits Tref and stays there for a while before the temperature starts rising again. This is a clear indication that the control system works as intended, but the current hardware in the EcoDome, here the large fan and the window, is not enough to keep the temperature down in the long run.

The strategy for controlling the EcoDome is very simple; the idea is to regulate the inside temperature through ventilation, and to extend the time for which it lies within a desired range by using stone beds. Due to the slow nature of temperature changes this can be done by keeping the fans either on or off, without using a PWM signal, making it possible to control them just by comparing the inside temperature to a specified threshold. This also means that it might not have been necessary to implement the PI-controller as the error between the reference temperature and the inside temperature would still be enough to tell how the latter relates to the threshold. The PI-controller was implemented anyway, but mainly for the learning outcome.

Although the controller program can control the stone beds it currently lacks the functionality to calculate the amount of heat that has been stored. This part has been intentionally

left out since the current state of the controller has no need for it. It would be possible to implement it in the future if the need arises, but while the fans for the stone beds are either on or off based on a predefined threshold that is regulated by external variables it is not necessary as it isn't essential to the core prototype.

Considerations

If this project was to be made again under the same prerequisites, it could be considered to finish the maintenance of the EcoDome quickly and start some small-scale tests to check the capability of the stone beds and other hardware. This could result in some of the faults discovered during the last few weeks of the project to be discovered and solved in the beginning, resulting in a more mature environment for the control system once the project reaches the second phase. It would also be worth considering to take control over the budget for the EcoDome, making the project have a defined budget instead of a 'ask and hope' mentality.

In the case of the project being remade, it would also be of value to make changes mainly concerning physical aspects of the EcoDome.

The first consideration, which would probably have the heaviest impact on the EcoDome's efficiency, is to have the stone beds fully covered, and then blow the air directly into the stones and out the other end. This would decrease the surface area of the air flow which would increase its flow velocity. This means that the temperature barrier that occurs when air of a different temperature is blown into the stone bed would move along faster, thus decreasing the time over which the heat transfer would take place. Perhaps, with a faster heat transfer, the stone beds would have time to store more heat during the day and release more heat during the night, which would increase the overall heat transfer.

Another way to increase the airflow would be by increasing the surface area of the air intake. This could be done by installing a larger fan, or by installing multiple smaller fans that would increase the total surface area.

It was at one point considered if a traditional rectangular greenhouse would be a better option than the current shape of the EcoDome, and there are positive and negative points that can be made about both.

Looking at the EcoDome, it seems that the focus was laid on the aesthetics. It might be better at circulating the air than a rectangular greenhouse, but perhaps that is not necessary. In the case of the latter, it would probably work just as well to draw in air from either side and have it mix in the middle, and it might even be enough to only draw in air from one side, depending on the size of the greenhouse. And, though the dome-shape does have an advantage when it comes to letting in sunlight, it also seems overly complicated. A design that dates all the way back to the 1970's is the solar greenhouse. This rectangular greenhouse only has windows along one side that is facing south, and this, combined with the building's thermal mass, seems to provide plenty of heat to grow plants year-round, and even in freezing conditions. So, although a dome-shaped greenhouse does have some positive aspects, like always having some part that is perpendicular to the sun, which helps to ensure that the maximum amount of heat enters the greenhouse, there are also some drawbacks. In the case of the maximum amount of heat going into the EcoDome, this could also result in the temperature inside the greenhouse rising to a point where the equipment that regulates it no longer can keep up. And while the wind speed and direction were not taken into consideration they still affected the temperature, the shape of the greenhouse surely plays a part in how these are connected. A dome will always guide the wind around it, which will help cool it down, though, as with the case of the sunlight, there is always a

small part that will be perpendicular to the wind direction, but since the EcoDome is sloped this wind will for the most part be directed upwards and over the greenhouse. For a traditional rectangular greenhouse, the wind will on most occasions be directed up or down along the side as it hits the dome at an angle. Depending on this angle the force that the greenhouse would have to withstand would vary, and on some occasions, it would be subjected to the full force of the wind, either from the sides of from the gables. The angle of the wind on the greenhouse would also affect how much heat would be left to transfer through the walls of the greenhouse, even for a constant wind speed.

But the shape of the greenhouse isn't just about how it reacts to the forces of nature. It also influences the interior. The EcoDome has an area of $28\ m^2$, and the planting boxes, now including the tanks in the center, have an area of almost $16\ m^2$. A rectangular greenhouse of the same size could then have the dimensions $4\ m\ x\ 7\ m$. With a walkway down the middle, $60\ cm$ wide, and leaving a meter after the entrance open for storing various tools, one could fit a planting box either side with a total area of $20.4\ m^2$, and if one would prefer to have planting boxes the entire length of the greenhouse they would cover an area of $23.8\ m^2$.

So, all in all, the shape of the greenhouse just depends on the individuals preferences; does one want to allow as much sunlight in as possible, is it of interest to be able to store the heat for a long time, or is it of interest to optimize the space inside the greenhouse?

Building a greenhouse from scratch, after having chosen a shape, would also allow one to plan where to put the different sensors, how to protect them from thermal radiation, and how to draw the cables that connect them to the control system.

Course of events

Because of this project being made for a leisure association, the requirements and support were mostly those of hobbyists. This was both good and bad, as the requirements became rather broad, resulting in the project being able to choose its path along the way, but also requiring more work and explanations. Regarding money the project did not have a predefined budget, with just the instructions to keep it as low as possible as long as it works. The components bought roughly included the Raspberry pi, the flex hose, some prototype boards with components, 2 relays, and 2 temperature sensors, as the rest was reused.

Because of the EcoDome's initial state, quite some time was spent on preparing it for this project. This included the maintenance of the control box and actuators, as well as the change of sensors.

Halfway through the project VedvarendeEnergi-Sønderborg had a general assembly where the future structure of the association was discussed, here the board of directors was also changed. In the end the general assembly did not show any influence on the project, but in this period of uncertainty it was speculated upon if the speculations in the risk assessment regarding hazards toward the project would take effect.

Later in the project when time started to become sparse, some problems of varying sizes started popping up. Of these problems, the realization that the stone beds had no effect was probably the worst. Luckily it was possible to partly fix this problem by using flex hoses.

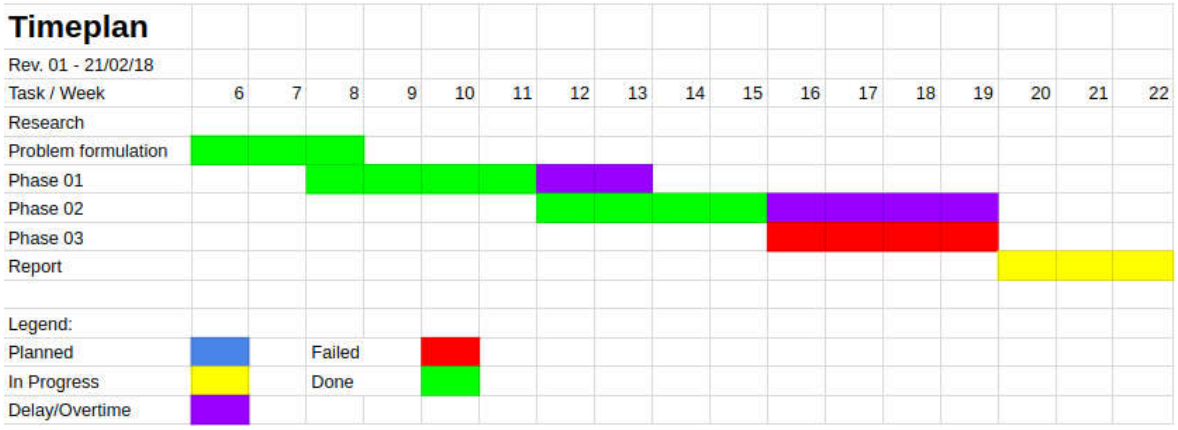

Figure 51: Time table at end of project.

Regarding the time plan, there were some delays in both the proof of concept and the core prototype. Although the third phase was meant as a buffer to allow delays in the core prototype, the delays in the proof of concept were unexpected.

Future work

The primary focus for future improvements would most likely be to incorporate the air pressure as well as the wind speed and direction, gotten from the weather forecast, into the control system, since these all affect how the temperature changes inside the EcoDome. Something to consider here would be to use the wind-chill factor to calculate the temperature outside the EcoDome to be included in the predictive control when regulating Tref.

Other improvements for the future include adding a fan on the other side of the door to double the air intake, as well as installing wooden boards by these fans to either close the opening when they are off and the window is closed, or to be put at an angle to guide the air along the walls of the EcoDome to better cool down the plants. It would also be a possibility to add a fan at the window to help draw out the warm air at the top of the EcoDome.

The final wish for the EcoDome is to connect everything back up that was decided to be left out in the beginning of the project, such as the perl hoses and the humidity sensors. These would of cause then need to be integrated into the control system.

Conclusion

The EcoDome control system is capable of regulating the temperature inside the greenhouse using a weather prognosis to do so and controlling a passive heat storage. Additionally, all its components are easily accessible, thus meeting all the overall requirements for this project.

Regarding the MoSCoW, the must haves are all completed, except for one small part; the program does not calculate how much heat is stored in the stone beds, but everything is made ready for this to be implemented if needed. Parts of should have and could have are also done, such as the PI-controller, the beginnings of an airflow control system, and design ideas and considerations for various heat storage systems, as well as recommendations on how to improve the one that is currently installed in the EcoDome. The won't have has also partly been met by Green Makerlab. The control system itself doesn't require an internet connection, but the predictive control still does.

Although the stone beds are able to increase the inside temperature at night, there is still room for improvements. These improvements include, but are not limited to, changing their current layout by making them fully incapsulated and blowing the air directly into them instead of via pipes that run along the bottom, and increasing the surface area of the air intake.

The control strategy is not very advanced, but the controller does its job, even though it is slightly more complicated than it needs to be because of the PI-elements.

References

EngineeringToolBox. (2003). *Convective Heat Transfer*. Retrieved from EngineeringToolBox: https://www.engineeringtoolbox.com/convective-heat-transfer-d_430.html

EngineeringToolBox. (2003). *Plastics - Thermal Conductivity Coefficients*. Retrieved from EngineeringToolBox: https://www.engineeringtoolbox.com/thermal-conductivity-plastics-d_1786.html

EngineeringToolBox. (2003). *Specific Heat of Solids*. Retrieved from EngineeringToolBox: https://www.engineeringtoolbox.com/specific-heat-solids-d_154.html

Gjødsbøl, F. S., & Vanselow-Rasmussen, H. (2018, May 31). *EcoDome - Intelligent greenhouse control system*. Retrieved from Github: https://github.com/Decclo/EcoDome---Intelligent-Greenhouse-Control-System

GreenhouseKits. (2012, December 26). *Greenhouse heat sinks*. Retrieved from YouTube: https://www.youtube.com/watch?v=rUj-5k9YqGQ

Henderson, G. (n.d.). Retrieved from wiringpi: http://wiringpi.com/

Hyperrealm. (2014). *libconfig*. Retrieved from github.com: https://github.com/hyperrealm/libconfig

LeBoeuf, J. (2016, January 4). *The Effect of Extreme Temperatures on the Tomato and Pepper Crop*. Retrieved from omafra: http://www.omafra.gov.on.ca/english/crops/facts/info_tomtemp.htm

McGhie, S. (2016, November 25). *Nyt energilager skal opsamle grøn energi i varme sten | Ingeniøren.* Retrieved from Ingeniøren: https://ing.dk/artikel/nyt-energilager-skal-opsamle-groen-energi-varme-sten-189135

Rottscholl, O. (2016). *The Ecodome - A connected greenhouse.* Sønderborg: Green Makerspace.

Appendix

The appendix includes everything that was deemed too spacious for the report. The appendix is split into multiple appendices, each having a theme.

The appendix A contains larger versions of graphs and other figures that were too small to efficiently read in the report.

The appendix B contains the schematics and information about the hardware setup.

The appendix C includes the code for the proof of concept, and appendix D the code for the prototype.