Institute of Architecture of Application Systems

University of Stuttgart
Universitätsstraße 38
D–70569 Stuttgart

Bachelorarbeit

Smart Home Gardening – Using modern AI-Techniques to optimize Water Consumption

Jannis Nicolai Buenaventura Westermann

Course of Study:	Softwaretechnik
Examiner:	Prof. Dr. Aiello
Supervisor:	Prof. Dr. Aiello
Commenced:	February, 2021
Completed:	August 2, 2021

Abstract

IoT and Smart Homes get more and more popular these days, thereby the desire to automize more areas of daily life, grows stronger.

The aim of this thesis is to create an autonomous smart home garden assistant that supports the user in the daily garden care and during longer time periods, like holidays. The garden assistant assumes the following tasks: Watering, (soil) heating, artificial lighting and hazardous precipitation protection by a shutter. Additionally, a new AI-based irrigation algorithm is developed to reduce the water consumption. This intelligent irrigation system uses an Artificial Neuronal Network-based evaporation prediction and the information of weather forecasts to generate optimal irrigation plans. Classically, equations with many specific parameters have to be used for an accurate evaporation prediction. Hence, the goal of the usage of ANNs is to simplify the evaporation prediction for the individual conditions in each garden, as the system is self-adapting.

A fully functional prototype of the garden assistant is developed and evaluated during a long-term experiment (end April to mid of July 2021), to prove the advantages of the system.

In this experiment, the intelligent irrigation system saves approx. 25% water and has a gain in growth, compared to the simpler irrigation method. The prototype showed the ability to use and prefer rain, instead of watering. The standard error of the evaporation prediction of this method was approx. 2 percentage points of soil moisture level per day, this is $0.64 mm/m^2 day^{-1}$.

A gain in growth, according to the other assumed tasks, could not be measured in this experiment and has to be evaluated in a greater scaled experiment.

The prototype is based on the previous 'Fachstudie' "Garden Planner for a Greenhouse – Autonomous Garden Assistant for Smart Homes" by Michael Hersam, Philipp Wonner and Jannis Westermann.

Kurzfassung

Internet der Dinge und Smart Homes werden immer populärer und gefragter, dabei wird der Wunsch nach Automatisierung im Bereich des täglichen Lebens immer größer.

Ziel dieser Abschlussarbeit ist die Entwicklung eines autonom agierenden Garten Assistenten für das Smart Home. Das System soll die Benutzerin / den Benutzer bei der täglichen Gartenarbeit unterstützen und ihr/ihm die Freiheit ermöglichen den Garten guten Gewissens, auch längere Zeit, in die Obhut des Garten Assistenten zu geben.

Dabei übernimmt das System folgende Aufgaben: Bewässerung der Pflanzen, Schutz gegen Frost, durch Heizen (des Bodens), künstliche Beleuchtung und Schutz vor gefährlichem Hagel oder Starkregen durch eine verschließbare Markise.
Die Bewässerung wird von einem intelligenten Algorithmus übernommen, der ein Künstliches Neuronales Netz für die Wasserverdunstungsvorhersage nutz und Wetterberichtsdaten einbindet, um selbständig einen optimalen Bewässerungsplan zu erstellen. Klassischerweise werden für die Berechnung der Wasserverdunstung komplexe Gleichungen verwendet, welche viele individuelle

Parameter des Gartens / der Landfläche, für ein genaues Ergebnis benötigen. Um diesen Prozess für die Benutzerin oder den Benutzer zu vereinfachen, wird das KNN auf die individuellen Bedingungen des Gartens trainiert, wobei es sich selbstständig anpasst.

Im Zuge der Abschlussarbeit wurde ein funktionierender Prototyp entwickelt und in einem Langzeitexperiment analysiert und bewertet. Das Experiment lief von Ende April bis Mitte Juli 2021.

Das Experiment zeigte, dass die intelligente Bewässerung ca. 25% Wasser, im Vergleich zu dem einfachen System, sparen konnte. Der Standartfehler der Verdunstungsvorhersage betrug ca. 2 Prozent Punkte der Feuchtigkeit innerhalb von 24 Stunden, das sind etwa $0.64 mm/m^2 Tag^{-1}$. Zusätzlich konnte ein besseres Wachstum der Pflanzen beobachtet werden.

Die zusätzlichen Funktionen des Garten Assistenten zeigten in diesem Experiment keinen Effekt auf das Wachstum der Pflanzen, speziell im Vergleich mit der Kontrollgruppe. Weitere Experimente im größeren Maßstab müssten für eine sichere Aussage gemacht werden.

Als Grundlage für den Prototyp dient die Facharbeit “Garden Planner for a Greenhouse – Autonomous Garden Assistant for Smart Homes” von Michael Hersam, Philipp Wonner and Jannis Westermann.

Contents

List of Figures

List of Tables

List of Listings

List of Algorithms

Acronyms

ASHGA Autonomous Smart Home Garden Assistant. 17

MHL Moisture Hard Limit (Hard Goal). 49

MSE Mean Squared Error. 7

MSL Moisture Soft Limit (Soft Goal). 49

MSLF Moisture Score Loss Function. 49

pp Percentage Points. 27

SE Standard Error. 7

1 Introduction

How can everybody save water, without any personal restriction? This thesis shows, how modern AI-Techniques can automize every garden and reduce water consumption, by ideally using the natural environment.

1.1 Motivation

The motivation was to create an Autonomous Smart Home Garden Assistant (ASHGA) that supports the user at the garden maintenance. For example, the system should give the freedom to go in vacations, knowing the garden is kept well. At the same time, there is a focus on ideally use the natural environment and just regulate, when needed. This way, the system should reduce the Ecological Footprint. In addition, the ASHGA is developed to be easily integrated into an existing Smart Home Solution, the user may has in action yet.

This all should be done in such a lightweight way, a single Raspberry PI is able to run all components of the system, the Smart Home Solution, the sensor and devices controlling instance and the intelligent irrigation algorithm.

Another important point is to get independent from cloud systems, as used by systems on the market, and give the user full control of the system, while keeping it simple to set up and use. Additionally, systems on the market that use weather forecasts to adapt the irrigation, are just an up-coming development and not fully established (see chap. 2 State of the Art).

1.2 Previous Work and Issues

In the summer semester of 2020, as part of our 'Fachstudie' "Garden Planner for a Greenhouse – Autonomous Garden Assistant for Smart Homes", we, that is Michael Hersam, Philipp Wonner and me, Jannis Westermann, created a prototype of the ASHGA. The system could water and light the plants, protect them against frost through a heating system. A built-in shutter shields against hail, snow and flooding as well as burns from solar radiation.

The hardware and mechanics were completed and also a big part of the software. However, the system was just controlled by the simple Fallback System, providing a "Water at Moisture X"-System, as the integration of a weather forecast and moisture prediction was not completed. To calculate the

evaporation and to check if watering is needed, we used a modified version of the Penman–Monteith equation and used a Constraint Satisfaction Problem Solver[1], OptaPlanner[2], to receive optimized watering plans, where the system automatically can decide, if watering is needed.

But we had problems with that planner, OptaPlanner had high calculation times, if we did not use Monte Carlo algorithms to optimize the calculation time. But such algorithms bring the disadvantage, not beeing complete and this way "a randomized algorithm [...] may produce incorrect results, but with bounded error probability"[Tem08].

Also, the evaporation prediction, using the mentioned equation, did not fit to the measured data, either because we had to do many assumptions, and/or such equations are more useful in bigger scales than raised beds and garden parts.

As the engineering of the prototype was very time consuming, we did not have the time to do an evaluation of resulting the prototype.

1.3 Goal of the Bachelor Thesis

With the mentioned issues above, the goal of the thesis is to improve and evaluate the ASHGA prototype. The thesis evaluates the prototype in terms of proving, if there is a gain in growth using a system that can artificially light the plants, heat the soil, water in a smart way and protect the plants with a shutter.
This is done by an experiment, comparing the use of all abilities of the system with a "Water at Moisture X"-System (implemented in the Fallback System) and a manual cared raised bed as control group (see chap. 6 Experiment: Design, Evaluation and Conclusion).

To improve the ASHGA, a better solution for the evaporation prediction and planning algorithm has to be developed.

Summarizing the goals:

- Finish the ASHGA: Improve the system, fix hardware and software issues, implement an intelligent irrigation system
- Compare the system to other systems on the market.
- Evaluate the system, especially the watering system: Run a long-term test and evaluate the results
- Evaluate if there is a gain in growth, when working with all abilities of the system: Do an experiment with three garden parts:
  - Garden Part 1: Full System, lights, shutter, heating plates, AI-Planner for watering
  - Garden Part 2: Only watering, using the Fallback System, less intelligent watering
  - Garden Part 3: Manually cared garden

[1] More information about CSPs, see "Automated Planning – Theory and Practice" [GNT04]
[2] "Eve Aqara - Scripting Automatic Weather Delay" by ModernDayExtras https://www.optaplanner.org/ [Accessed on 2021-07-30]

1.4 Structure of the Thesis

Chapter 2 State of the Art provides an overview of existing smart irrigation systems and other smart home gardening projects on the market and comparing their functionality with the ASHGA and gives an insight of intelligent use of weather forecasts, besides smart home gardening. Also, other studies about evaporation prediction are discussed.

Chapter 3 Overview and short Description of the System gives a short insight into the components of the ASHGA and the communication between them.

Chapter 4 ANN-Based Evaporation Prediction describes the functionality of the evaporation prediction algorithm. It provides short background information about ANNs in general, for a better understanding of the ANN-based algorithm.

Chapter 5 AI-Planner describes and evaluates the implemented intelligent irrigation planning algorithm and shutter system.

Chapter 6 Experiment: Design, Evaluation and Conclusion describes the long-term experiment, the experimental set up and discusses the results.

Chapter 7 Costs and Reliability lists the costs of the ASHGA and the experiment and discusses the evaluation of the reliability of the ASHGA.

Chapter 8 Outlook summarizes the evaluation of the single sub-systems and discusses further possible areas of application, the ASHGA can be used, especially the ANN-based evaporation prediction / irrigation algorithm.

2 State of the Art

To compare the ASHGA and its benefits to current systems on the market, this chapter gives a short overview of the actual developments and how they differ from this project. Additionally, there is a short outlook of intelligent systems besides of smart home gardening and currently used evapotransipration prediction methods.

2.1 Irrigation Systems for Smart Homes

There are different kinds of irrigation systems, especially for smart homes. They can be divided into three categories:

- Simple Irrigation Plans: The user can set days and hours of a week/month and set the irrigation time, or if it is calibrated with the water flow, the amount in liters. In summary, these are two connected timers, where the first is activated when it is to irrigate, the second determines the amount of water. There are many similar solutions on the market, they are cheaper and often have an interface for smart home solutions.
- Moisture Level Controlled: In addition, compared to the plan based watering systems, a moisture sensor is needed. This brings the benefit to water as needed. Many systems also use the moisture sensor to stop the irrigation, if a specific amount of soil moisture is detected. Such systems are often included in plan based systems, making them smarter. The additional sensor brings extra costs. As the simple irrigation plan systems, they get more and more displaced from the market or get updated in the newer versions to the third kind of irrigation systems.
- Weather Forecast depending Plans: These are the newest developments on the market. Specially in the smart home environment, they have only been established since 2-3 years but are dominating now. They have in common, to use weather forecast data and, if possible, data of a moisture sensor, to modify plans, the user has given. So, the watering amount is automatically adapted or irrigation is completely stopped, if rain is expected. In addition, they can water more on hot and dry days. They use commonly simple decision trees or IF-THEN statements to modify the plans. Currently, there are some companies, using such systems. In the next section a few of them will be introduced.

2.1.1 CloudRain

CloudRain started in 2018 as a Kickstarter Project and is settled in Germany. Their product falls into the third category of irrigation systems. They use six parameters to variate watering plan of the user. These are:

Precipitation intensity, precipitation probability, air temperature and humidity, wind speed and cloud coverage. In addition, they use some information from the past, like the last watering amount.

The data are received from their cloud, also the calculation of the modified plan is cloud-based. They do not use a soil moisture sensor, this will make the system more inaccurate than others, because the actual state and expected state can deviate due to the missing feedback loop. However, it is a stand-alone system, easy to install and fail-safe from sensor failure. For this concept, they use a solar panel to power the valve device.

For further information, see their homepage[1] and the Kickstarter Campaign[2].

2.1.2 Eve Aqua

Eve Aqua is an irrigation solution for Apple's smart home solution[3]. As all of the actually intelligent solutions, it works as the third kind of irrigation systems by modifying the plans. The company was founded in 2018.

This product is able to use information from moisture sensors and adapts the watering by IF-THEN statements. Poorly, I only found a german homepage SmartApfel[4] and a english youtube video[5] that describes this functionality. It is only available since update version 4.4. The scripts seems to be realized in Apple's environment and is not part of Eve Aqua itself. Eve Aqua itself, only provides the category of irrigation systems.

For further information, see their homepage[6].

2.1.3 Hydrawise

Hydrawise started very early to bring smart irrigation systems to the market. It was founded in 2011 and was acquired by Hunter Industries in 2016. The company has settled more into professional area of application then the other smart home solutions.

They also use a IF-THEN decision system, called "water triggers". Most notably is the use of an evaporation prediction that also affects the plans, called "Predictive Watering^{tm}"[7]. There is not much information about their prediction algorithm published, but it can be assumed that they do not use a specified algorithm for the actual garden, as a moisture sensor is not essential for their system.

[1] https://cloudrain.com/ [Accessed on 2021-07-30]

[2] https://www.kickstarter.com/projects/1120637488/cloudrain-your-smart-garden-irrigation [Accessed on 2021-07-30]

[3] https://www.apple.com/ios/home/ [Accessed on 2021-07-30]

[4] https://smartapfel.de/homekit-bewaesserungssteuerung-wetterabhaengig-starten/ [Accessed on 2021-07-30]

[5] https://www.youtube.com/watch?v=XGgMf34uWhY [Accessed on 2021-07-30]

[6] https://www.evehome.com/en [Accessed on 2021-07-30]

[7] User Manual with a short description of the System of Hydrawise https://www.hydrawise.com/printpdf/api/printsupport/en-us/article/360009285814#:~:text=Water%20triggers%20are%20the%20settings,watering%20schedule%20or%20cancel%20watering. [Accessed on 2021-07-30]

For further information, see their homepage[8].

2.2 Gardena

Gardena's Smart Irrigation Control, in combination with their smart App, also gives the possibility to generate adaptive watering plans. In addition, Gardena has an optional moisture sensor, that can be used to adapt the watering plans to the actual needs. And since June 2020 they support weather forecast data to adapt the watering plans. It seems very likely to Hydrawise, but the algorithm is a Blackbox (no direct user defined IF-THEN statements). Making it user-friendly on the one hand, but on the other hand, the user has less direct control.

They provide a "Watering Wizard", that supports the user to generate watering plans. It asks a few questions, like the kind of plants, is it potted or in the garden, location (sunny, shadow) and the used watering system (sprinkler, drip irrigation).

For further information, see their homepage[9].

2.3 Comparison to the Bachelor Project System

All mentioned systems have in common, to modify given plans according to the weather forecast. In comparison, their systems can be described, as a top-down process, whereas the ASHGA, using the ANN-based evaporation prediction and AI-Planner, is using a bottom-up process.

Instead of modifying given plans by the information, and if possible, they check, if it fits to the moisture level, the ASHGA generates the watering plans complete autonomous. The planer tries to find the most efficient plan, by a cost function, where drying out and watering have costs. In this way the planner can evaluate on its own, if watering is needed or if it is possible to wait for rain, without hazard for the plants (see chap. 5 AI-Planner and 4 ANN-Based Evaporation Prediction). In this way it is a bottom-up process, where the moisture state of the garden is in main focus and the plans are developed, best fitting to the (future) moisture level.

Using a score function gives the ability to add constraints in a simple way, for example by raising the costs for watering during hot daytimes. The ability to weight up the decisions gives a clear advantage against simple decision trees or IF-THEN statements, as it is less static and can adapt to unpredicted situations, by a cyclic repeating planning.

In addition, the ASHGAt takes control over more tasks than only an irrigation system. It integrates many devices to care about the garden autonomous, as artificial lighting, heating and a shutter to protect against hazardous precipitation.

[8] https://www.hydrawise.com/ [Accessed on 2021-07-30]
[9] https://www.gardena.com/ [Accessed on 2021-07-30]

	Cloudrain	Eve Aqua standalone/Apple	Hydrawise	Gardena	ASHGA
pre def. Plans	X	X / X	X	X	–
adapt pre def. Plans	X	– / X	X	X	–
uses Weather Forecast	X	– / X	X	X	X
uses Moist. Sensor	–	– / X	X	X	X
self planning	–	– / –	o	o	X
add. Tasks	–	– / –	–	–	X

Table 2.1: Functionality of the different systems.
– = not available; o = limited/unknown; X = available

2.4 Outlook besides Smart Home Irrigation

Besides the domain of smart home gardening, focused on irrigation, there are many more examples, where intelligent systems are in action and/or weather forecasts are used.

2.4.1 FarmBot

There are many interesting projects, especially in the DIY-Sector that assume more tasks than only watering. One project to mention is the FarmBot, it is an open-source project, which highly automize smart home gardening. They use a CNC style robot with different device attachments to realize the work.

The system can handle the following tasks:

- Weed: Using a camera and intelligent algorithms, it can recognize weeds and differs them from the cultivated plants.
- Sow: With the CNC robot, sowing new plants is also possible.
- Watering: Of course, as the most smart home garden assistants, it takes control over the irrigation.
- The system is still under development and gets improved the whole time, so even more possible tasks will follow.

The Project is not only open-source, it is also open-hardware. For further information, see their project homepage[10].

[10] https://farm.bot [Accessed on 2021-07-30]

2.4.2 Climate Computers for Greenhouses

In professional greenhouses, climate computers are fully established. They differ from the affordances of smart home gardening systems. They are rather regulative than predictive or preventive and have many more aspects to control. In comparison, the part of this thesis on saving water consumption by rain, is obsolete, as they are closed and the water consumption is mainly optimized by using cisterns instead of using rain directly.

The tasks of climate computers have a huge range[Von10], this is only an extract:

- Watering
- Air-Conditioning: The climate in a productive greenhouse has to be controlled in a very restrictive range, they have to be cooled or heated.
- Ventilation: Plants in greenhouses are very prone to diseases like meldew. As the plants are concentrated, this is a huge issue. To prevent this, air humidity has to be regulated for example by a ventilation system.[npc]
- Light Regulation: Even the light income can be regulated by artificial lighting or by shutters. This optimizs the condition for the growth phase of the plants, as some plants regulate their grwoth by light hours. This is called photoperiodism.
- CO_2 Regulation: As the plants are in closed environment, additional CO_2 is supplemented, as it can drop under a healthy level and higher concentrations boosts the growth.[PD17]

2.4.3 RimPro

A completely different domain, also using weather forecasts and information of plants, is scab fungus forecast for orchards. RimPro uses information of the individual, last fungicide appliance, prognoses of occurrence of the fungus spores, wind direction, air temperature and humidity to predict the probability of an infection and compares it to the last fungicide appliance, checking if it is still sufficient.

This is a very helpful tool for organic farmers, as the use of insecticides and fungicides are restricted. It also brings benefits to the whole environment, as such hazardous substances are only in use, if they are needed.

RimPro itself is again a cloud-based system. See their homepage[11] for further information.

[11] https://www.rimpro.eu/ [Accessed on 2021-07-30]

2.5 Evapotranspiration Prediction Methods

Of course, there are many attempts to predict the evapotranspiration. Precise information of the required water is very important in agriculture, as it can reduce costs and increase the harvest. The most equations need very detailed information of the climatic conditions, the structure of the surrounding landscape, precise weather forecasts, soil characteristics and many additional parameters. These parameters are often hard to measure precise enough and come with high efforts. So, there is a lot of research to predict the evapotranspiration with high accuracy, even with incomplete data.

In the field of agriculture, commonly the ET_0 is used as a standardized value for the evapotranspiration. Thereby it is used in many applications. The FAO56-PM equation is one of the most popular solution to calculate the ET_0.

"An application of the FAO56-PM equation requires data on solar radiation, wind speed, air temperature, vapour pressure, and humidity. However, all these input variables may not be easily available at a given location. In developing countries in particular, difficulties are often faced in collecting accurate data on all the necessary climatic variables, and this can be a serious handicap in applying the FAO56-PM equation. Among the inputs needed, temperature data are routinely measured and solar radiation can be estimated with sufficient accuracy. But the other variables are generally measured at only a few locations." [JNS08]

To clarify the units: The most equations (as FAO56-PM) calculate the evaporation per day = mm/m^2day^{-1}.

2.5.1 Method of Sharad K. Jain at al., ANN-based

The work of Jain, Sharad K., Sudheer, K. P. and Nayak, P. C, "Models for estimating evapotranspiration using artificial neural networks, and their physical interpretation"[JNS08], also uses an ANN-based method. They tried different models and trained them with different (in)complete datasets. In their research, they used hourly data, to predict the ET_0, but had a different approach than this bachelor thesis. They tried to train the ANN to predict the results of the FAO56-PM equation, instead of real measured results.

Their model, using the most complete information (Temperature, Humidity, Dew point, Radiation, Wind speed), had an SE of 0.26, to the equation.

2.5.2 Method of Wei-guang Wang et al., using Phase-Space Reconstruction

The work of Method of Wei-guang Wang, Shan Zou, Zhao-hui Luo, Wei Zhang, Dan Chen, and Jun Kong, "Prediction of the Reference Evapotranspiration Using a Chaotic Approach"[KWZ+14] had even a different method. They used to do a multidimensional phase-space reconstruction. As reference they also used the FAO56-PM equation, based on the data of the time interval from 1966 to 2005 in four different locations. Their Method had an SE of 0.1999 to 0.3054, depending on the location (compared to the FAO56-PM equation).

2.5.3 Method of Mario Córdova et al., evaluating the FAO56-PM Equation on incomplete Data

In their publication "Evaluation of the Penman-Monteith (FAO 56 PM) Method for Calculating Reference Evapotranspiration Using Limited Data"[CCC+15], Mario Córdova et al. tried to figure out, how good the FAO56-PM equation works, if information is incomplete and assumption have to be made. For example, if there are no existing measurements of windspeed, in this case, they set it to a default value of 2m/s.

As reference, they used the prediction of the FAO56-PM equation with a full dataset. They had an SE of 0.06, if only windspeed was missing to 0.73, if solar radiation, humidity and windspeed was missing.

2.5.4 Comparison to the results of this thesis

For comparison: The FAO56-PM equation gives the prediction as mm/m^2 per day. The best model, generated during the experiment of the bachelor thesis, had an SE of 1.8Percentage Points (pp) of moisture level per day. From experiments, it could be evaluated that one liter gives 8pp in moisture level, where the pods have a size of $0.35m^2$. So, one liter is assumed to give 2.8pp in a similar raised bed of the size of $1m^2$. Using this calculations, the model of this thesis had an SE of $0.64mm/m^2$ per day (see chap. 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models).

Of course, this comparison is not very meaningful, as this thesis has just measurements of a short time and many assumptions are made, like calculating with the same SE from a smaller raised bed, as a $1m^2$ sized one and not expecting an error from the FAO56-PM equation. Also, they only calculate the ET_0, instead of calculating the ET_c or the real evapotranspiration. The models of this thesis, calculate the complete expected evaporation. So, there are additional error sources in the calculation process, from ET_0 to the prediction of the real evapotranspiration.

In the paper "EVAPOTRANSPIRATION OF IRRIGATED ALFALFA IN A SEMI-ARID ENVIRONMENT"[EHT+21], a SE of 0.6 mm to 0.8mm is measured. They compared the predictions of the Penman-Monteith equation to real measurements from weighing lysimeters[12]. In their experiment, they cultivated alfalfa.

Comparing the results of this paper with the results of the thesis, could give the conclusion that the Penman-Monteith equation performs as good as the prediction model of the thesis. But it is highly speculative that the experiment would give the same results in raised beds or small garden parts, instead of an alfalfa field or vice versa. In conclusion, no direct comparison can be drawn between the method of the thesis and the mentioned methods. At least for now, without further experiments.

[12] A weighing lysimeter is a permanently installed device to measure the soilwater balance

3 Overview and short Description of the System

The ASHGA has three subsystems. The Smart Home Solution, in this case, openHAB. The Raspberry Pi Devices and Sensor System and the AI-Planner. They communicate with each other, where openHAB acts as distributor for commands and measurements by network interfaces. So, they are highly independent and can run as a distributed system. As example, openHAB may run on an separate server, handling also the other tasks of the users Smart Home. The Raspberry Pi Devices and Sensor System can be placed, near/in the garden, as separate device and the AI-Planner either runs on the Smart Home Server, the Raspberry PI or a third server. But the whole system is lightweight designed, the Raspberry PI is able to run all components, as is realized in this prototype.

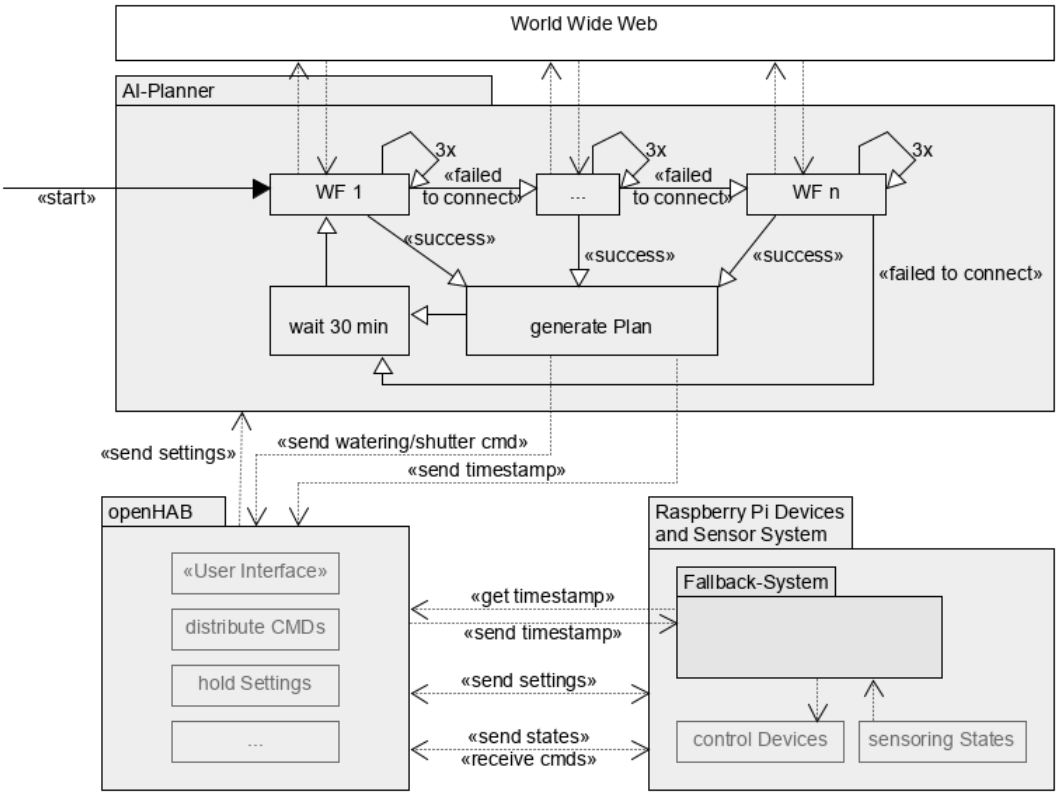


Figure 3.1: System Components and Interaction between them

3.1 Smart Home Solution: openHAB

The smart home solution “openHAB” is an open-source, Java-based project. Because it is Java-based, it is independent from the used hardware and operating system. It provides a web-based UI, but also provides a smart-phone app.

The principle of openHAB, is to hold states of devices and sensors in “items”. But items can also be internal states, strings, numbers and so on. Their “binding”-system is used as interface, connecting items with real devices. It can be seen as driver. They offer a repository system for a huge number of devices and sensors, e.g. Gardena devices, Philips Hue, Smart TVs, intelligent power sockets and more. The bindings offer the connection to WI-FI based devices, but support even more protocols, like MQTT.

OpenHAB provides a rule system, that can be triggered by time (scheduling tasks), state change of items or by receiving commands. These rules are written in a Java-based Xtend variation and offers the possibility to process many tasks, directly in openHAB.

The ASHGA uses a TCP-Binding, for the communication with the Raspberry PI Device and Sensor System. Therefore, a string item is bound as TCP-Client. Incoming messages trigger a rule, that distributes new (sensor) states to openHABs other items. In this way, the temperature, or other sensor data, are held. Outgoing messages are written to the TCP-Socked and send to the Raspberry PI Device and Sensor System. See chap. 3.4 Raspberry Pi Devices and Sensor System for an example, where a command to water is send. Additionally, all items can be read or changed, by a REST-API. This is used in the AI-Planner, to get the current soil moisture level.

The functionality of openHAB:

- Items: hold states of devices, sensors or other states.
- Binding: Connect real devices to Items.
- UI: Different UIs, like Sitemaps, HABPanel or smart-phone apps.
- Switches/Sliders: Simple manual triggers for Items/Devices.
- Rule-Engine: Provides more complex tasks. Can be time based or triggered by state change.
- Persistence: Holds the history of item states. Can be used for charts of measurements, or to read them via REST-API (used to train the ANN).
- Connectivity: TCP, MQTT, REST-Api and many others.

Items can be displayed in generic defined sitemaps or in a WYSIWYG-Editor (called “HABPanel”). There are predefined sliders or switches, to change the state of the items.

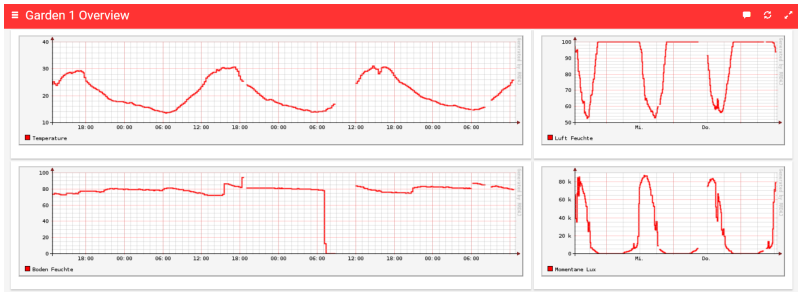


Figure 3.2: UI for charts of the measured data of Garden Part 1

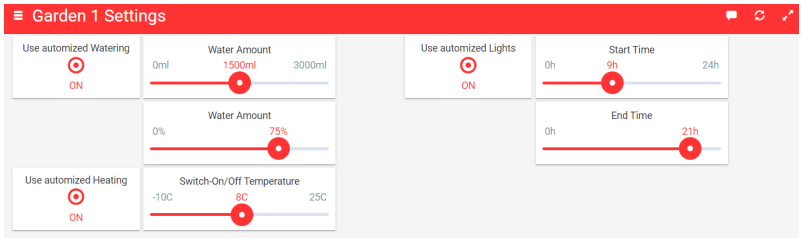


Figure 3.3: Fallback System settings of Garden Part 1

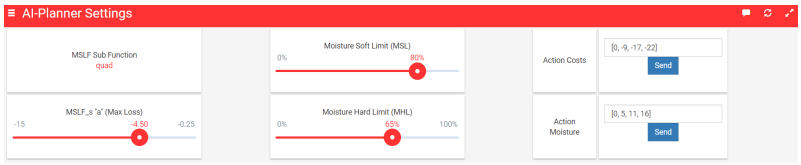


Figure 3.4: AI-Planner settings of Garden Part 1

See the project homepage for further infromation.

3.2 AI-Planner

The AI-Planner is the heart of the implemented irrigation system. As mentioned in the introduction, it uses weather forecasts, information of the actual moisture and the evaporation prediction, to calculate watering plans. If it is needed to water, the command is distributed to the Raspberry Pi Devices and Sensor System. There are also some fail-safe methods, if something goes wrong. See chap. 5 AI-Planner and chap. 4 ANN-Based Evaporation Prediction) for a detailed description and evaluation.

3.3 Weather Forecast

The weather forecast, actually used in the ASHGA, is tomorrow.io, formerly ClimaCell. They are commercial, but offer a restriced free business model for developers. Tomorrow.io offers a REST-API, returning the weather data as JSON-Objects. They offer all required information for the project, but can provide very interesting additional data, as pollen count for allergy sufferers. See their homepage[1] for further information.

Of course, it is possible to use any other weather forecast provider that provides the required information, such as temperature, humidity, solar radiation and precipitation prediction. For this, however, the interface of the system has to be rewritten and adapted, as every provider has its own API.

3.4 Raspberry Pi Devices and Sensor System

Most of the used sensors and devices of the ASHGA are wired. They use either one-wired protocols or more common protocols as i^2c. There are also sensors and devices that use simple circuit switches as input or output. To give access to them, the Raspberry PI centralizes the communication in the physical layer, realized with the GPIO-Interface of the Raspberry PI. A Raspberry PI 3b is used for this task. The communication, to receive commands or send measurements of the sensors, are abstracted to JSON-Style messages between openHAB and the Raspberry PI, TCP is used as protocol. This separation by the network interface, gives the option to deploy a distributed system. The Raspberry PI Devices and Sensor System can be connected to the smart home solution, for example by using a power-line communication, if there is a power-line to the garden. In this way, the wires to the sensors and devices can be shortened and prevent signal interferences.

For example, if openHAB sends the command to water, the message looks as following and provides, besides the command itself, the amount of water:

```
{"startWatering":"600"}
```

This sub-system parses the JSON string and distributes the command to the physical devices. The time to enable the relay of the 230V pump is calculated by the flowrate. For example, a flowrate of 1800 ml/min and the message to water 600 ml, gives a trigger time of 20 seconds.

The measurements of the sensors are triggered in a loop of a few seconds. The most measurements are denoised by measureing three times and taking the median, also there is exponential smoothing[Nau] used. The collected data are sent to openHAB, again capsuled in a JSON message.

[1] https://www.tomorrow.io/ [Accessed on 2021-07-31]

3.5 Sensors and Devices

This is a short description of the used sensors and devices, to give an overview. See Appendix A.1 Detailed Sensor and Device List for more information.

The sensors are:

- Moisture Sensor: Capacitive moisture sensor
  Protocol: i^2c
  This kind of moisture sensors have big advantages over other systems.[Sha18] They are simple to install, sample rate and measure process is fast, they are contact-less and not disturbed by ions in the soil. MakerPortal gives qualified information of the functionality.[Hri20] In short: The dielectricity of the surrounding material is measured, the dielectricity changes (almost linear) by the containing water of the soil. This can be measured by a frequency drop of the oscillator in the sensor.
- Sun Radiation Sensor: Lux-Meter
  Protocol: i^2c
  Such lux-meter are realized by a photo diode, the resistance depends on the incoming light. The behavior depends on the light wave, but if the light source is always the same, the proportion of the resistance is also the same. So, it has to be calibrated once, to the spectrum of the sun. In the latitude of Germany, 100,000lx are assumed on clear-sky day light.
- Temperature and Humidity Sensor:
  Protocol: Vendor specific 1-Wire protocol
  It is a combined sensor for temperature and humidity and pre-calibrated.
- Water Tank Level Sensor:
  Protocol: Voltage ON-OFF-Switch
  Float Sensor, switches circuit on or off, can be read directly via GPIO. Disables Watering, if water level is too low.

The devices are mostly 230V Devices, switched by a relay. This allows the compatibility to exchange them with similar devices. For example, instead of a water pump, a electro-magnetic valve can be used, without changing the code.

- Watering: Water Pumps
  Protocol: 230V, relay
  The time interval for the watering is calculated by amount/flow rate.
- Artificial Lighting: LED-Grow-Lights
  Protocol: 230V, relay
  Over 200 single LEDs, in red and blue light spectrum, optimized for plants. LEDs are used, to minimize the power consumption.
- Shutter: Self constructed mechanics
  Protocol: Specific stepper motor protocol

- Soil Heating: Heating Plates for Gardening
  Protocol: 230V, relay
  Two of them are installed in the raised bed. They use heating wires, 15W each. Could be replaced or extended with air heaters.

3.6 Automatous Sensors

Of course, it is possible to use sensors, that are not integrated or controlled by the Raspberry PI Sensor and Device System. This can be realized, for example, by sensors that have integrated WI-FI or other network protocols, so they can communicate with openHAB directly.

To do a show case of such an implementation, a LILYGO Higrow[2] sensor is used. It is a moisture sensor, but includes a temperature/humidity sensor and a light sensor. For a full autonomous system, a solar panel and a lithium battery is attached.

The sensor uses an ESP-32 processor with a included WI-FI module. In this way, it sends the measurements directly to openHAB, without the Raspberry PI Device and Sensor System.

Specification:

- Processor: ESP-32
- Sensors: Capacitive soil moisture sensor, and others (not used in this project, as they are redundant or not practicable, due to the outdoor protection case)
- Communication: WI-FI, JSON messages, direct to openHAB via REST-API
- Power Supply: Solar panel and lithium battery
- Measurements: Every 2 minutes. To save power, after the measurement, the device goes into deep sleep mode for 2 minutes.
- Runtime: Without recharging from the solar panel, about 10 days. Even cloudy days can recharge more, then it consumes. With solar panel: Infinity runtime.

[2] http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=1172&FId=t3:50033:3 [Accessed on 2021-07-31]

Figure 3.5: LILYGO Sensor with solar panel and battery

3.7 Fallback System

The origin idea was to create a irrigation fallback system, but it turned out that it is usefull to apply some of the less complex tasks to this system. As the Raspberry PI Device and Sensor System has direct access to the most sensors and devices, it can do simple tasks, without communication and commands from openHAB. This tasks are the control of the heating plates and lights. In case, openHAB or the AI-Planner is not reachable, the garden is cared anyway, even if parts of the whole system have crashed (see chap. 5.5 Fail-Safe Methods). Settings of the System, like minimum moisture, time interval of the artificial lighting and the temperature threshold to start heating are set in openHAB and loaded on start.

As long as openHAB is online, the Fallback System can operate on data, provided from sensors that are not directly integrated into the Raspberry Pi Device and Sensor System itself. Garden Part 2 is observed via the mentioned LILIGO Sensor and watered by the Fallback System.

Tasks of the Fallback System:

- Irrigation Fallback System: The AI-Planner sends a timestamp as life-signal to openHAB, if it is outdated or openHAB is not reachable, the Fallback System steps in, with its own, less intelligent watering routine. Realized in Garden Part 1.
- Simple Irrigation System: As mentioned, it waters, if the moisture falls under the given moisture level. Then it waits 15 minutes, until the water has spread and the measurements are correct again. The functionallity is a “Water at Moisture X”-System. Used in Garden Part 2

- Artificial Lights: The artificial light are turned on in the given time interval (e.g. from 9:00 to 21:00) , if the light falls under the threshold, sunshine duration is measured. “[...] sunshine duration during a given period is defined as the sum of the time for which the direct solar irradiance exceeds 120w/m^2”.[Org]
- Heating: The heating plates are turned on, if the temperature falls under the given value (e.g. 10°C)

To prevent the relays from flickering, a hysteresis is implemented. It compensates noise from the sensors. For example, the heating starts at 9°C but stops at 11°C. This is used for all measureands, except for the watering.

4 ANN-Based Evaporation Prediction

The goal is to do the evaporation prediction simple and user-friendly, especially for Smart Home Gardening Assistants, but accurate enough for good irrigation plans. Considering the mentioned issues, especially in gathering of precise data (see chap. 2.5 Evapotranspiration Prediction Methods), a self-learning and robust algorithm has to be developed.

Section 4.1 Background Information about ANNs provides provides a short introduction into the field of ANNs, to clarify some terms, used later in this chapter. The first part is about the functionality of ANNs in general, the second part provides a short overview of the used ANN-engine and ANN-API, TensorFlow and Keras.

The most equation for evapotranspiration prediction are good for huge landscapes and time intervals of months and years. So, such equations are good for climatic scale predictions but not suitable for highly localized small garden parts; a raised bed will differ from huge fields in terms of evaporation. A refined and nowadays commonly used equation is the FAO-Penman-Monteith (FAO56-PM)[CCC+15][npa].

But even if such equations, that need many parameters them self, would work for a small garden or raised beds, there are many more parameter to figure out for the garden, the most important once are as following:

- ET_0: Evapotranspiration, it is the amount of water, that evaporates per day mm/d (e.g. FAO 56 PM). Besides other information, this equation already needs to make use of the knowledge concerning incoming and outgoing radiation that depends on plant coverage and the light absorptions index of the soil.[IIAJ03] "[...] temperature and radiation data are the most crucial inputs" [JNS08].
- K_c[npb]: Crop coefficient of the plants. It defines the additional transpiration from the plants. This value depends on the growth of the plants and the cultivated plants themselves. Together with ET_0, the actual evapotranspiration ET_c is calculated.
- nKF Usable field capacity of the soil. It is the amount of water, the specific soil of the garden can hold.
- Groundwater level and/or stored water in the soil of deeper layers: Remoisturize the upper layer, by capillary water transport.

And some of these parameters would have to be updated over time, as the crop coefficient K_c.

So, either, the user has to have a degree in agricultural science to figure out all parameters of their garden, or the system is able to adapt automatically to the specific conditions of the garden and cultivated plants. And the system should do a precise prediction of the evaporation, based on data a

few and affordable sensors can provide. They also should be common sensors that are often used in smart home solutions. In the case of the ASHGA, there is a lux meter, temperature and humidity sensor and a soil moisture sensor.

According to the good results of the studies of Sharad K. Sudheer et al., a neuronal network based solution seems to have great potential to solve the problem of evaporation prediction. This are the advantages and disadvantages of ANNs, the list is related to the summaries of Maad M. Mijwe[Mij18] and Jahnavi Mahanta[Mah17]:

Huge data sets: They handle a huge amount of (almost) unfiltered data. After one week, with a sample rate with one stored value per hour, there are 168 measurements of each sensor stored. After a month, there are 720 data points per sensor.

Ability to work with incomplete knowledge: Learning patterns in the data gives the ability to predict unseen and untrained conditions.

Not domain specific: There is nearly no need to process the data in any way to work on the algorithm.

Learning non-linear and complex relationships: The evapotranspiration is a highly dynamic process and is influenced by many factors. As mentioned, the work of Sharad K. Sudheer et al had already good results on a similar problem:

> "The results of the study indicate that the ANN can estimate ET_0 accurately even if data for only these two variables [temperature and radiation data] are available. An ANN model whose input consists of temperature and humidity or temperature, humidity and wind speed cannot provide a good estimate of ET_0 because the major predictor variable (radiation) is not present in the input vector."[JNS08]
>
> Another dynamic process is the upward flow from deeper layers to the top layers, remoisturising them.[npe]

Disadvantages of ANNs:

Hardware dependence: They work best on highly parallel computing systems. This is contrairy to a Raspberry PI, but only predicting is less computationally intensive than learning. In fact, doing the 24 predictions for a full plan of the AI-Planner is done in less then 5 seconds (see chap. 5 AI-Planner).

Unexplained behavior of the network: That is a huge point, as it is not possible to check the correctness. So, they have to be evaluated after generating and training.

Determination of proper network structure: There is no algorithm to get the best structure for an ANN. The structure means, number of nodes each layer has, how much layers there are in total and the activation function of these layers. The structure has a big impact of the learning ability of the ANN. They are, more or less, handcrafted and must be figured out by trial and error. A solution to this challange are genetic algorithms. They adapt the structure to the problem by evolution. This means, taking the best trainable models and slightly modify them random, step by step. The first step of a genetic algorithm is implemented in this work: By training a bunch of different, random generated models, I found a good structure for this problem.

Difficulty of showing the problem to the network: In fact, as there are just a few numerical input parameters and one output, this is a minor problem. The different behavior, by adding additional inputs, seen in 4.2 Data and Models of the Evaporation Prediction, is part of this issue.

4.1 Background Information about ANNs

ANNs are graphs of connected artificial neurons. A simple dense ANN is a full connected directed graph, but there are many other possible structures for ANNs, for example in the area of image processing or speech recognition. ANNs are defined by the number of neurons per layer (shape of the layer), number of layers and their activation function (see Figures 4.1[1] and 4.2).

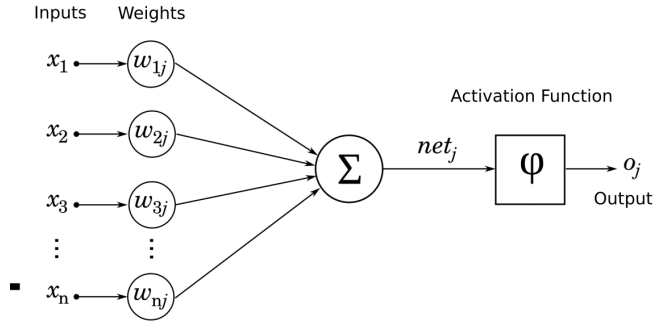


Figure 4.1: Functionality of Artifical Neurons, there is often an additional node x_0, called bias and set to 1.

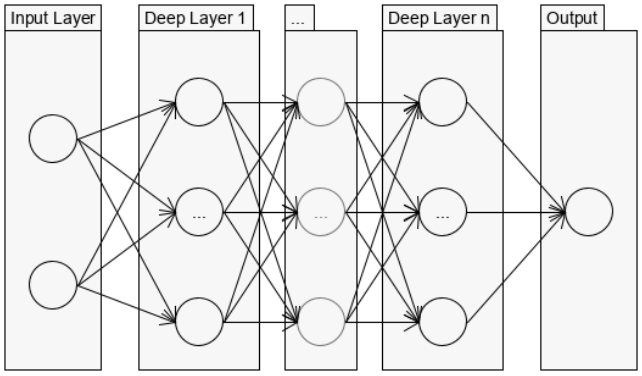


Figure 4.2: Structure of an ANN

[1] Translated and modified from https://de.wikipedia.org/wiki/K%C3%BCnstliches_neuronales_Netz#/media/Datei:NeuronModel_deutsch.svg [CC BY-SA 3.0] [Accessed on 2021-07-30]

The output of a neuron j is defined as:

$$o_j = \varphi(net_j), \text{where } net_j = \sum_{i=0}^{n} x_i w_{\text{ij}}$$

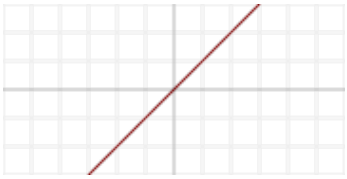
φ is the activation function of the layer of neuron j
n is the number of inputs of the previous layer.
x_i is the input i
w_{ij} is the weight between input i and neuron j

Often there is an additional node x_0, called $bias$ and set to 1. Very similar in functionality is the use of a threshold given to φ. Such additions give the ability to shift to the curve of φ, independently of the inputs. The whole ANN can be represented as a matrix, so the computation is done by multiplying the resulting matrix with the input as vector.

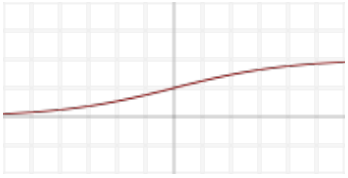
4.1.1 Activation Functions

To choose different activation functions, especially in deep layers, can improve the behavior of an ANN.[NIGM18] There are many different activation functions. In this section, some of the most interesting ones are discussed.

Linear: [2] Is often used in the input and output layer. In deep neuronal networks, they often do not give huge benefits, as the combination of the single linear functions can be described by one single function.
$\varphi(x) = x$



Sigmoid: [3] They have the benefit to keep values in the deeper layers small. If the input values of a neuron get too high, this has to be compensated by the weights. As result, there is a high dynamic in the network. Small differences in the input can result in huge differences in the output. This is often not desired, as near inputs often should give similar outputs.
$\varphi_a(x) = \frac{1}{1+e^{-ax}}$.

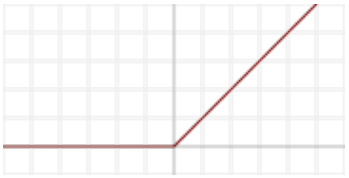


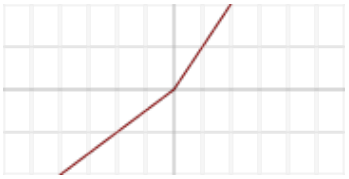
[2] Plot from https://en.wikipedia.org/wiki/File:Activation_identity.svg [CC-BY-SA 4.0] [Accessed on 2021-07-30]
[3] Plot from https://en.wikipedia.org/wiki/File:Activation_logistic.svg [CC-BY-SA 4.0] [Accessed on 2021-07-30]

ReLU and parametrized LeakyReLU: [4] They are very new functions (for use in ANNs), as simple they are, it turned out that they give a huge improvement to neuronal networks.[MHN13][ZRM+13]

The ReLU function has a disadvantage in training: The backpropagation uses the derivative during training. This is 0 if $x < 0$. So, the neuron may die and is not trainable anymore. An improvement to this issue is the (parametrized) LeakyReLU function.

$$\text{ReLU: } \varphi(x) \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{if } x < 0 \end{cases} \qquad \text{LeakyReLU: } \varphi_a(x) \begin{cases} x, & \text{if } x \geq 0 \\ ax, & \text{if } x < 0 \end{cases}$$





4.1.2 Training

An ANN can be trained by a technique, called backpropagation.[Roj96][5] To train an ANN, a prediction on the training data is made. By comparing the prediction with the desired result (in this case the measurements), the error is evaluated, this step is called feed-forward. Usually the error function is (half) the MSE on the training data (but there are some other functions, e.g. for classification problems):

$$E = \frac{1}{2} \sum_{i=1}^{n} (t_i - o_i)^2$$

t_i is the desired value and o_i is the prediction.

To minimize the error, the weights are newly balanced by:

$$\Delta w_{\text{ij}} = -a \frac{\partial E}{\partial w_{\text{ij}}} = -a \delta_j o_i$$

$$\delta_j = \begin{cases} \varphi'(net_j)(o_j - t_j), & \text{if it is an output neuron} \\ \varphi'(net_j) \sum\limits_k \delta_k w_{\text{jk}}, & \text{if it is a hidden layer neuron, } k \text{ is a neuron, having neuron } j \text{ as input} \end{cases}$$

$$w_{\text{ij}}{}^{new} = w_{\text{ij}}{}^{old} + \Delta w_{\text{ij}}$$

[4] Plot from https://en.wikipedia.org/wiki/File:Activation_rectified_linear.svg[Accessedon2021-07-30] and https://en.wikipedia.org/wiki/File:Activation_prelu.svg [CC-BY-SA 4.0] [Accessed on 2021-07-30]

[5] Englisch PDF version, aviable on the homepage of the "Freie Universität Berlin" https://page.mi.fu-berlin.de/rojas/neural/neuron.pdf (page 158ff) [Accessed on 2021-07-30]

a is the learn rate, it must not be too small, because the convergenze to an optimum would take very long, but it also must not be too high, as it then adapts only to the last training, and forgets the previous learning.

The optimizer has to find the minimum of E by finding $\partial E = 0$. To do this, it variates the learning rate, each training step. It has to figure out the right learning rate and step size to not step over the minimum but also find the real minimum and not a local minimum.

The Evaporation Prediction Models uses Adam as optimizer, since it is robust and fast.[Bus18][KB17]

4.1.3 Keras and TensorFlow

Keras is an open-source ANN library for python that acts as an interface for TensorFlow, whereas TensorFlow is a powerful ANN engine. TensorFlow supports parallel computing and GPU computing, but is still efficient enough to make predictions on a Raspberry PI. Keras is used for this project, because it is simple to use, open-source and as python is used for the other sub systems, it fits well to the whole system.

Keras has a simple pipeline structure to generate and train models.

- Load/define training data
- Define the structure of the model
- Compile it
- Train it
- Save it

This is done in less than 10 lines of code[6]:

[6]Code is inspired from https://de.wikipedia.org/wiki/Keras [Accessed on 2021-07-30]

Listing 4.1 Example: Generate, train and save a model

```
# Simple example of input and output values
input_values = np.array([[0, 0], [0, 1], [1, 0], [1, 1]])
output_values = np.array([[0], [1], [1], [0]])

# Define a model with 2 input nodes,
              2 hidden nodes and one output node
num_inner = 2

model = Sequential()
# New Layers are defined here.
model.add(Dense(num_inner, input_dim=2, activation='sigmoid'))
model.add(Dense(1))

# Compile the model
model.compile(loss='mean_squared_error',
              optimizer='adam',
              metrics=['accuracy'])

# Train the model on the input and output values
model.fit(x= input_values, y= output_values, epochs=10000, verbose=0)

model.save('filename')
```

Use the model to predict values is even simpler:

Listing 4.2 Load a model and predict values

```
model = load_model('filename')
model.predict([some input])
```

4.2 Data and Models of the Evaporation Prediction

The sensors of the Garden Assistant provide information about the temperature, air humidity and solar radiation (lux meter). These are the possible inputs of the ANN; the measured soil moisture is the output that should be predicted.

The data of the measurements are centralized and stored in openHAB in a round robin database, rdd4j[7]. To use them for the training of a model, they are collected and stored into a csv file (see chap. 5.11 Additional Features and Tools). Each entry of openHAB's stored data is the mean of the measurements during one hour, so the sample rate of the data is 1/h.

[7] https://www.openhab.org/addons/persistence/rrd4j/ and https://oss.oetiker.ch/rrdtool/ [Both accessed on 2021-07-30]

The weather forecast also provides this data, only the solar radiation is given in w/m^2 instead of lux. So, the solar radiation from the weather forecast has to be transformed to fit to the model. Therefore, the equation from the Institute of Electrical and Electronics Engineers[Mic19] is used:

$$1000\frac{w}{m^2} = 120,000lx$$

The output should be the predicted change rate in the soil moisture level per hour, so it is the derivation of the moisture level curve. This is simply calculated (n is the index of an actual entry in the data at a specific time):

$$\Delta M_n = \frac{(M_n - M_{\text{n-1}}) + (M_{\text{n+1}} - M_n)}{2}$$

4.2.1 First Attempt, Model Type "THL"

The first attempt to train the model, was to use the mentioned inputs and ΔM as output. The data was filtered as following:

- If it was watered, this is done because the curve of the moisture raises fast and then drops fast after a few minutes, until the real moisture is measured.
- $\Delta M > 0$ is filtered, because the system should not learn to predict rain.

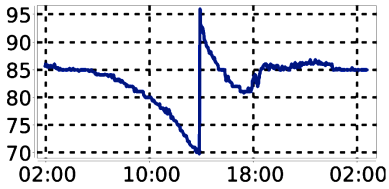


Figure 4.4: Moisture Curve, after watering

Based on the inputs, this kind of models is called "THL", the relevant data of the model are:

- Shape: [3,33,18, 9, 3, 1]
- Activation Function: [LeakyReLU, Sigmoid , Linear, Linear, Linear]
- Inputs:
  - T: Temperature [°C]
  - H: Humidity [%]
  - L: Light / Solar Radiation [lx]
- Output:
  - ΔM: change rate of the soil moisture level [pp/h]
- Filter: $\Delta M \leq 0$, watered

Analyzing this kind of model shows, that the ANN method has the potential to predict the moisture over time, for the AI-Planner, but it had to be improved. Even, if it has an acceptable MSE on the training data, it failed to predict the moisture during longer time intervals. On the one hand, the ANN understands that the moisture level drops faster on day time conditions and drops less over nightly conditions, in terms of temperature, humidity and solar radiation (see Figure 4.6a). On the other hand, it did not learn the remoisturize in the evening, as negative ΔM were filtered, and it would give overlay effects as the conditions in the morning looks same to the ANN as in the evening (same T, H and L).

So, all in all, it has a very bad MSE and SE, and even worse, it fakes a better statistics by watering far too much (see Figure 4.6b).

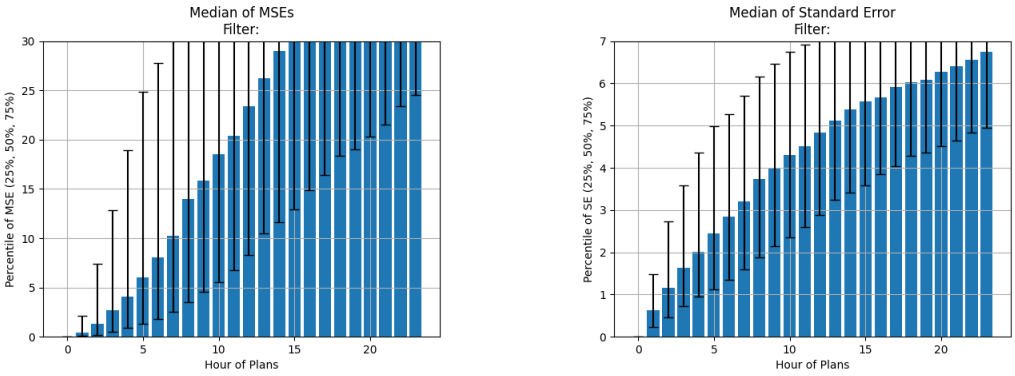
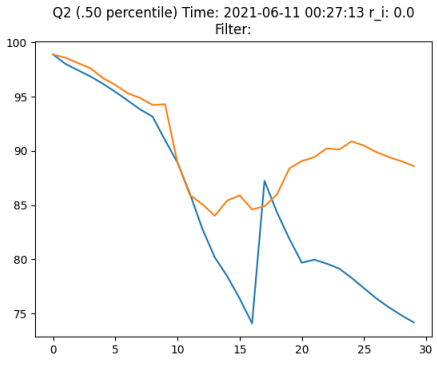
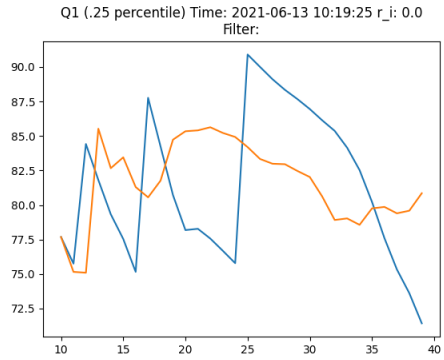


Figure 4.5: MSE and SE of model type “THL”



(a) The model understands a higher evaporation during day times and lower during night. But fails on the remoisturize in the evening.



(b) Faking a better MSE by overestimating watering events.

Figure 4.6: Curves on the quartiles Q2 (median) and Q1 of model type “THL”. The X-axis shows the time of the day (switch to 0:00 at 24:00 could not be implemented)

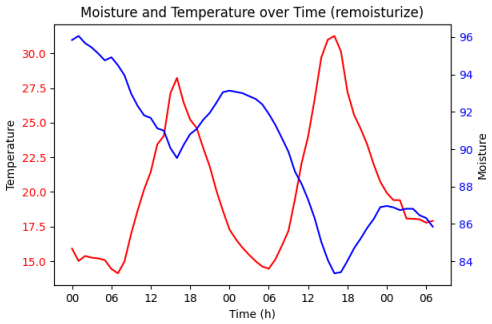
4.2.2 Improved Model Type “THLTD”

To improve the model, more information of the actual state of the garden has to be provided.

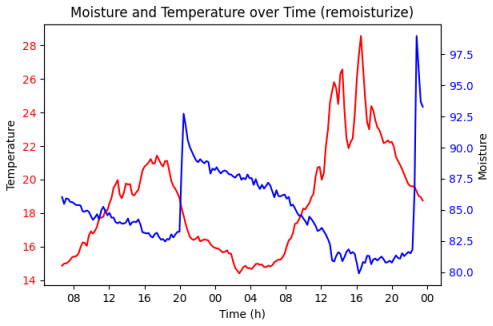
The model had only knowlage of the actual measured soil moisture, but in fact the pot even contains more water in deeper (not measured) layers, thereby it comes to a remoisturize effect, if the temperature drops in the evening (see Figure 4.7a). Also, the higher decreasing rate of soil moisture in the morning depends highly on the temperature change rate. As explanation, the higher layers dry out faster, as the surface is directly exposed to the sun. Later, the up-stream from deeper layers dominate the evaporation, when it gets colder and the sun sets. Besides this explanation, with some speculation on this phenomenon, I think it is also related to the difference of soil and air temperature. ΔT is an indicator of this difference. ΔT, as change rate/ derivation of the Temperature is calculated like ΔM.

As this effect occures nearly the same time (of the day) and ΔT has a huge influence to it, this two parameters are added to the inputs.

In addition, only data with a high ΔM is filtered. This way, it now should be able to learn the remoisturize from the lower soil layers. It could be differentiated from rain, as this effect is about 1pp/h and rain is, depending on the precipitation, far above this effect (see figure 4.7b).



(a) ΔM depending on temperature and time, we can see a increasing moisture level at about 18:00h



(b) Remoisturize from lower soil layers vs. rain. The remoisturizing is less pronounced as left, but it can be differed clearly from rain at 20:00h and 23:00h

Figure 4.7: Moisture and Temperature over Time

Adding the current hour of the day and ΔT (calculated as ΔM) brought this kind of model, called by the inputs "THLTD":

- Shape: [5,33,33,18, 5, 1]
- Activation Function: [LeakyReLU, Sigmoid , Linear, Linear, Linear]
- Inputs:
  - T: Temperature [°C]
  - H: Humidity [%]
  - L: solar radiation [lx]
  - $Time$: hour of the day

  - ΔT: Change rate/ derivation of the Temperature

- Output:
  - ΔM: change rate of the soil moisture level [pp/h]
- Filter: $\Delta M \leq 1.5$, watered

Modifying the inputs in this way, has a huge impact to the error and reduced the MSE and SE strongly.

An important factor is how fast the new model is trainable, in terms of how many days have to be measured for the training data. To evaluate this, a model of this new type is trained on only two weeks of measured data, this model is called "2WM". For a full analysis on this new model type, see chap. 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models. Figure 4.8 shows a short extract to visualize the improvements. Due to the fast-learning ability, it is possible to use the Fallback System as "Water at Moisture X"-System only for a few days and then deploy the AI-Planner.

But nevertheless, using more sensors and inputs surely would bring additional improvements in prediction. But the goal in this thesis is to create a working system on minimal setup, nearly every smart home user either has, or can additionally afford.

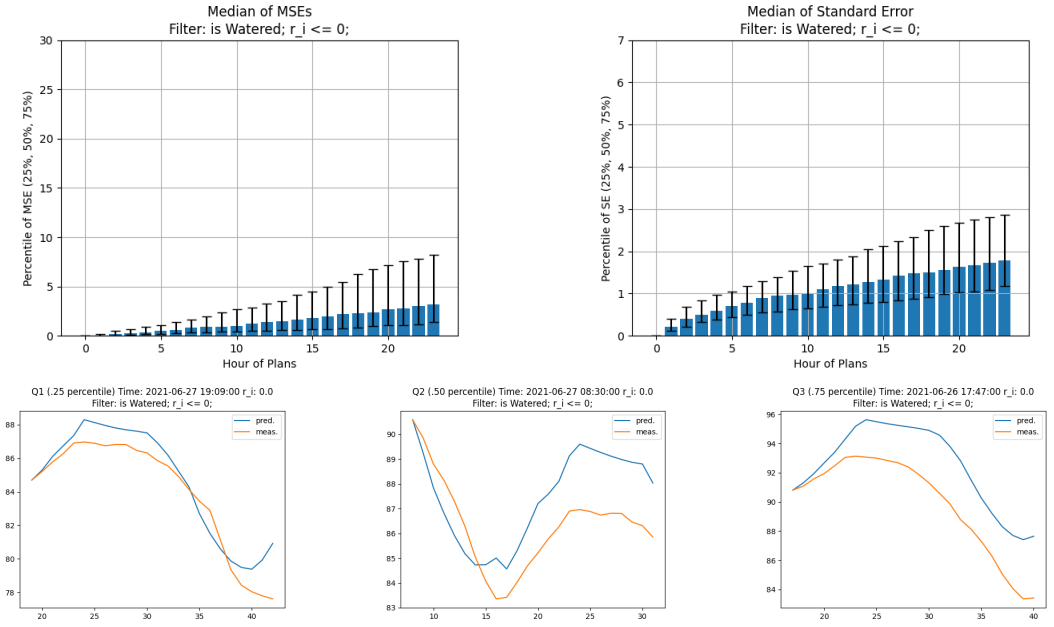


Figure 4.8: A Few Extracts of the plots in chap. 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models

4.3 Possible further Improvements of the Models

As discussed in the previous section, more inputs can improve the predictions of the models. Therefore, it is assumed the model will improve, if there is more information on the actual states of the garden. They could be measured by additional sensors or calculated as ΔT. But there is also the issue of overtraining and overfitting.[TLL95] The ANN may learn false patterns and also can learn noise in the data, overtraining is might be a factor, why "2WM" performs better than model "THLTD1" with less training data apart from the better selection process (see chap. 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models). Subsequently, using more inputs to improve the prediction performance, has to be evaluated.

The possible additional data can be:

- Soil Temperature Sensor: An additional temperature sensor in middle soil layers can make the input of ΔT and the Time obsolete, or support their functionality, to predict the dynamics of the moisture over time.
- Rain Sensor / Precipitation Amount Sensor: The information, if it has rained at a specific time could improve to differentiate remoisturize from rain. It would be possible to filter data not by analyzing of the curve (faster moisture level increase from rain); instead it would be possible to filter it on real rain events.
- Soil Moisture Sensor in deeper Soil Layers: Using a secondary moisture sensor in deeper layers, will give information on the real, total containing water of the garden. Of course, it also affects the remoisturizing and leads to a better prediction of this effect.
- Information on Irrigation in the Past: From Watering or from rain (with a rain sensor). As the additional moisture sensor in deeper soil layers, this can give information of the total containing water in the raised bed/garden.
- Re-Feeding the current (predicted) Moisture Level or from the Past: For example, 5 hours before the current prediction. It can be assumed that the evaporation is not linear vor all moisture levels, it might depends on the actual moisture.
- Wind Speed: If the user has the possibility to use a small weather station, wind speed will be a good additional input, it is also provided by nearly every weather forecast. The FAO56-PM equation suggest that the wind speed influences ET_0. Surely it also influences the evapotranspiration in smaller garden parts.
- ...

5 AI-Planner

The AI-Planner uses the weather forecast and measured states of the garden and acts accordingly to the given information. It delegates commands to openHAB that are distributed to the devices. Primarily, the AI-Planner calculates the watering plan, yet also controls the shutter, if the prognosed precipitation would be hazardous to the plants.

Based on the weather forecast, the evaporation of each hour in the planning time interval is calculated by the ANN-Based Evaporation Prediction. With this data, the planning algorithm calculates when it is best to water the garden. It can also wait for rain under certain conditions instead of watering and hence save water.

5.1 Planning Algorithm

The goal of the Planner is to keep a healthy moisture level for the plants. Two limits of moisture are defined: The first value, Moisture Soft Limit (Soft Goal) (MSL), defines the level of moisture, when the soil begins to be too dry for the plants. The second value, Moisture Hard Limit (Hard Goal) (MHL), defines the level of moisture that would be harmful for the plants. The moisture can drop under the MSL for a short period of time, but must not drop under the MHL. Given these conditions, the AI-Planner can decide not to water and let the moisture drop under the MSL, if it would rain early enough. Or it can distribute a small amount of water to bridge time till rain occurs.

The planning algorithm is an action-based search tree algorithm with a loss function. The actions of the system are to water in different discrete amounts. A moisture level under the MSL and MHL decreases the score, also the actions have costs. The Moisture Score Loss Function (MSLF) defines the loss in score, caused by low moisture. For now, there are three sub functions for the MSLF implemented that define the curve of the costs between the MSL and MHL, a fix, linear and a quadratic function. Switching these functions has a huge impact of the irrigation behaviour. The quadratic function repressents the assumption that drying out near the MSL is less hazardous, but at a certain point, the MHL, it gets critical fast.

In each node of the tree, the current moisture (affected by the action, evaporation and rain) and the current score (effected by the current moisture and costs of the action) is calculated. The Planner searches for a plan (sequence of action for each hour), that maximize the resulting score.

Therefore, the planner is related to game theory and the algorithm can be viewed as a game tree, where the second player is the real world, with only one action – evaporate water. Within this point of view, it is a one player game.

The Planner periodically replans, so it can react to changing weather forecast predictions and false evaporation predictions. Therefore, the first element of the resulting plan is the action if and how much to distribute now. The AI-Planner recalculates the plans every 30 minutes, as it doubles

the timestep of the Planner, of one hour. So, it can react to changing weather conditions or false predictions. The total plan time interval is 24h, assuming that a longer period of time would not be useful due to the unpredictability of the weather and the inaccuracy of forecasts and evaporation prediction. Weather and garden states are nondeterministic, chaotic sytems.

The behavior of the Planner is defined by the following parameters, listed below, the values/settings of these parameters are solved by trial and error and were evaluated during the testing time. Every parameter can be set by command-line argument or openHAB to change the behavior and adapt it to the user's needs:

Actions Each action has cost/loss in score and a moisture gain. The costs are set to 0, if moisture is below MHL

Action	a0	a1	a2	a3
Watering Amount	0ml	600ml	1200ml	1800ml
Loss in score	0	-9	-17	-22
Moisture gain	0	5	11	16
Costs/pp in moisture gain	/	1.8	$1.\overline{54}$	1.375

MSL : Moisture Soft Limit, is set to 80%

MHL : Moisture Hard Limit, is set to 65%

scoreLoss_s : defines the maximum loss in the $MSLF_s$ function, is set to -4.5

scoreLoss_h : defines the loss of score, if moisture is below MHL, is set to -30

MSLF : Moisture-Score-Loss-Function, defines the loss of score, depending on the current moisture m.

$$MSLF(m) = \begin{cases} 0 & \text{, if } m > MSL \\ MSLF_s(m) & \text{, if } MSL > m > MHL \\ scoreLoss_h & \text{, if } m < MHL \end{cases} \tag{5.1}$$

The $MSLF_s$ function defines the loss, if $MSL > m > MHL$ The following types are implemented and can be selected: fix, linear and quadratic:

$$MSLF_s(m) = scoreLoss_s$$

$$MSLF_s(m) = \frac{scoreLoss_s}{MHL - MSL} \cdot (m - MSL)$$

$$MSLF_s(m) = \frac{scoreLoss_s}{(MHL - MSL)^2} \cdot (m - MSL)^2$$

5.2 Pseudocode

The algorithm of the AI-Planner as Pseudocode 5.1

Algorithm 5.1 Planning Algorithm

```
d = 24                                          // d is the length of planning time, here 24h
actions = {a0,a1,a2,a3}          // Actions with their moisture gain and costs as described in 5.1
procedure PLANNING(moist,hour,plan,score, bestScore)
    if hour > d then              // Termination Condition, as end of planning time is reached
        if moist > (MSL + MHL)/2 then                                          // Explanation*
            return plan, score, score as new bestScore
        else
            return plan, −∞, bestScore
        end if
    end if
    if score < bestScore then                    // Pruning, if score is less then bestScore
        return plan, −∞, bestScore
    end if
    score_best = None
    plan_best = None
    for all a ∈ actions do
        score_a ← CALCSCORE(moist, a, hour)
                                  // Calculates the new score, based on the action and the MSLF
        moist_a ← CALCMOIST(moist, a, hour)
             // Calculates the new moisture, based on the action and evaporation/rain prediction
        plan_a ← plan.append(a)
        plan_st, score_st, bestScore_st ← PLANNING(moist_a, hour+1, plan_a, score_a, bestScore)
                                                  // X_st is the relating return value of the sub tree
        if bestScore_st > bestScore then
            bestScore ← bestScore_st
        end if
        if score_st > score_best then
            score_best ← score_st
            plan_best ← plan_st
        end if
    end for
    return plan_best, score_best, bestScore
end procedure
```

Explanation:* If no minimum moisture level is set at the end of the planning time, the algorithm let the moisture dry out the last few hours. This happens because it does not receive the consequences of low moisture in that left over period of time, as the costs of a watering action exceeds the loss in score from the MSLF by a few steps. To clarify this situation: The loss, caused by a moisture level just above 65%, is near 4.5; the costs of the action with the smallest amount of water is 9. So, it is cheaper to have a moisture level of 65% in the two last steps than to water even the smallest amount of water.

Of course this is not desired, because it can have an effect backwards the path in the tree, even to the first element, that is the executed action. To at least maintain the moisture level between MSL and MHL is a good compromise as the Planner does not know how the conditions will be after planning time and it is also updated often enough during runtime to adapt to these conditions. Figure 4.6b shows this behavior, in the early state of the planner (where this plot is extracted) the additional condition was not implemented. The predicted moisture falls below 72.5% in the last hours of the plan, contrary to the 75% at the beginning.

5.3 Theoretical Evaluation of the Planning Algorithm

5.3.1 Correctness and Optimality

A plan defines a sequence of actions for each timestep, a full plan is sequence of actions $(a_0, .., a_d)$, where d is the max depth: The correctness in terms of finding the plan $(a_0..a_d)$, that maximizes the score, or better that minimizes the costs is trivial, if we do not use pruning. In that case, all resulting plans and their scores are calculated and compared, the plan with the best score is the optimal plan.

However pruning the unnecessary sub trees greatly improves calculation time. To prove the correctness of the algorithm with pruning, we have to look on the following equations.

$evapRain(n)$ is the evaporation and rain prediction for hour n, $moist(a_n)$ is the moisture gain of action a_n. The moisture at each timestep n, based on a plan is:

$$M(0) = \text{initial moisture}$$

$$M(n) = M(n-1) + evapRain(n) + moist(a_n)$$

Score Function:

$$S(\emptyset) = 0$$

$$S(a_i..a_j) = MSLF(M(i)) + costs(a_i) + .. + MSLF(M(j)) + costs(a_j), a_i..a_j \in Actions$$

The score of a plan $(a_0..a_d)$ is:

$$S((a_0..a_d)) = MSLF(M(0)) + costs(0) + .. + MSLF(M(d)) + costs(d)$$

As the MSLF and the actions only cause a negative change in score, the Score Function is monotonously decreasing

The best score B is:

$$B = max_{a_0..a_d}(S(a_0..a_d))$$

If we evaluate a partial path (a partial plan), to the depth $n-1$ with the solved actions $(a_0..a_{n-1})$ in the search tree with a potential new best score B' and look at a certain timestep n where we evaluate an action a_n, we get the following equation for this new potential best score:

$$B'_n = \underbrace{S(a_0..a_{n-1}) + MSLF(M(a_n)) + cost(a_n)}_{\text{current score}} + \underbrace{max_{a_{n+1}..a_d}(S(a_{n+1}..a_d))}_{\text{remaining sub tree with score } \leq 0}$$

We assume to have a current potential best score B_c from previous recursions. So, if the current score falls under B_c, the following sub tree of the node n cannot optimize B' anymore as the score function is monotonously decreasing. It is not necessary to calculate the path to the end and we can prune it. On the one hand, the previous part of the plan was not part of the best plan or the selected action was not part of the best plan. On the other hand, if $B' > B_c$, we go deeper into the tree, until we reach the max depth and get a new B_c, or we detect that no path results in a better score. At the end, the algorithm has had a look at each path that could give the best score, until it discovers the optimal plan, because $B \geq S(a_0..a_d) \forall a_0..a_d \in Actions$ and paths with $B' > B_c$ are not pruned and updates B_c. At the end applies $B_c = B$ and the plan that that had the score B_c is the optimal plan. If there are more plans with a score equal to the best score, the first is selected. So, the algorithm is correct, complete and optimal.

5.3.2 Analysis of Complexity and Calculation Time

Analysis of Complexity: As each plan has a limited length of depth d, the complexity in time and space is the same.

Best Case:

The best case is, if the algorithm finds the best score B on the first evaluated path in the tree. That is, if the moisture does not drop too low and the sum of all loss from the MSLF is less than the cost of the cheapest watering action. Then all other sub trees, with watering actions, are pruned directly. The complexity is $O(d)$, where d is the max. depth.

Worst Case:

In the worst case, every time the last tested watering action will return the optimal plan. So it will test all paths in the search tree. That would also be the complexity without pruning. The planner has to go through every node in the tree.

In this case the complexity is $O(|Actions|^d)$. The implemented Planner, with 4 watering classes as actions, has a complexity of $O(4^d)$.

Under real conditions, the evaporation is far less than the water gain of the smallest watering action. Hence, it does not water the plants at every timestep. Even on the hottest days of the runtime (over 33°C), it resulted in 1-2 times of watering during the planning time. Thus, the Planner is near the best case.

It is possible to improve the calculation time by sorting the excecution of actions. A heuristic for the selection of the next sub tree is implemented, by calculating the moisture of the next five hours (just evap. prediction and rain), if the moisture level is under MSL and the summed MSLF, exceeds the costs of an action with a higher watering amount, the order of actions is switched. The action that barly was exceeded, comes first and then alternating the next higer and lower. For example, if the loss in score by MSLF would be -11 in sum during the next 5 steps, than order of action execution would be $a1, a2, a0, a3$, as the costs of $a1 = -9$ and the costs of $a2 = -17$.

Calculation Time under Real Conditions:

The calculation time settles to an average of less than one second. For example, the calculation on a hot day, where the Planner has to water the maximum amount (last processed action in the first element of the plan), the calculation time on my notebook was 0.053 seconds on a AMD Ryzen 5 2500U CPU and throttled clock speed to 1.5GHZ

On the Raspberry PI 3, the calculation time of 3 days run-time (> 100 calculated plans) resulted in an avgerage time of 0.017 seconds and a maximum of 0.062 second. The total process, including gathering the weather forecast data, calculating the moisture prediction and planning is done in less than 5 seconds.

In comparison, without pruning, the calculation time exceeded 15 minutes on a planning time of 12h (instead of 24h).

5.4 Handling Rain – Implementation and Evaluation

To calculate the natural irrigation corresponding to precipitation/rain, the system uses the predicted precipitation intensity (mm/h) and the precipitation probability, gained from the data of the weather forecast provider. In the first step, the Planner classifies the precipitation into 5 classes. The intensity of precipitation is multiplied with the probability to rain. The weather forecast often seems to underestimate the probability (generally less than 15% - 30%). In order to take these cases into account, the minimal probability is raised to 0.5. In the second step, the Planner estimates the total change of the moisture level according to the evaporation prediction (from the ANN, in the following table as e_p) as well as this classification and adds the result to the current moisture, of the current timestep (see table 5.1).

$E_m(X) = precip_i * max(P(X = rain), 0.5) = precip_i * max(precip_p, 0.5)$

Index	$E_m(X)$	Calculation of the Planner
0	< 0.1	No expected gain in moisture $\rightarrow e_p$
1	>0.10	$e_p/2 + 0.2$, if $e_p < 0$, else $e_p + 0.2$
2	>0.4	+0.5, if $e_p < 0$, else $e_p + 0.5$
3	>0.8	+5
4	>2.5	+ 10

Table 5.1: Change in moisture caused by rain and the evaporation prediction (e_p)

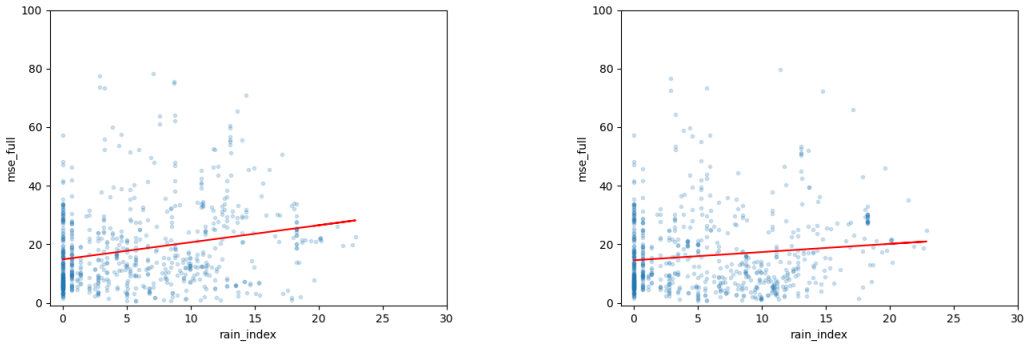
The parameters of the classification affect the estimation of impact due to rain. Thereby, the Planner will overestimate or underestimate the moisture level trend during time. To evaluate this, I used an evaluation quantity, the *rain_index*, and generated scatter plots with best fitting line.

$$rain\_index = \sum_{n=0}^{plan\ length} \ln\left(1 + classification\_index_n\right)$$

Figure 5.1 shows the MSE depending of the *rain_index* of the plan, Figure 5.2 shows the tendency of the deviation. In each case, the first plot has the actual used standard parameters as in table 5.1, the second has lower moisture gain values.

The plots show, that the actual system overestimates the impact of rain, the MSE and deviation could be optimized, as in the left plots. But as the system should prefer rain instead of watering, it is intended to overestimate the rain impact. And the overestimation does not have negative consequences to the plants; if the moisture level drops too low, the system recognizes that early enough as the AI-Planner is updating each 30 min and reevaluates new plans on the real conditions. Figure 5.3 shows that the moisture is always in a good state near MSL or above. The only outlier with approx. 50% moisture was a sensor failure.

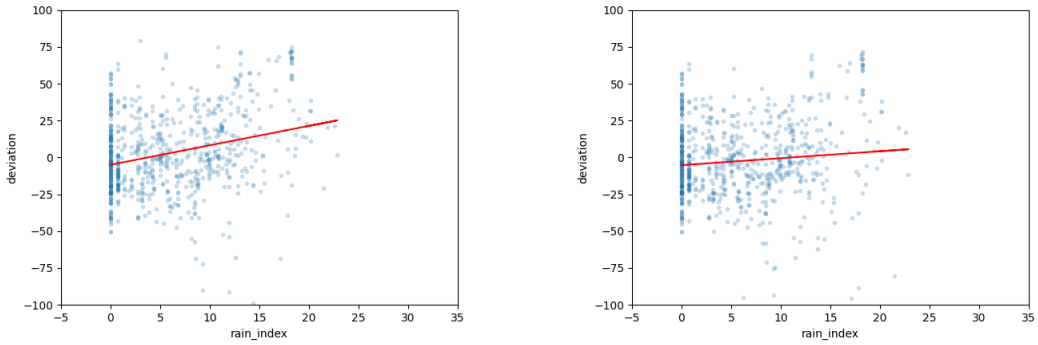
See Appendix A.2.1 Evaluation of Rain Handling for a short evaluation on three selected examples.



(a) Actual settings of the system

(b) Setting with lower values

Figure 5.1: MSE depending on *rain_index*



(a) Actual settings of the system

(b) Setting with lower values

Figure 5.2: Please note that in this figure the deviation or error is meant as summed absolute deviation of each hour during 24h of the plan, in this case it is a qualitative value, not a quantitative. So, a value of e.g. 20 does not mean the plan had an prediction error of 20 in any timestep.

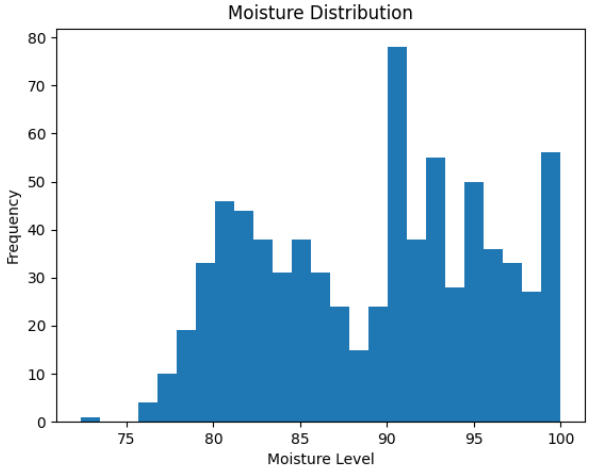


Figure 5.3: Moisture Distribution during runtime of ANN-Model “THLTD1”.
The explanation of the two seperated maxima is that the first peak is caused by dry time intervals, where the system holds the moister level near the MSL. The second is caused by longer rainy time intervals. The plot also proves that the Planner let the moisture level drop below the MSL, depending on the estimated further conditions.

5.5 Fail-Safe Methods

The AI-Planner requires a connection to openHAB and the weather forecast. In case of this prototype, openHAB, the Raspberry Pi Devices and Sensor System and the AI-Planner work on the same Raspberry PI, so the connection is guaranteed, as long as no component of the system crashes. But in case of a real integration to an existing smart home solution, the components and subsystems can be distributed. Therefor, a stable connection to the router and the internet is necessary.

To ensure that the garden is maintained, the planner has two stages to guarantee fail-safety. The first stage is an intrinsic: If the weather forecast provider is not reachable, it tries to reconnect three times with a short delay. If the provider is still not reachable, it chooses another. In this case, it is the old API version of tomorrow.io. This pattern could be continued with several providers, until one is reached. If the plan can be calculated and it is evaluated to irrigate or not, a timestamp as life sign is send to openHAB (and if needed the watering command).

If no provider is online or the system itself is not connected with the internet, or the system crashed, the Fallback System recognizes the missing or outdated timestamp. As the Fallback System is integrated into the Raspberry Pi Devices and Sensor System, it can act autonomous from all connections. As long as the timestamp is not older than one hour, as it is the double timespan of the planner's circle, the Fallback System is supressed to control the watering. Otherwise, if the planner wants to let drop the moisture level under the limit of the Fallback System, this plan will be overridden.

Depending on the setup of the smart home solution, it is possible to send a warning to the user if the timestamp is not updated. For example, openHAB offers a "Mail Binding" that provides the possibility to send an e-mail to the user.

The code would look like Listing 5.1

Listing 5.1 E-Mail Notification via openHAB

```
rule "Cron 30 Minutes"
when
    Time cron "0 0/30 * * * ?"
Then
 if (garden1_bu_ai_timestamp.state < now().getMillis()/1000)
 getActions("mail", "mail:smtp:gmail"). sendMail("mail@example.com", "Failure", "AI-Planner
Failed")
End
```

The same patter can be used for other warnings. For example, if water level of a tank is low, fail states of sensors are detect and so on.

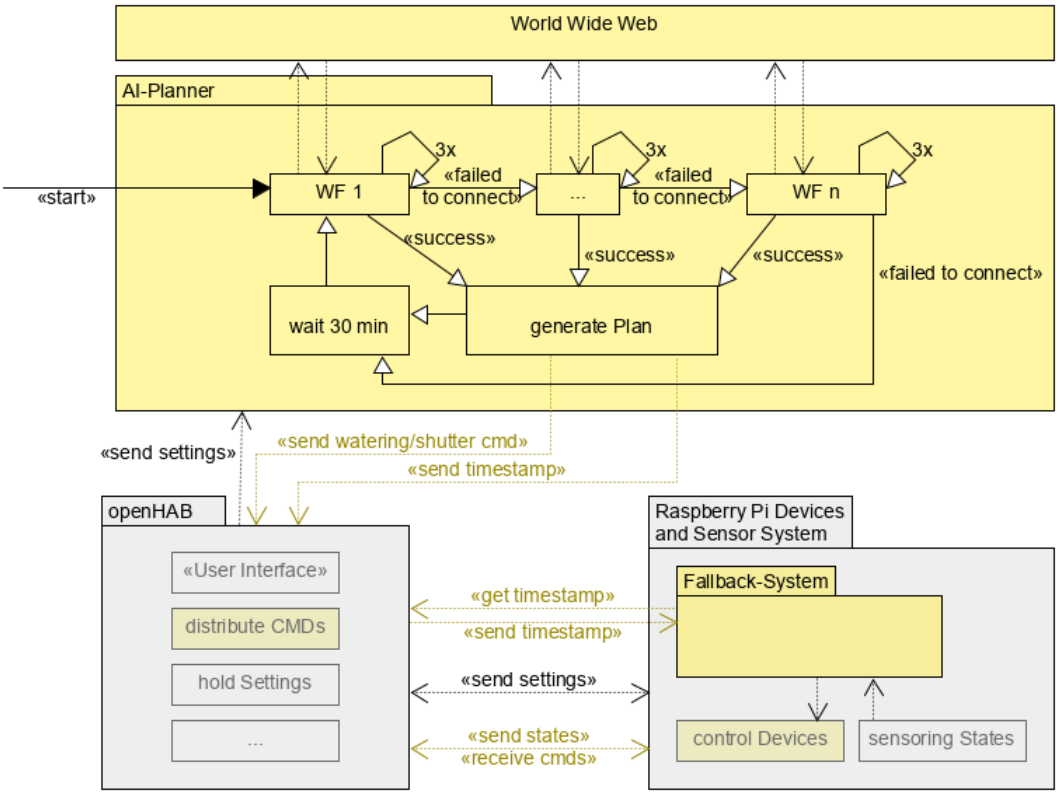


Figure 5.4: Fail-Safe System

5.6 Repressing Irrigation due to Rain on an Example

For proof of concept, the Planner behaves as expected and represses watering if rain is predicted, a dump of a weather forecast that did not predict rain is manipulated. So, with the manipulation: 6 steps/hours after start time, rain is insert to the dumped weather forecast data. The parameters for the Planner are the same as above, the single plans differ by the initial moisture.

The finally implemented Planner stores some more information in the returning plan than only the resulting list of actions. This is made, to retrace the decisions and to compare plans with different settings.

Explanation of the entries in the calculated plan:

[Y]XX	: Y -> Action [a1,a2,a3], on Action a0 (no watering), one of the other cases is printed
m:[A]s:[B]	: A is the predicted moisture (after this step), B: current score
m:[A]s:[B]loss:[C]rain:[D]evap:[E]	printed if evaporation > 0 : A, B as above, C is the moisture change, D is the rain classification, E is the predicted evaporation from the ANN

Plan (initial moisture = 80):
['m:79s:0', 'm:79s:0', 'm:79s:0', 'm:79s:0', 'm:79s:0loss:0.62rain:0evap:0.62', 'm:80s:0loss:0.83rain:0evap:0.83', 'm:90s:0loss:1e+01rain:4evap:0.77', 'm:100s:0loss:1e+01rain:4evap:1.0', 'm:99s:0', 'm:99s:0', 'm:98s:0', 'm:98s:0', 'm:98s:0', 'm:97s:0', 'm:97s:0', 'm:97s:0', 'm:96s:0', 'm:96s:0', 'm:95s:0', 'm:94s:0', 'm:93s:0', 'm:92s:0', 'm:92s:0', 'm:92s:0']
$\rightarrow a_0 = a0 \rightarrow$ Does not water

Plan (initial moisture = 72):
['m:71s:-1', 'm:70s:-2', 'm:70s:-4', 'm:70s:-6', 'm:70s:-8loss:0.27rain:0evap:0.27', 'm:71s:-9loss:1.3rain:0evap:1.3', 'm:81s:-11loss:1e+01rain:4evap:0.75', (...)]
$\rightarrow a_0 = a0 \rightarrow$ Does not water

Plan (initial moisture = 71)
['1 XX', 'm:74s:-11', 'm:74s:-11', 'm:74s:-12', 'm:74s:-12loss:0.27rain:0evap:0.27', 'm:75s:-13loss:1.3rain:0evap:1.3', 'm:85s:-13loss:1e+01rain:4evap:0.75', (...)]
$\rightarrow a_0 = a1 \rightarrow$ Waters 600ml

Plan (initial moisture = 67)
['2 XX', 'm:76s:-20', 'm:76s:-20', 'm:76s:-20', 'm:76s:-21loss:0.27rain:0evap:0.27', 'm:77s:-21loss:1.3rain:0evap:1.3', 'm:87s:-21loss:1e+01rain:4evap:0.75', (...)']
$\rightarrow a_0 = a2 \rightarrow$ Waters 1,200ml

For comparison, the same weather forecast, but without manipulation, so it does not predict the rain in 6h:

Plan (initial moisture = 77)
['2 XX', 'm:86s:-17', 'm:86s:-17', 'm:86s:-17', 'm:86s:-17loss:0.27rain:0evap:0.27', 'm:87s:-17loss:1.3rain:0evap:1.3', 'm:88s:-17loss:0.75rain:0evap:0.75', 'm:89s:-17loss:0.56rain:0evap:0.56', 'm:89s:-17',(...)]
$\rightarrow a_0 = a2 \rightarrow$ Waters 1,200ml

The Planner let the moisture level drop far beneath the MSL of 80%, if rain is predicted but also always keeps the moisture level above the unhealthy MHL limit of 65%. The Planner tries to bridge the time by watering a small amount, if possible and needed. In this example, the lowest moisture level is 70% with the initial moisture of 72%. Below that initial moisture, the Planner tries to raise the moisture level by watering a low amount (600ml). In comparison, without the manipulation, the system waters far earlier.

5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models

To prove, if the AI-Planner has the desired behaviour under real conditions, the generated plans are evaluated by a statistical analyse.

As measurand to evaluate the accuracy and behavior of the Planner, the MSE and the SE as $SE = \sqrt{MSE}$ is used. For the plots, the Median (Q2) and 25%/75% quartiles (Q1, Q3) of the MSE are used and to select representative curves, because it is robust against outliers. In all figures, "r_i" is the $rain\_index$ as in chap. 5.4 Handling Rain – Implementation and Evaluation, the higher this value, the higher and more frequently rain is predicted. Plots can show filtered data, for example filtered plans with high $rain\_index$ (for example as sort of an outlier elimination). Sometimes plans are filtered if they are in a time interval, when it was watered in reality (especially on plots of model "2WM"). If data were filtered, it is mentioned in the plot; also if outliers above 1.5 times the interquartile range (1.5 IQR) are eliminated.

Every performed circle of the planner (30 min), the planner writes a log and dumps the weather forecast data. The Planner logs all plans and some additional data (initial moisture, rain classification, predicted evaporation and used ANN-Model). Based on this data, the moisture prediction of the plan is reconstructed and compared to the real measurements. So, it is possible to calculate the MSE of each plan and each timestep in the plan. As the system has been running for several months, there are over 1000 single plans to analyse. Actually, the analysis is only done for the new kind of models ("THLTD" -Type). More details of the different model types can be seen in chap. 4 ANN-Based Evaporation Prediction.

In addition, it is possible to recalculate plans retrospectively, based on the logged weather forecast dumps. This way, it is possible to generate alternative plans of the past with a different ANN-Model on previous time intervals and compare their behavior. This is done for the model "2WM" as it was generated near to the end of the long-term test.

To compare this new model to the model that was originally in action, it is needed to filter out all plans, when the original systen had watered. Because the plans would differ as the actual performed actions doesn't match the expected new, retrospectively plans and raises the prediction error. This can be seen in Figure 5.5. This effect can statistically be proven, if the filtered and not filtered data are compared. The filtering has nearly no effect on the original model, but shows an improvement to the MSE of the different model, see Figure 5.6. In addition, the fact that there is nearly no effect on "THLTD1" compared with its own plans, shows that the predicted gain from watering (as parameter of the actions) are well set. If there was a huge difference, the moisture gain parameter would need to be adjusted, so the plots are helpful tools to check the settings.

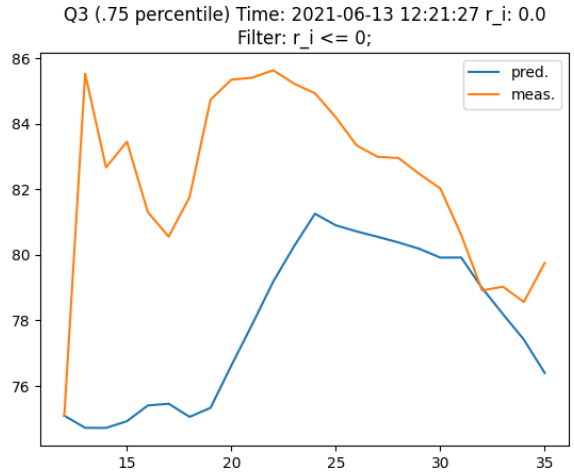


Figure 5.5: Example for high deviation in a retropspective plan, as watering in this plan differs from real watering events.

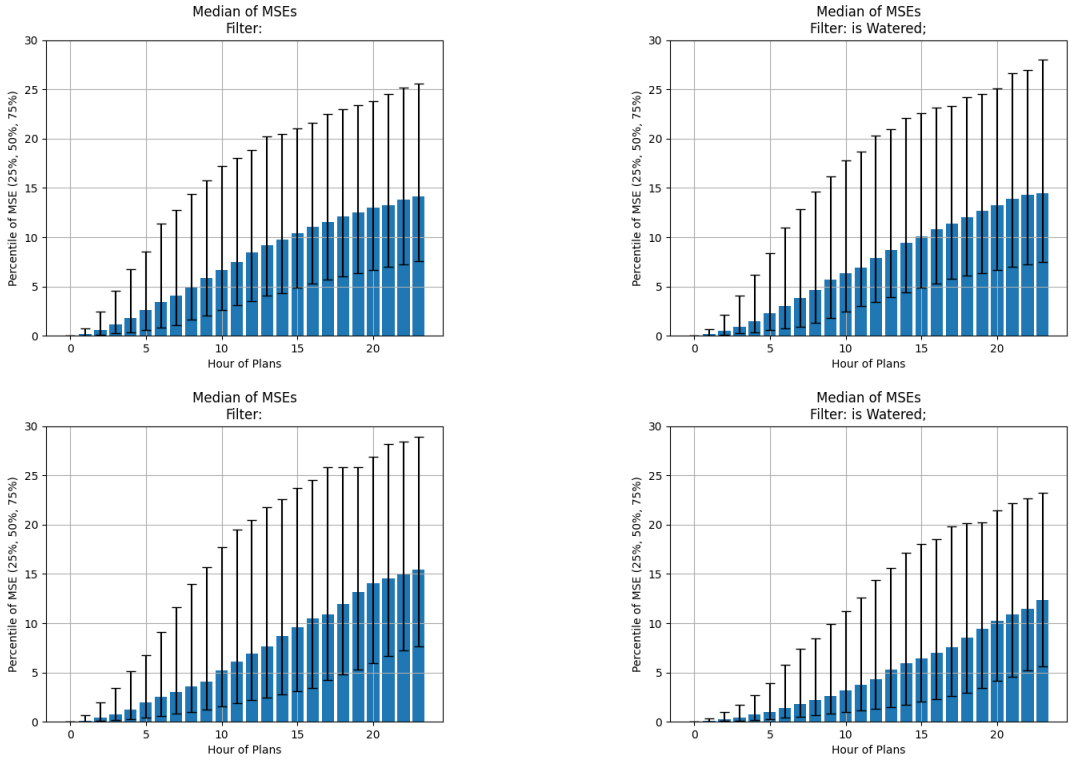


Figure 5.6: Substantiation to filter data of retropspectively generated plans. Filtering on the origin plans has nearly no effect, contrary to retropspective plans that used a different model or settings. (Upper Row: Original Plans; Lower Row: Retrop. Plans)

To explain why the new model “2WM” performs so much better than “THLTD1”, even as it had far less training data (measurements of two weeks vs. > 4 weeks), “THLTD1” was selected and generated manually. It was the best of 10 generated models. Selecting and generating modeks manually needs much luck to find a good model. The tools to semi-automize this process was implemented later. Using these tools, “2WM” was a result of 400 random generated models and had the less MSE (on the trainings data). See chapter 4 ANN-Based Evaporation Prediction and chapter 5.11 Additional Features and Tools.

Figure 5.7 shows the frequency distribution of the MSEs of all plans (after 5h, 10h and the full 24h of planning time). The bias in mean and median shows that it is more likely to have better predictions and some few (but strong) outliers. Such outliers are mainly caused by false weather forecasts. See Appendix A.2.1 Evaluation of Rain Handling, in the first example, the previous plan had an MSEs of 244, whereas the following plan, 30 minutes later (and a complete updated different weather forecast), had an MSEs of 11.2.

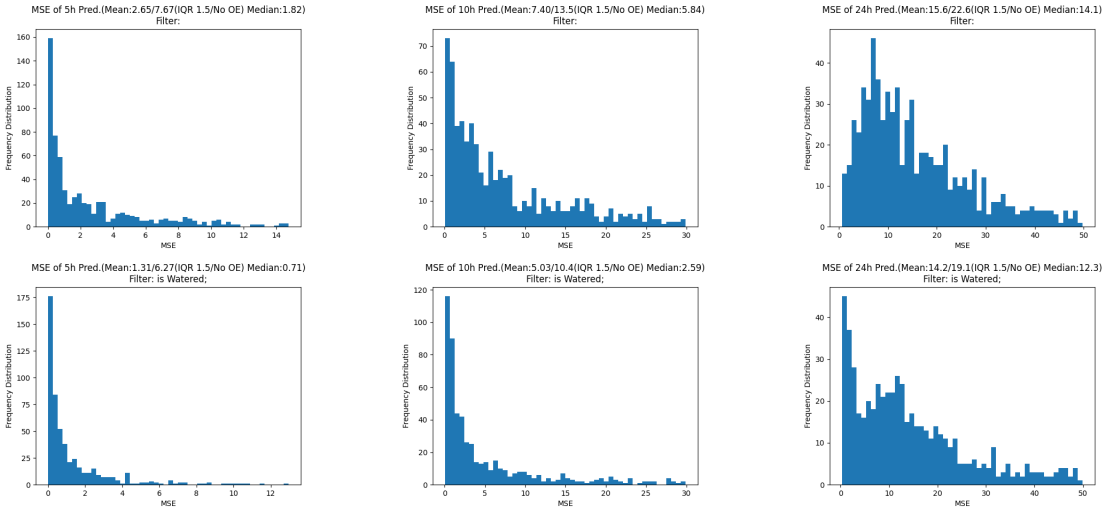
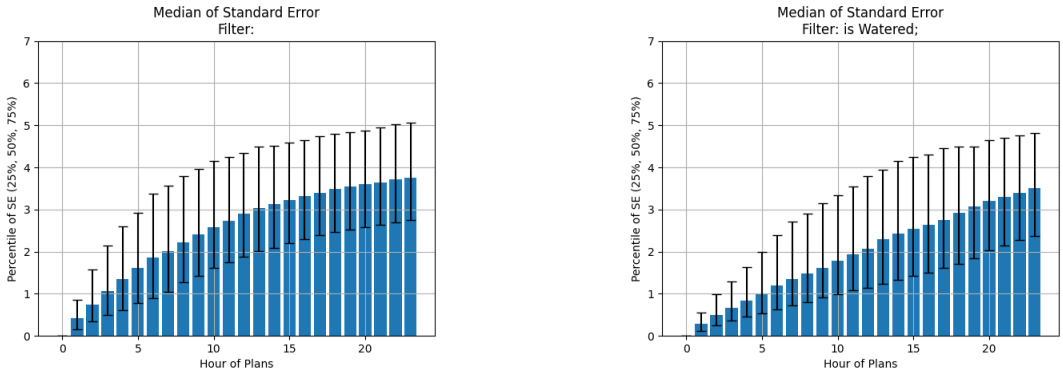


Figure 5.7: MSE distribution of the plans after 5, 10 and 24 hour of planing time.
Upper Row: “THLTD1”; Lower Row: “2WM”

To evaluate the accuracy of the prediction, the SE is more meaningful than the MSE (Figure 5.6). Because it is an estimate of the standard deviation and demonstrates how much the prediction is expected to differ from the real measurements. The plots show the trend of the SE over planning time. Upper and lower borders of the error bars are the 25% (Q1) and 75% (Q2) percentiles, see Figure 5.8. It can be noted, that the accuracy is 3.5pp in median, 75% are better than 4.8pp. If rainy days are filtered, the accuracy rises to 1.2/1.8/2.8 pp (Q1/Q2/Q3), see Figure 5.10.



(a) "THLTD1" **(b)** "2WM"

Figure 5.8: SE of the plans over planning time.

Figure 5.9 shows the predicted moisture curve (blue) and the measured curve (yellow) of the plans that are the quartiles Q1, Q2 and Q3 of all plans. These plots show that the system predicts the trend of the real moisture progression very well and also can act on the rain prediction of the weather forecast. Sometimes the measured curve is missed by a few hours, but this is a minor problem, as the plans are recalculated often enough and it adapts to the real conditions. And the weather forecast becomes more precise over time.

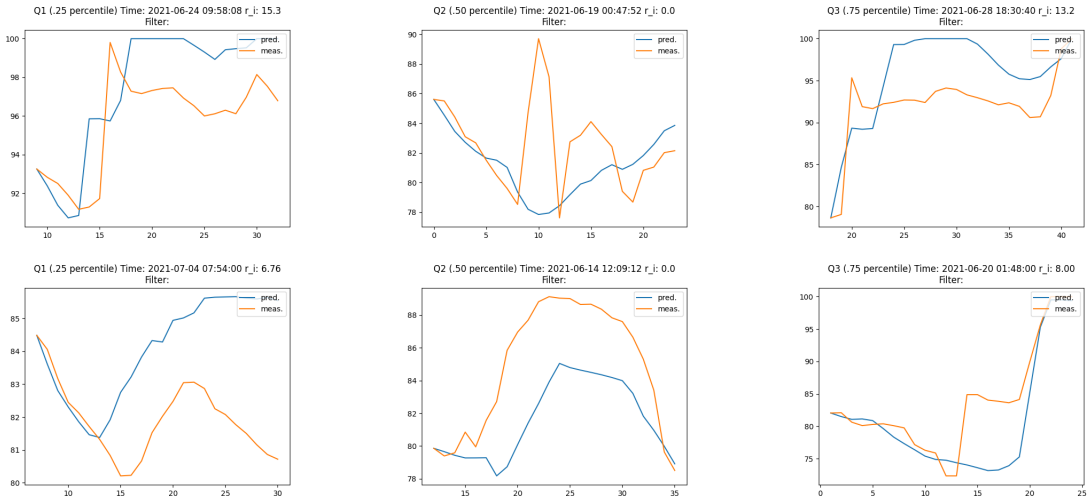


Figure 5.9: Q1, Q2 and Q3 Curves. Upper Row: "THLTD1"; Lower Row: "2WM".
The curve of Q2 of "THLtD1" was during a time intervall, where i had trouble with the sensor and had to recalibrate it a few times, that is the explantation of the strange real measurements.
Q3 of "2WM" shows at 12:00 that the original planner watered, whereas it did not expect that. Though, the weather forecast and the real rain event at 20:00 fit very well.

To check the accuracy of the evaporation prediction itself, days for which rain was predicted, are filtered. So, the inaccuracy of the weather forecast to predict precipitation is cancelled out. In doing so, this leads to verry accurate predictions of “2WM”. In conclusion, as the evaporation prediction displays this high level of accuracy, the accuracy of the planner is dominated by the weather forecast (see Figure 5.10). It can be noted, that the accuracy of the planner itself is in range of 2 percentage points off the measurement after 24 hours. The inaccuracy of the weather forecast is doubling the error.

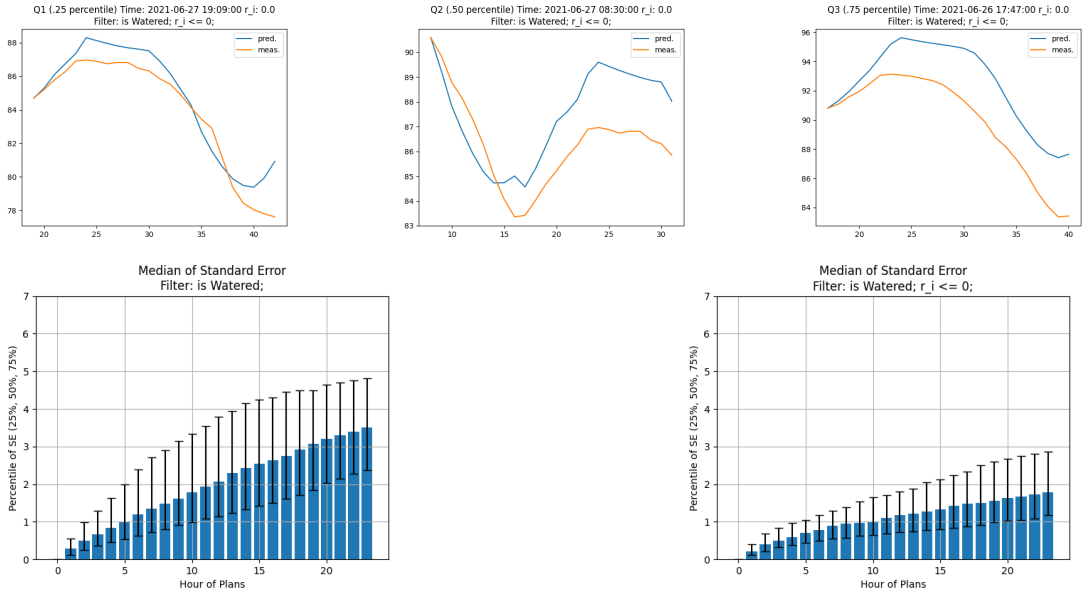
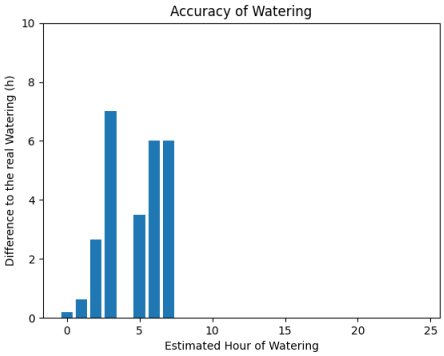
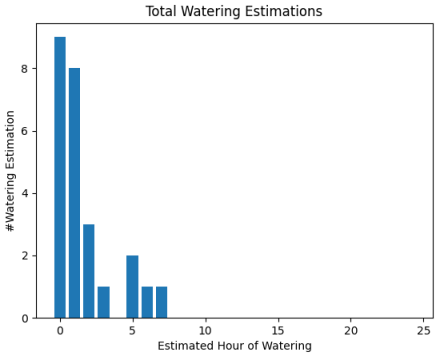


Figure 5.10: Weather Forecast dominates the Accuracy: As “2WM” is very accurate in predicting the evaporation, the SE drops from about 3.5 to 1.8, if rainy days are filtered out. In comparison: “THLTD1” does not show such a high effect, as it is overlapsed by its own inaccuracy, so the SE just drops from 3.8 to 3.1

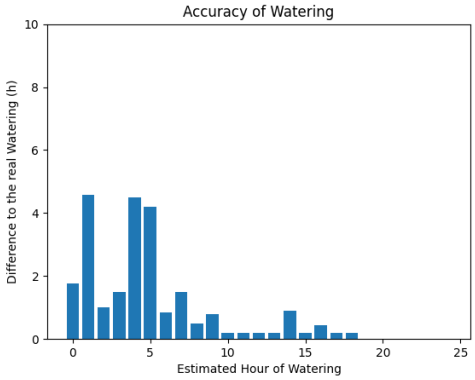
Also, very meaningful is Figure 5.11. It shows the prediction, when following plans will water and how accurate this estimation was (time distance to the real watering event). If the ANN-Model would predict the evaporation absolute perfect and if the weather forecast would be precise, it should be a constant curve. The planner would exactly know to water in 24 hours. The next circle would know it will be watered in 23 hours and so on. Interesting is, that model “2WM” could better predict when model “THLTD1” would water than “THLTD1” itself, because it is far more accurate (see Figure 5.11). To eliminate confusion on why “2WM” watered 50 times on timestep 0, there is an easy explanation: If “2WM” wanted to water, but it did not happen in reality, it wants to water during the next timestep over and over again, until the original planner has watered in reality or the moisture is raised by other effects (rain/capillary remoisturizing). Each plan that this happened in, differs greatly from the real measurements, as the plan succeeds with the higher moisture level. This is also the explantation, why filtering out such plans, when it was watered, is legit, as it corrects the MSE and SE.



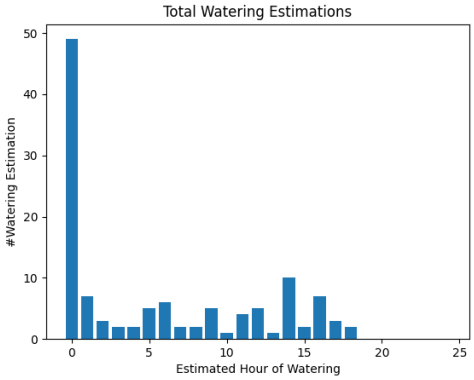
(a) "THLTD1": Accuracy



(b) "THLTD1": Number of estimated Watering Events



(c) "2WM": Accuracy



(d) "2WM": Number of estimated Watering Events

Figure 5.11: Accuracy to predict Watering Events

5.8 Analysis of Water Consumption

To evaluate the system's behavior under real conditions, we have to take a look at the diagram in chapter 5.4 and the log files. Diagram 5.3 in chapter 5.4 shows the frequency distribution of soil moisture during the runtime. As we can see, the system let drop the moisture below the MSL as expected. This is caused by two effects. The first is expected rain, the second effect is the remoisturizing effect from lower soil layers in the evening. Including both is desired and the system acted correct. The entries of the log files of the Planner provide the information, if a plan differs from a plan that does not integrate expected rain. So, in each circle of planning, two plans are actually calculated and compared. The output in the log looks like this:

```
Changed␣watering␣behavior␣because␣of␣rain:␣True;␣with␣rain:␣'m:79s:0'␣without␣rain:␣'1␣XX'
```

Hence, it is easy to check how often the system did not water, due to rain, during the time interval, the better prediction model type "THLTD" was in use. This is from 16th June 2021 until now (09th July 2021). In total, it repressed watering three times, but had to correct the decision once and watered 1:30h later. The system watered 9 times during that time interval. So, the system did not water in two out of 11 cases that is a rate of approx. 18%. And the prediction of the futur moisture level often makes the system just water a small amount.

Timestamps of this events:

```
[2021-06-28␣19:31:34]
[2021-07-05␣08:47:06]
[2021-07-05␣18:25:51]␣//but␣watered␣2␣h␣later
```

The system watered 12 liter at all in the time interval from 16th June 2021 to 09th July 2021. The control group (Garden Part 2), watered by the Fallback System, was watered with 14 liter. Poorly there were some sensor issues from 19th June 2021 to 21th June 2021 (see Figure 5.3), the sensor measurements were far too low, so the system watered falsely three times in succesion. Hance, it would have watered even less than the 12 liter.

In total, I filled 27 liter in the tank of Garden Part 1 (AI-Planner) and 36 liter in Garden Part 2. Garden Part 3 was manually watered with 24 liter. So, the water consumption of the intelligent system was reduced by 25% against a simple "Water at Moisture X"-System and was close to my own, manual watering decision.

5.9 Evaluation of the Shutter behavior

The weather forecast is also used to close the shutter in case of heavy rain and hail. As the forecast is not as accurate in local events, the shutter is closing one hour before and stays closed for an additional hour after heavy rain/hail is predicted.

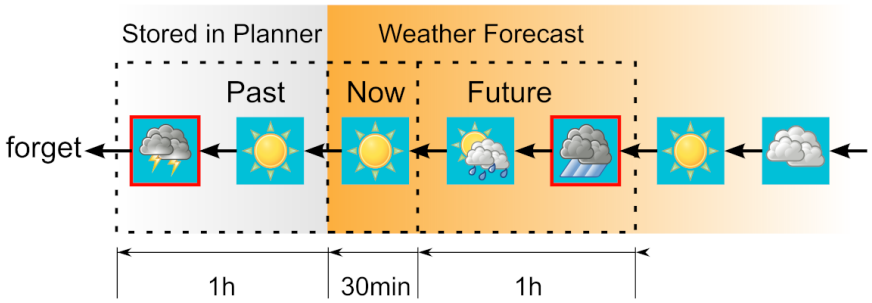


Figure 5.12: [1] Sample Rate of the forecast is 30min, holding the last two predictions (1h) and looking 1h into the future should eliminate the issue of inaccuracy of the highly located predictions. If there is a hazardous event predicted, it remains closed 2:30h.

Three conditions can lead to closing the shutter:

- Hail/Ice Pellets: Always close
- Heavy Rain (> 8mm/h): Just close if moisture is > MSL, because the planner might expect watering from this rain, as it is better to use this rain, instead of watering when the shutter is closed.
- High Rain (> 4mm/h): Only closes if moisture is > 90%.
- Medium rain (< 4mm/h rain): System does not close the shutter.

To evaluate the behavior of the shutter, the timestamps in the log, when the shutter was closed, are compared to a weather history[2] (location: Stuttgart Airport, less than 4 km distance to the system), as the ASHGA has no sensors for the precipitation. Unfortunatly, all the weather history providers, I could find, just had a summary of the whole day instead an hourly history. So, the data cannot display whether there was a high peak. However, the data gives an indication of high/low precipitation (see 5.13).

On 24th June 2021, closing the shutter was apparently wrong as the weather history shows low precipitation. To proof the correctness: The moisture increased >8pp in less than 30 min and indicates heavy (but short) rain. The shutter closed correctly short after the start of rain. This must have happened between 16:00 and 17:00 (see the following extracts of the log of the measured moisture that day).

As we can see, on all days with high precipitation, the shutter closed correctly.

[2]Got the data for the diagram from https://www.wetterkontor.de/de/wetter/deutschland/rueckblick.asp?id=195 [Accessed on 2021-07-31]

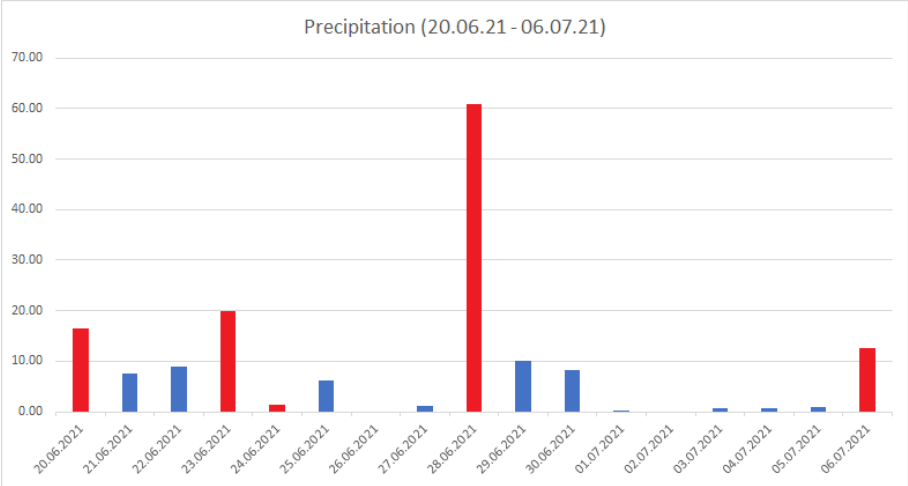


Figure 5.13: Weather History; Days, the shutter was closed at least once, are red marked

Log of the measured moisture:

```
1624539600 Thu␣Jun␣24␣15:00:00␣CEST␣2021 91.30
1624543200 Thu␣Jun␣24␣16:00:00␣CEST␣2021 91.73
1624546800 Thu␣Jun␣24␣17:00:00␣CEST␣2021 99.80
```

Log of the Raspberry Pi Devices and Sensor System:

```
[2021-06-20␣20:37:23]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-06-20␣22:08:43]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}
[2021-06-20␣22:39:11]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-06-20␣23:40:11]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}

[2021-06-23␣19:45:43]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-06-23␣20:46:36]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}

[2021-06-24␣16:33:57]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-06-24␣17:34:50]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}

[2021-06-28␣21:03:12]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-06-29␣00:05:56]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}

[2021-07-06␣21:24:09]␣TcpServer:␣Received␣msg:{␣"changeShutter":"CLOSE"␣}
[2021-07-06␣22:25:04]␣TcpServer:␣Received␣msg:{␣"changeShutter":"OPEN"␣}
```

5.10 Conclusion

In conclusion, the Planner performs the desired behavior. However, it is recommended to find ideal settings by trial and error and evaluating the behavior. To compare the behavior on different settings and different ANN models, there are some helpful tools implemented, see Additional Features and Tools 5.11.

Because the AI-Planner uses the data, provided from the smart home system, it is independently from the way the data are measured and full compatible to any smart home system that offers an API, likely to openHABs REST-API. The system, is integrable and deployable to the most smart home solutions, at the current state.

A weak point of the system is the inaccuracy of the weather forecast, especially predicting highly localized precipitation events.[Her19] Appendix A.2.1 Evaluation of Rain Handling outlines a closer look at specific events, where the weather forecast failed. The weather forecast can change completely in less than one hour with the next update. Plans before the update can result in unreasonable decisions as the data base does not match the real weather conditions. However, the Planner works precisely enough to generate reasonable plans for normal weather conditions and at least has an educated guess of the following precipitation. This year, meaning 2021, concessions for bad weather forcasts have to be made. On April, the World Meteorological Organization (WMO) reported that the decrease of air traffic due to corona, has a huge impact to the weather forecast accuracy as in-flight measurements are part of the global observing system.[Org20]

Despite these problems, using the rain prediction, the ability to predict the moisture level based on the weather forecast and learning the capillary moisturization, gives a huge benefit compared to simpler irrigation systems. Knowing smaller amounts of water will be sufficient or it is not needed to water at all, saved in total over 25% water during the long-term experiment.

For now, the moisture gain in percentage points of each action by the watering amount from these actions, has to be observed and set. I am certain, it is possible to automize this procedure. A self-adapting solution can be to set a rough estimation of the action's moisture gain, the system can evaluate and refine the predicted gain after each watering.

The behavior of the planner can be adapted to the needs of the plants. For example, setting the MSL and MHL higher or lower will keep the moisture level in a desired range. Setting the costs of the actions, the loss by moisture ($scoreLoss_s$) or choosing another $MLSF_s$, changes the system's behavior in terms of how conservative the system reacts to rain, as the costs of drying out exceeds the costs of watering sooner or later.

It would be possible to simplify these inputs for the user. As an idea: Using two simple sliders to change the behavior, in which:

- Slider 1 sets the desired moisture interval by setting MSL and MHL ,
- Slider 2 changes the ratio of $scoreLoss_s$ and action costs to define a more agressive or conservative irrigation behavior.

Scoring the actions and states of the garden, offers the option to refine the behavior of the system. The system can be extended by additional constraints. For example, avoiding watering in full sunshine by raising the actions costs, based on sunshine intensity (see Apendix A.3 Modify the

Action Costs). It is a myth that the leaves get burned through intense sunshine. However, part of the additional water will evaporate before entering the soil, so it makes sense to implement such a routine.[Jua]

Figure 5.14: Using the script A.1 in Appendix A.3 Modify the Action Costs. The system watered at 21:00 and 6:00, as the solar radiation was below 30,000lx. Comparing with the other plots in this chapter: Without this script, the system often waters around 10:00 to 14:00, as this is the time, the moisture level usally falls below the MSL

5.11 Additional Features and Tools

Knowing about the actual behavior of the system, supports the user to find the best settings for the system to work in the desired way. To analyze the behavior and simplify the training and selection precess of the ANN model, helpful tools are implemented.

There are five tools with different tasks:

- The User Interface in openHAB, to offer an easy way to set the parameters of the AI-Planner and Fallback-System.
- Sensor Data Gathering Tool: Loads the measured data from openHAB's data base and stores it as csv-files.
- Planner Evaluation Tool: loads the log files of a planner. This can be the original logs from the running system or retrospectively generated logs from the Retro Log Generator. It generates the plots shown in the chapter 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models.
- Retro Log Generator: Based on the weather forecast dumps, it generates simulated log files of the new planner (new settings, new model).
- Model Generator: Simplifies to generate new models, selects and saves the best couple of models.

Using all tools simplifies the process of updating the current model by a new, better one. The process works as following:

1. All actual logs, measurements, and weather data dumps are gathered.
2. The best few of many generated and trained models are selected (4-6 hours for about 400 models).

3. The alternative log files of the plans, resulting from the selected best models, are generated (1h per model on 1000 log entries).
4. Generate the plots to analyse the behavior, select the best fitting model and use this from now. This is needed, because models that are good on the training data (just the measurements of the garden), do not have to be the best model for the forecast data.
5. Redo the process, after some time to adapt to the growth of the plants or better adapt to the new season. A model that is only trained with data from the conditions in spring, maybe fails in summer, as it only trained low temperatures.

It would be possible to create a fully automized pipelining of this sequence of steps, if the Planner Evaluation Tool is modified to select the best model automatically. But because this process will take, all in all, at least about one day of calculation time on a "Intel Core I7-4790" @3.6GHZ, it is not practical to run this on the Raspberry Pi or a low-performance smart home server. In addition, it is reasonable to check the resulting best models manually until better automized evaluation methods are developed. Therefore, a simple analysis of the MSE seems not be sufficient, an intelligent curve analysis method is required. For example, the behvior of the "THL" model shows the problem, as it faked a better MSE through overestimating the watering events (see figure 4.5).

5.11.1 Short User Manual and Description of the Tools

Setting up the AI-Planner The settings of the AI-Planner can be changed in two ways:
By command-line parameters on start, or during runtime by the UI that openHAB provides.

Listing 5.2 Calling the AI-Planner

```
>python3 ai_planner.py
  //optional parameters
  msl       =[float: the Moisture Soft Limit]
  mhl           =[float: the Moisture Hard Limit]
  mslf_s        =["hard"/"linear"/"quad": defines the score_loss_s function]
  score_loss_s  =[float: parameter for score_loss_s (max(mslf_s(m))=score_loss_s)]
  score_loss_h  =[float: defines loss in score below the mhl]
  action_costs  =[[a0_c,a1_c,a2_c,a3_c,a4_c]]: sets the costs of each action]
  action_moist  =[[a0_m,a1_m,a2_m,a3_m,a4_m]]: sets the moisture gain of each action]
  rain_class_values=[[a,b,c,d,e]]: mm/h < a --> Index 0 of the rainclassification]
  rain_class_gain  =[[g_b,g_c,g_d,g_e]]: gain of each classification]
```

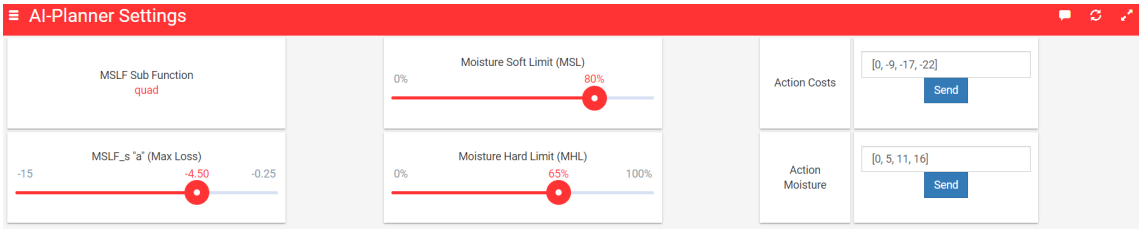


Figure 5.15: AI-Planner UI, openHAB

Sensor Data Gathering Tool The tool "*get_csv_hour.py*" updates the saved csv files with the last measurements, stored in openHAB's Round Robin Database (RRD). This should be done periodically, because the RRD just saves the one hour sampled data about one month, after that, the files are overwritten. In this way, all measurments are gathered over time and can be used to train the Models. A possibility to do this automatically, is to use a CronJob.

Listing 5.3 Calling get_csv_hour.py; get Measurements as CSV

```
>python3 get_csv_hour.py
/*
  gives:  measurements of the sensors as csv file:
        gp_air_hum_updated_hour.txt; gp_light_aktLux_updated_hour.txt;
        gp_soil_hum_updated_hour.txt; gp_Temperature_updated_hour.txt
*/
```

Planner Evaluation Tool Generates the plots, as in chap. 5.7 Statistical Analysis of the Planer and ANN-Based Evaporation Prediction Models, to evaluate the model.

Listing 5.4 Calling planner_eval_plots.py; the Planner Evaluation Tool

```
>python3 planner_eval_plots.py load=[planner log w/o file-ending]
  //optional parameters:
  start=[interger: only eval. plans that starts at [X:00h]-[X+1:59h]]
  r_i=[float: filter plans with rain_index > r_i]
  waterf=[true, false: filters plans, within the real system watered]
/*
  needs:  the (retrop.) planner's log file (log_watering(_XX).txt)
      log file from the Raspberry Pi Devices and Sensor System found in /gp_rasp/log.txt
  gives:  plots if the statistic analysis
*/
```

Retro Log Generator Generates a log file of alternative planner settings/ANN model. With the Planner Evaluation Tool, it is possible to check how a new model, or different settings, effect the behavior of the AI-Planner.

Listing 5.5 Calling retro_log_generator.py; generates retrop. log files

```
>python3 retro_log_generator loadall=[folder to weather forecasts] model=[m_1];...;[m_n]
  //optional parameters: same as ai_planner.py
/*
  needs:  Log file from the Raspberry Pi Devices and Sensor System found in /gp_rasp/log.txt
    Dumps of the weather forecast found in /gp_rasp/log.txt ai_planner/dumps
    ANN-Model
  gives:  retrop. log files of the AI-Planner
*/
```

Model Generator Trains new models. With this tool, it is easy to generate a whole bunch of models and save a preselection of the models, that performs best on the training/test data. Till now a [5,33,33,18, 5, 1] shape, with a leakyReLu and sigmoid function in the first layers and linear functions in the deeper, seem to performe best. With the "generateRandomModels" methods it is easy to generate randomized variations of models (layers and activation functions), if there is a better performing model type as the mentioned, it might will be found in this way. Also, it is the first step to optimize the generator, by implementing a genetic algorithm.

A good feature is the "*time_bann_list*", the user can insert time intervals, that are filtered from the training (and test) data. This is useful, to filter events where the measurements were wrong (sensor failure, other issues).

Listing 5.6 Model Generator; extract of the user relevant code section

```
>python3 model_generator.py
/*
  needs:  measurements as csv file from "get_csv_hour.py"
        gp_air_hum_updated_hour.txt; gp_light_aktLux_updated_hour.txt;
        gp_soil_hum_updated_hour.txt; gp_Temperature_updated_hour.txt
      log file from the Raspberry Pi Devices and Sensor System found in /gp_rasp/log.txt
  gives:  trained ANN-Models

  interessting methods:
    generateXModels(input_data, output_data, model_type,
      [shape], [a. functions],
      [# of trained models of this type], [# of training epochs])
    generateRandomModels(input_data, output_data, model_type,
      [# of variations], [# of trained models of each variation], epochs_a)
*/

#-------------------------- Generate and define Models here in the Code------
epochs_a = 4000 #number of training epochs (4000 is save but can be less)

# sort out this dates from the input data, because something went wrong during
  the measurements.
  E.g. issues with some sensors, -> prevents it from learning stupid behavior
time_bann_list=[[datetime(2021, 6, 17),datetime(2021, 6, 20,16)] #example to ignore
time_bann_list.append([datetime(2021, 4, 19, 12),datetime(2021, 5, 8, 5)]) #e.g. sensor
failure

#Standard Model description, that fitted best till now
#defines the type of the models
model_type = "temp,hum,lux,time,d_temp"
# defines the shape of the ANN Model, first entry is the input shape,
    the others are the deep layers and their shape, last is the output
model_input_shape = [5,33,33,18, 5, 1]
#activation functions of the layers, None = f(x)=x
model_activation = ["leaky_relu","sigmoid",None,None,None]

#load training data
input_data, output_data = getData(model_type, time_bann_list)
#use something like this, to load data for different types of models
#input_data_thl, output_data_thl = getData("temp,hum,lux", time_bann_list)

#generate Standard Model
models_set = models_set + [generateXModels(input_data, output_data, model_type,
    model_input_shape,model_activation, 3, epochs_a)]
#generate some more manually set Models (For example add manually some of the
    best of the last random models)
models_set = models_set + [generateXModels(input_data, output_data, model_type,
    [5, 33, 39, 6, 24, 19, 4, 1],
    ["leaky_relu", "leaky_relu", "sigmoid", None, "leaky_relu", "leaky_relu", None],
    3,epochs_a)]

#generate random Models
models_set = models_set + generateRandomModels(input_data, output_data, model_type,
    2, 3, epochs_a)
save_best = 6 #number of saved models
#------------------------------End of Definition Part ---------
```

6 Experiment: Design, Evaluation and Conclusion

To realize the evaluation of the hypothesis that the ASHGA has a benefit to the growth of the plants, a long-term experiment is made. Three pods were in use, compared on different treatments.

This long-term test ran from 29th April to the 10th July 2021. Of course, the test was not only for evaluating the gain in growth, additionally it was used to evaluate the behaviour of the intelligent irrigation system and the reliability of the system. This section is focused on the growth, the other evaluations are discussed in the chapters 5 AI-Planner and 7 Costs and Reliability.

This experiment should reveal, what parts of the system have the biggest impact to the growth of the plants. One point is that the automized watering can bring a benefit to growth, compared with an inconsequent manual watering. The second point to evaluate is, if the additional devices have a huge influence, as they bring additional costs and efforts.

The null hypothesis is that there is no difference in the three groups. Actual there are two null hypotheses: That there is no gain in growth from the different watering methods and there is no effect from the additional devices (lights, heating, shutter).

6.1 Design of the Experiment

Three big pods, with a raised bed like construction, were build. One with the full functionality of the ASHGA, a second with only the simple watering method of the Fallback System and a third, as control group, totally manual cared. They are later called Garden Part 1 (full system), Garden Part 2 (Fallback System), Garden Part 3 (manual care), or short GP1 – GP3.

The ASHGA assumed the following tasks (for the details of each controlled device, see chap. 3.7 Fallback System):

- Intelligent Irrigation: The AI-Planner was used to do the intelligent watering of Garden Part 1. The planner combines a weather forecast and an evaporation prediction to use natural watering from the environment (rain) as good as possible and optimize the water consumption. Of course, this should be done without a negative effect to the growth of the plants.
- Artificial Lighting: With the use of artificial lighting, a gain in growth is expected. The lights give additional 6000 lx.
- Heating Plates: Until mid of May, the temperature dropped sometimes at night below 0°C. The heating of the soil can prevent the roots from freezing. This should give a gain in growth during the early grow phases of the plants.

- Shutter: Should prevent damage to the plants from hazardous precipitation, especially hail. If strong hail occurs once, the whole harvest can be destroyed instant.

The experiment had a pre-phase, as the seedlings had to be raised. This started at the mid of March. To boost the growth in that phase, the same LED-lights were in use that later were installed on the ASHGA. After the the coldest period was over at the end of April, the seedlings were planted into the pods and the main experiment started.

The plants, per pod, are:

- 3x Cauliflower
- 3x Swiss Chard
- 3x Lettuce
- 2x Tomato; the tomatoes replaced the lettuce, after the harvest on the 12th June 2021.

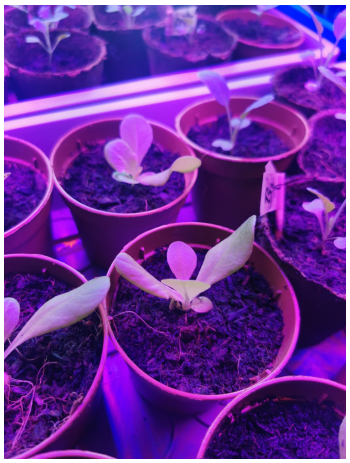


Figure 6.1: Seedlings

The pods had the following setup:

- Holes on the side and bottom of the pods, to let the water pass and guarantee a good air circulation to prevent the roots to rot.
- A layer with branches for a good drainage
- A layer rough mulch also for the drainage
- Two different layers of soil

In the lower soil layer of GP 1, the heating plates were installed.

Figure 6.2: Construction of the pods / raised beds

To select the plants for the pods, the best seedlings were pre-selected. They were sorted by size, and then distributed to the pods. The way, every pod got one of the biggest, middle and smaller ones. So, it was guaranteed, that the pods had nearly the same starting conditions.

Figure 6.3: Sorting the seedlings

Then, one pod was placed in the ASHGA prototype and one was only connected with the hose and a pump of the system. The foil, used to protect the plants at the beginning and supported the heating plates to store the heat, was removed after two weeks, as the temperture did not fall below 0°C anymore.

Figure 6.4: Finished Setup

6.2 Analysis

To compare the different groups, the weight of the plants is measured. Unfortunately, the cauliflowers and tomatoes were not ready. So, the cauliflowers were planted out and weigh (and then re-planted), the tomatoes were measured in size/hight of the plants.

The control group, GP 3, is the reference for the values. So, the mean of each type of plants and pods are calculated and normalized to the reference values. The results in the diagram is the proportion from reference value to the values of GP 1 and GP 2. In this way the tendences of the pods can be visualized.

The hypothesises are:

- Hypothesis 1: A gain in growth due to the additional devices: If this hypothesis is correct, GP 1 will shows a better growth than GP 2 and GP 3, as they do not use the devices.
- Hypothesis 2: There is a gain in growth due to the used irrigation systems. GP 1 should have a better growth than GP 2, both should exceed the control group GP 3, as the garden is better observed by the system than by manual care, in terms of the irrigation.
- Null Hypothesises: There is no significant difference in growth, especially in the automized garden parts.

To prove, if there is a significant difference between GP 1 / GP 2 and GP 1 / GP 3 an independent two-sample t-test is made, see table 6.2 for the results.

Measurements:

Plants	GP 1	GP 2	GP 3
Lettuce [g]	263; 242; 160 (221.7)	224; 194; 179 (199.0)	246; 246; 172 (221.3)
Swiss Chard [g]	406; 195; 121 (240.7)	396; 140; 82 (206.0)	409; 174; 129 (237.3)
Cauliflower [g]	271; 234; 107 (204.0)	127; 68; 41 (78.7)	239; 216; 194 (216.3)
Tomato [cm]	74; 68 (71.0)	72; 70 (71.0)	48; 44 (46.0)

Table 6.1: Results of the Measurements, the mean is given in brackets

Plants	t-value GP 1 / GP 2	t-value GP 1 / GP 3
Lettuce	0.66	0,01
Swiss Chard	0,27	0,03
Cauliflower	2,247	-0,24
Tomato	0	7,54

Table 6.2: T-Test Analysis; with 4/2 degrees of freedom (3/2 plants per type and pod), t has to be > 2.132/2,92 for significance ($p = 0.05$)

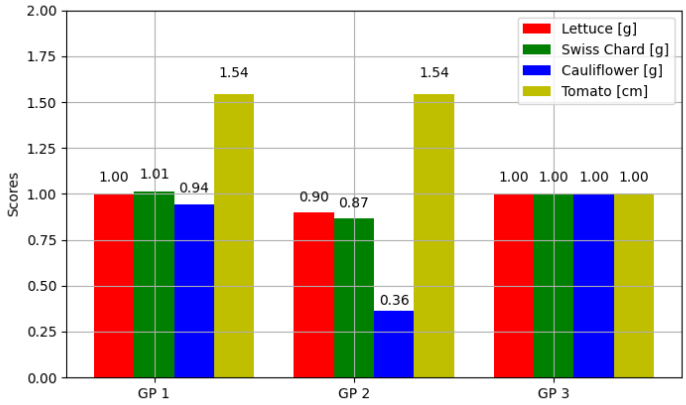


Figure 6.5: Diagram of the Results, values are normalized to the Control Group GP 3

6.3 Evaluation and Conclusion of the Experiment

The sample size is far too low to make a meaningful statement and the variance in each pod itself is very high. This can be seen in the t-test (table 6.2).

It is difficult to make a meaningful statement, as the t-test do not show a high significance neither for the hypothesis that the watering method affects the growth, nor the hypothesis that the additional devices give a benefit to growth.

It is hard to say, if the null hypothesises can be rejected. On the one hand, the growth of GP 1 compared to the control group GP 3 shows no benefits, except in the growth of the tomatoes. So, it seems that the additional devices like lights, shutter and heating plates had no effect to the

growth, especially as GP 2 also shows a better growth at tomatoes, without the devices (H1). On the other hand, GP 1 has a better score in growth compared to GP 2, but only with significance at the cauliflowers (H2).

These results could be explained by the assumption, that GP 2 did water too much. At the weighting process, I could see, that the roots of the plants (especially of the cauliflowers) were significant shorter and some roots were rotting. As discussed in chap. 5.8 Analysis of Water Consumption, GP 2 was watered with 36 liter, whereas GP 1 and GP 2 were watered with 27 and 24 liter. This might explains the better score of GP 1 and GP 3 than GP 2. So, probably, a test with more plants can give meaningful results. The threshold to water of the Fallback System might has to be calibrated.

The leak in gain from the additional lighting, heating and the shutter could be explained, as it was a real condition experiment.

- Lights: The power of the lights was maybe too insufficient to show a huge effect. During the pre-phase, when the seedlings were raised, the lights had a visible effect, as they were in a distance of about 30 cm from the seedlings. In the prototype, the distance to the plants was between 70cm and 120cm, this resulted in additionally 6000lx. This less than a tenth of clear sky daylight and even a half of a cloudy day.
- Shutter: During the experiment, it did only hail once, short and not heavy. So, it is to assume that in this experiment, the impact is negligible. This way, it is a showcase of functionality.
- Heating: Only the soil was heated, maybe additional heating lamps or air heating is needed. And the heating plates were activated only the first 2-4 weeks, as the temperature raised in May to June. Perhaps the heating plates would have had a visible effect, if the plants were planted earlier in season.

In this experiment, the null hypothesis that the additional devices gives a benefit, is not to be rejected.

7 Costs and Reliability

This chapter provides short overview of the costs and the reliability of the prototype.

7.1 Costs

The components of the ASHGA prototype are made of DIY sensors and devices on the marked, making it cheap, but not very reliable.

The costs of the prototype:

Component	Costs
Raspberry PI	40€
Sensors and Devices, Lights and Pumps, where most expensive	150€
Parts for the shutter	50€
Parts for the greenhouse (screws, wood, sockets, foil)	60€
Miscellaneous Parts (Wire, Breadboards, etc. . . .)	40€
Autonomous Sensors, Solar Panel, Battery (2x)	50€
Total Costs of the Prototype	390€

Table 7.1: Costs of the Prototype

The costs of the raised beds for the experiment:

Component	Costs
Raised Beds: Soil, mulch, pods (3x)	150€
Seeds and seedlings	15€
Total Costs of the Experiment	165€
Total Costs of the Thesis (Prototype and Experiment)	555€

Table 7.2: Costs of the Experiment and total Costs

7.2 Reliability

During the whole runtime (from May to July 2021), all in all, the prototype was not very reliable, nearly every week, there were some issues. Some sensors and devices did not have any problems, others had a very bad failure rate. But that were mostly minor problems, fixed in a few minutes,

often by restarting the system or check the connection. The Software and Raspberry PI hardware, itself, was very stable, openHAB, the Raspberry PI Device and Sensor System and the AI-Planner did not crash any time itself (except if there were bugs to fix during the development and integration). This makes the ASHGA, as it is, not suitable to care about the garden, during longer time periods, like holydays. If the user can check the functionality every day, there are mostly minor fixes to do.

The most issues can be put down to the use of cheap sensors and devices, and the fact that the whole wiring is realized by breadboards and jumper cables. So, using more professional sensors and a soldered circuit boards, would give a high benefit to the total reliability.

Detailed information of the reliability of the components during the long term test:

Software and Hardware (Raspberry PI): No problems at all. Only during development and integration, some bugs had to be fixed. The Raspberry PI had an up-time of nearly 100%. The AI-Planner had two times the issue that it could not reach the new tomorrow.io API V4 for some hours, but automatically used the older version. More providers would counteract this problem.

Moisture Sensors in general: The sensors were very contact sensitive. If they were displaced minimal, they often had to be calibrated. Also, strong wind could bring this issue. All used moisture sensors had not the best reliability.

Chirp! Moisture Sensor The connection on the breadboard was not very stable. In Addition, the i^2c address changed sometimes, as the command is not protected on the bus. Sometimes it had a total failure, causing that the whole system had to be unplugged and restarted.
The failure rate was about 2/month. But sometimes it take 1-2 days to fix the problems.

Autonomous Moisture Sensor: The used LilyGo sensors had a very bad reliability. They often crashed, going in a deadlock and supressed the deep-sleep. This caused a high drain of battery power during the failure. If they failed, it was fixable by disconnect the battery for a short time, so it is no major issue, but making it impossible to provide autonomous mesurements for longer time periods.
The failure rate was 1/week.

Relay: At first, 3.3V Relays were used. They had not enough power to switch reliable. After the exchange with 5V relays and using a own power supply, there was no issue anymore.

Lights, Shutter, Lux-Meter: This components were absolute stable. At the beginning the Shutter was modified, using eyelets instead of linear glide bearings. The glide bearings were very rust/dirt prone.

Water Pumps: The water had been changed during longer rain intervals, because algae grow in the tanks, blocking the pumps. Apart from that, no issues at all.

8 Outlook

Summarizing the conclusions, the ASHGA, especially the AI-Planner shows a good behavior and learning ability. Of course, the issues of the reliability have to be fixed. But it is remarkable that, even as the system had some issues in reliability, the two automized pods were never watered manually.

The AI-Planner is very independent from the other components, so it is possible to implement interfaces for other smart home solutions and more weather forecast providers. The AI-Planner itself is full functional now, the process of generating the ANN-based evaporation prediction can be optimized. It would be desirable, to find a good measurement method to automize the selection process of the ANN models.
So, as outlook, there is the possibility, to do an open-source project of the AI-Planner, as this component is nearly finished and already useable.

It saved already now, during this long-term test, 25% water in comparison to the simple automized watering system, even as the models "THL" and "THLTD1" had not the best prediction accuracy, but were in use during a long time interval, as development / improvement and the long-term experiment fell together. The better the model, the better the plans and the better the water saving. If the additional effort of the other parts of the ASHGA is worth it, could not be answered certainly.

The results of the evaluation of growth, comparing the two automized garden parts, can be interpreted the way that the ability of the AI-Planner to predict the real amount of water in the soil, instead of using the measured soil level in the upper layers, brought the most benefit in growth and protected the roots from rotting.
The aditional lighting, heating and the shutter did not show any benefits. Following experiments could clarify, if it was due to the environmental conditions or the false hypothesis itself. Greenhouses, of course, prove the benefit in growth, by regulate the climate conditions. So, I guess it was due to the environmental condition of this year in this garden. Additionally the power of the lights might was insufficient and only soil heating, indstead of air heating or using heat lamps was the wrong decision.

All in all, I expect a huge potential of this ANN based planning optimized irrigation method. Further experiments could figure out, if it also has a benefit, compared to the mentioned Smart Home Irrigation Systems on the marked, as they have a different approach to use the weather forecast, as they modify the users plans, instead of generating the plans by a planning optimizing algorithm that focuses on the (predicted) moisture level.

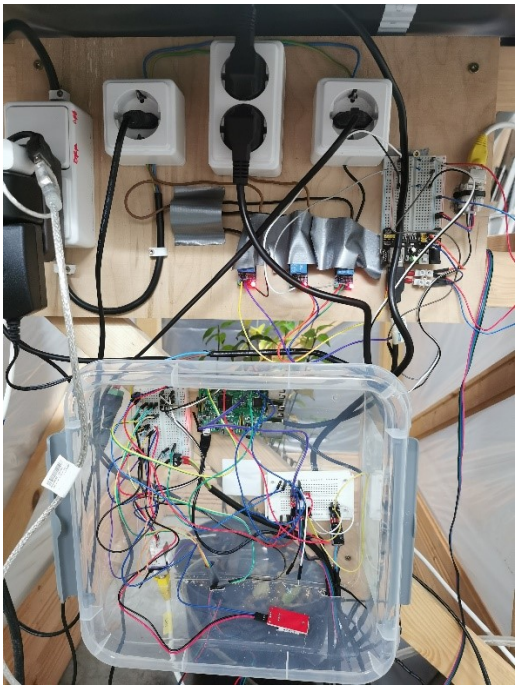
But even with the benefits, as a smart home system, there is the potential to do the irrigation in a greater scale and improve the evapotranspiration prediction and irrigation of agriculture fields. Therefore, more sensors have to be installed in the fields and the planner has to coordinate them, but I think it is possible to do these modifications for such scales. The work of Sharad K. Jain et al. showed that is possible to train ANNs to predict the ET_0, why should it not be possible to predict

the real evaporation? This thesis shows, that it works in small scales. Further research is needed to evaluate this. If it turns out, that it is useable in practice with low effords, it would make the need of standardized equations, like the FAO56-PM equation, obsolete in many areas of agriculture and farming.

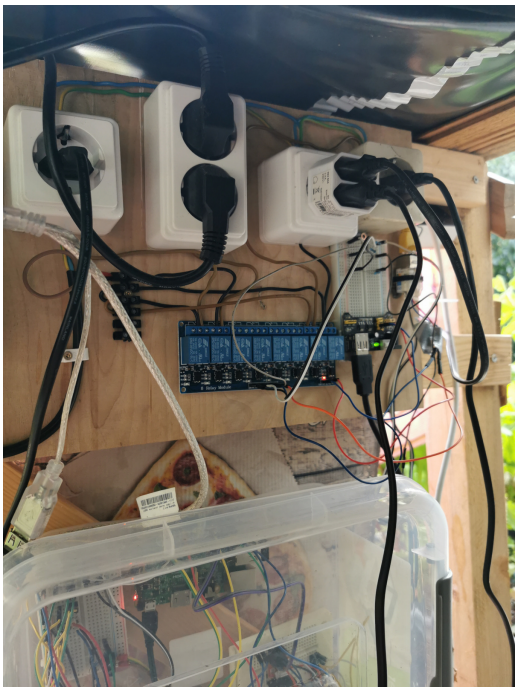
Water scarcity is a big problem in many countries, agriculture has a huge effect to this problem and is highly affected by it. The OECD claims, that the agriculture irrigation causes 70% of the water usage, worldwide.[npd] As the world population is still rising, there has to be developed and improved the methods to minimize the water consumption, as the food consumption will raise. And with the future climate changes, even more countries will be affected by water scarcity. Systems that can use the environmental conditions, as good as possible, to minimize power and water consumption, will be a very helpful tool for all.

A Appendix

A.1 Detailed Sensor and Device List



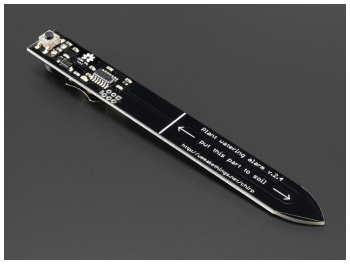
(a) Whole Raspberry PI Device and Sensor System



(b) Relay for 230V Devices

Sensors:

- Moisture Sensor: Chirp! Capacitive Moisture Sensor
  Protocol: I2C, using lib from Miceuz https://github.com/Miceuz/PlantWateringAlarm [Accessed on 2021-07-30]
  Provides: Soil Moisture Level in %
  Calibration: Measurement in dry soil for min value, and fresh watered, with a waiting time of about 30 min, to let the water spread and seep.
  Photo is taken from the provided Git-Repo



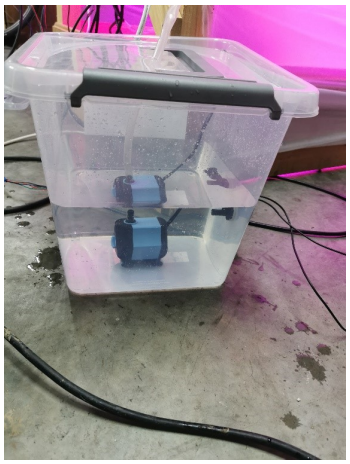
- Sun Radiation Sensor: Lux-Meter BH1750
  Protocol: I2C, using a lib from oskar456 https://gist.github.com/oskar456/95c66d564c58361ecf9f [Accessed on 2021-07-30]
  Provides: Solar Radiation in lx
  It is covered with a semitransparent plastic (filter factor 2.75) and silicone to protect the sensor from the weather conditions.
  Calibration: 100,000lx at clear sky daylight as reference is used.

- Temperature and Humidity Sensor: DHT22
  Protocol: Specific 1-Wire Protocol, Using Adafruit_DHT lib. https://github.com/adafruit/DHT-sensor-library [Accessed on 2021-07-30]
  Provides: Temperature in °C and Humidity in % (Relative Humidity)
  Calibration: Not needed

- Water Tank Level Sensor: No-Name Product
  Protocol: Voltage ON-OFF-Switch
  Provides: Boolean, Full/Empty Float Sensor, switches circuit on or off, can be read directly via GPIO. Disables Watering, if water level is too low.

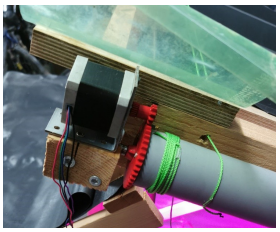
Devices:

- Watering: Water Pumps
  Protocol: 230V relay
  The time interval for the watering is calculated by amount/flow rate. Can be excheanged by electro-magnetic valves.



- Artificial Lighting: LED-Grow-Lights
  Protocol: 230V relay
  Over 200 single LEDs, in red and blue light spectrum, optimized for plants. LEDs are used, to minimize the power consumption.

- Shutter: Self constructed mechanics, Stepper Motor Driver, similar to the A4988.
  Protocol: Self-written Protocol
  Uses a 12V Stepper-Motor



(a) Stepper-Motor

(b) Open

(c) Closing

(d) Closed

- Soil Heating: Heating Plates for Gardening
  Protocol: 230V relay
  Two of them are installed in the full automized raised bed (Garden Part 1). They use heating wires, 15W each. Could be replaced or extended with air heaters or IR heating lamps.

A.2 Further Evaluation of the AI-Planner

This section evaluates the AI-Planner on specific examples.

A.2.1 Evaluation of Rain Handling

Of course, the planner just can only be as good as the weather forecast prediction. Here are some examples in which the planner failed or successfully predicted the moisture level:

First Example (21.06.21 20:16 – 20:46)

This example shows a failure of the used weather forecast. It predicted completly different conditions within 30 minutes between the two plans. Figure A.3 is an extract of the logging, the planner writes after each circle into the log-file and disblays the data on which the plans are based. Table A.1 shows the resulting difference of the moisture development over time of the two plans in comparison with the later real measurements.

Of course, the MSE differed a lot between the two plans. The first had an MSE of 244, after the update of the forecast, the MSE decreased to 11.2 (over the full 24h), which is a mean deviation of $\sqrt{MSE} \approx 15.62$ (Plan 1) and 3.34 (Plan 2)

```
------------------------------------------------------------------------
[2021-06-21 20:16:12]
Weather Data raw (8 hours)
Date (3h Forecast)      temp    hum     cloud   p mm/h  p %     p type          Date (long term FC)     temp    hum     cloud   p mm/h  p %     p type
2021-06-21T20:00:00     23.31   61.02   100.0   0.359   37.5    rain            2021-06-22T00:00:00     18.17   90.72   99.31   0.0     0.0     rain
2021-06-21T21:00:00     22.87   66.83   100.0   0.570   40.0    rain            2021-06-22T01:00:00     17.81   91.86   76.04   0.0     0.0     rain
2021-06-21T22:00:00     21.71   76.15   100.0   0.734   40.0    rain            2021-06-22T02:00:00     17.4    93.12   83.69   0.0     0.0     rain
2021-06-21T23:00:00     19.83   84.64   100.0   0.789   40.0    rain            2021-06-22T03:00:00     17.07   94.54   85.85   0.111   15.0    rain
[...]
Watering Plan (Moisture at Start  80.55 ):
['m:81s:0loss:0.98rain:1evap:0.78', 'm:81s:0loss:0.47rain:1evap:0.27', 'm:82s:0loss:0.34rain:1evap:0.14', 'm:82s:0loss:0.38rain:1evap:0.18', 'm:82s:0', [...]]
Watering: 0ml
[...]
------------------------------------------------------------------------
[2021-06-21 20:46:38]
Weather Data raw (8 hours)
Date (3h Forecast)      temp    hum     cloud   p mm/h  p %     p type          Date (long term FC)     temp    hum     cloud   p mm/h  p %     p type
2021-06-21T20:00:00     22.0    69.49   100.0   2.460   40.0    rain            2021-06-22T00:00:00     18.17   90.72   99.31   0.0     0.0     rain
2021-06-21T21:00:00     21.65   76.52   100.0   2.656   40.0    rain            2021-06-22T01:00:00     17.81   91.86   76.04   0.0     0.0     rain
2021-06-21T22:00:00     19.77   84.84   100.0   2.906   40.0    rain            2021-06-22T02:00:00     17.4    93.12   83.69   0.0     0.0     rain
2021-06-21T23:00:00     18.81   87.98   100.0   3.164   38.7    rain            2021-06-22T03:00:00     17.07   94.54   85.85   0.111   15.0    rain
[...]
Watering Plan (Moisture at Start  81.58 ):
['m:86s:0loss:5.0rain:3evap:0.16', 'm:91s:0loss:5.0rain:3evap:0.18', 'm:96s:0loss:5.0rain:3evap:0.21', 'm:100s:0loss:5.0rain:3evap:-0.43', 'm:99s:0', [...]]
Watering: 0ml
[...]
------------------------------------------------------------------------
```

Figure A.3: Planner Log-File Extract 1

Second Example (28.06.21 19:31)

In this example, Figure A.4, the first weather forecast predicted rain, but far too little and 4 hours to late. However, it is a good example where the rain prediction correctly prevented the system to water, the comparison calculation (that does not use rain in the model) would have watered 600ml (see the last line of the first log entry, the action was a0/“m:78s:” vs. the comparison plan a1/“1 XX”). This resulted in a MSE of 64.6. Also, the shutter correctly closed after the weather forecast update. In addition, this was an exceptional harsh storm[1]. In conclusion, it cannot be considered a total failure as the weather conditions were specal and the system did in fact act as was desired.

Log entry of the raspberry system:

```
[2021-06-28␣21:03:12]␣TcpServer:␣Received␣msg:{␣"changeSutter":"CLOSE"␣}
```

[1] https://www.stuttgarter-zeitung.de/inhalt.unwetter-ueber-stuttgart-land-unter-in-der-stadt.f377f3b6-3b27-47a5-92fd-f6410b493bf9.html [Accessed on 2021-07-30]

	Plan 1	Plan 2 (+0:30)	Measured
21	80.55	81.58	81.58
22	81.53	86.58	82.23
23	82	91.58	99.44
0	82.34	96.58	99.44
1	82.72	100	98.02
2	82.3	99.58	96.95
3	81.92	99.2	96.99
4	81.61	98.89	95.6
5	81.35	98.62	95.42
6	81.16	98.44	95.08
7	80.95	98.23	94.88
8	80.23	97.51	93.62

Table A.1: This table shows, how the two plans (time difference: 30min) differ by an incorrect (Plan 1) weather forecast. The second Plan based on the updated weather forecast and shows that the planner has a good behaviour involving the rain in his moisture model, if the weather forecast is precise enough.

```
----------------------------------------------------------------------
[2021-06-28 19:31:34]
Weather Data raw (8 hours)
Date (3h Forecast)      temp    hum     cloud   p mm/h  p %     p type          Date (long term FC)     temp    hum     cloud   p mm/h  p %     p type
2021-06-28T19:00:00     20.0    86.32   100.0   0.0     0.0     rain            2021-06-28T23:00:00     17.36   83.89   100.0   1.741   60.0    rain
2021-06-28T20:00:00     19.74   86.39   100.0   0.0     0.0     rain            2021-06-29T00:00:00     16.99   85.48   100.0   0.413   65.0    rain
2021-06-28T21:00:00     19.61   87.78   100.0   0.0     0.0     rain            2021-06-29T01:00:00     16.42   87.96   100.0   0.660   65.0    rain
2021-06-28T22:00:00     19.45   86.66   100.0   0.0     0.0     rain            2021-06-29T02:00:00     16.29   89.42   100.0   2.100   70.0    rain
[...]
Watering Plan (Moisture at Start  78.55 ):
['m:78s:0', 'm:78s:0', 'm:78s:0', 'm:77s:0', 'm:82s:0loss:5.0rain:3evap:-0.44', 'm:82s:0', 'm:82s:0', 'm:87s:0loss:5.0rain:3evap:-0.36', [...]]
Watering: 0ml
Used Model: ['temp,hum,lux,time,d_temp', 'THLTD1']
Changed watering behaviour because of rain: True;    with rain: 'm:78s:0' without rain: '1 XX'
----------------------------------------------------------------------
[2021-06-28 20:32:45]
Weather Data raw (8 hours)
Date (3h Forecast)      temp    hum     cloud   p mm/h  p %     p type          Date (long term FC)     temp    hum     cloud   p mm/h  p %     p type
2021-06-28T20:00:00     19.65   87.62   0.0     1.376   45.0    rain            2021-06-29T00:00:00     17.49   86.36   100.0   1.765   70.0    rain
2021-06-28T21:00:00     19.65   88.66   100.0   17.46   60.0    rain            2021-06-29T01:00:00     16.78   85.02   100.0   0.677   70.0    rain
2021-06-28T22:00:00     19.11   87.57   100.0   17.46   60.2    rain            2021-06-29T02:00:00     16.32   87.53   97.33   0.976   70.0    rain
2021-06-28T23:00:00     18.59   87.13   100.0   15.36   62.7    rain            2021-06-29T03:00:00     16.25   90.21   100.0   0.0     0.0     rain
[...]
Watering Plan (Moisture at Start  100.00 ):
['m:100s:0', 'm:100s:0loss:1e+01rain:4evap:-0.17', 'm:100s:0loss:1e+01rain:4evap:-0.27', 'm:100s:0loss:1e+01rain:4evap:-0.43', [...]]
Watering: 0ml
Used Model: ['temp,hum,lux,time,d_temp', 'THLTD1']
Changed watering behaviour because of rain: False;    with rain: 'm:100s:0' without rain: 'm:99s:0'
----------------------------------------------------------------------
```

Figure A.4: Planner Log-File Extract 2

Third Example (04.07.21 12:59)

This shows a normal average case of a rainy day without the extreme weather conditions of the other two cases. In the logged plan, Figure A.5, it can be seen, that rain is predicted in the 5^{th} to 7^{th} hour. Table A.2 compares the predicted moisture to the measured values. The MSE was about 9, so the mean deviation was about 3. That is an acceptable error, as weather is such a chaotic system.

	Prediction	Measured
13	81.65	81.71
14	81.65	81.31
15	81.82	80.83
16	82.75	80.2
17	83.24	80.22
18	84.71	80.66
19	85.69	81.52
20	86.26	82.02
21	86.22	82.47
22	85.99	83.04
23	85.72	83.06
0	85.28	82.86
1	84.86	82.25
2	84.47	82.07
3	84.11	81.76
4	83.79	81.5
5	83.48	81.15
6	83.26	80.86
7	83.02	80.71
8	82.32	80.06

Table A.2: Example 3: No extreme weather conditions, the predictions fits quite good to the measured values.

```
----------------------------------------------------------------------
[2021-07-04 12:59:23]
Weather Data raw (8 hours)
Date (3h Forecast)      temp    hum     cloud   p mm/h  p %     p type          Date (long term FC)     temp    hum     cloud   p mm/h  p %     p type
2021-07-04T12:00:00     22.03   63.19   100.0   0.0     0.0     rain            2021-07-04T16:00:00     19.44   80.02   100.0   0.565   85.0    rain
2021-07-04T13:00:00     22.28   62.03   100.0   0.0     0.0     rain            2021-07-04T17:00:00     19.01   80.37   100.0   0.312   85.0    rain
2021-07-04T14:00:00     22.28   64.79   100.0   0.0     0.0     rain            2021-07-04T18:00:00     18.11   86.61   100.0   0.700   75.0    rain
2021-07-04T15:00:00     21.31   71.13   100.0   0.0     0.0     rain            2021-07-04T19:00:00     18.33   83.29   100.0   0.198   65.0    rain
[...]
Watering Plan (Moisture at Start  81.65 ):
['m:81s:0loss:0.0029rain:0evap:0.0029', 'm:81s:0loss:0.17rain:0evap:0.17', 'm:82s:0loss:0.93rain:0evap:0.93', 'm:83s:0loss:0.49rain:0evap:0.49',
'm:84s:0loss:1.rain:2evap:0.97', 'm:85s:0loss:0.9rain:1evap:0.78', 'm:86s:0loss:0.5rain:2evap:0.069', 'm:86s:0', 'm:85s:0', 'm:85s:0', [...]]
Watering: 0ml
[...]
----------------------------------------------------------------------
```

Figure A.5: Planner Log-File Extract 3

A.3 Modify the Action Costs

To provide an extended option to change the behavior of the AI-Planner, a python script can be modified to do more complex cost functions than the command-line option or the settings in openHAB.

This way, additional aspects in behavior, like avoiding watering during sun shine, can be defined. To keep these modifications simple as possible, the cost calculation is out-sourced to an own file. The method ‘modActionCosts’ provides the most interesting data for improved cost calculation. Please keep in mind, that the resulting costs should be negative, to keep the planning algorithm correct.

This example script is a show case, how to avoid watering in sun shine:

Listing A.1 Modify the Costs per Script, modify_costs.py

```
'''
    action: integer, from 0 = no watering to 3 = max watering
    action_costs: float, the actual costs from the settings in OH
                        or cmd-line argument
    temp: float, the predicted temperature
    lux: float, the predicted sun light intensity in lx
    hour: integer, daytime (hour) of the plans actual step
    moisture: float, predicted moisture
'''
def modActionCosts(action, action_costs, temp, lux, hour, moisture):
    #add extra costs to the action, if the lux > 30000 and
    #if it is not the 'no watering action' a0
    if lux > 30000 and action > 0:
        action_costs = action_costs - 3

    return action_costs
```

Listing A.1 Modify the Costs

Bibliography

[Bus18] V. Bushaev. *Adam - latest trends in deep learning optimization*. Accessed on 2021-07-30. 2018. URL: https://towardsdatascience.com/adam-latest-trends-in-deep-learning-optimization-6be9a291375c (cit. on p. 42).

[CCC+15] M. Córdova, G. Carrillo-Rojas, P. Crespo, B. Wilcox, R. Célleri. "Evaluation of the Penman-Monteith (FAO 56 PM) Method for Calculating Reference Evapotranspiration Using Limited Data". In: *Mountain Research and Development* 35.3 (2015), pp. 230–239. DOI: 10.1659/MRD-JOURNAL-D-14-0024.1. URL: https://doi.org/10.1659/MRD-JOURNAL-D-14-0024.1 (cit. on pp. 27, 37).

[EHT+21] S. Evett, T. Howell, R. Todd, A. Schneider, J. Tolk. "Evapotranspiration of irrigated alfalfa in a semi-arid environment." In: (July 2021). URL: https://www.researchgate.net/publication/43255616_Evapotranspiration_of_irrigated_alfalfa_in_a_semi-arid_environment (cit. on p. 27).

[GNT04] M. Ghallab, D. Nau, P. Traverso. "Chapter 8 - Constraint Satisfaction Techniques". In: *Automated Planning*. Ed. by M. Ghallab, D. Nau, P. Traverso. The Morgan Kaufmann Series in Artificial Intelligence. Burlington: Morgan Kaufmann, 2004, pp. 167–191. ISBN: 978-1-55860-856-6. DOI: https://doi.org/10.1016/B978-155860856-6/50013-2. URL: https://www.sciencedirect.com/science/article/pii/B9781558608566500132 (cit. on p. 18).

[Her19] T. Herrington. *Why is the weather so hard to predict?* Accessed on 2021-07-30. 2019. URL: https://letstalkscience.ca/educational-resources/stem-in-context/why-weather-so-hard-predict (cit. on p. 69).

[Hri20] J. Hrisko. *Capacitive Soil Moisture Sensor Calibration with Arduino*. Accessed on 2021-07-29. 2020. URL: https://makersportal.com/blog/2020/5/26/capacitive-soil-moisture-calibration-with-arduino (cit. on p. 33).

[IIAJ03] S. Irmak, A. Irmak, R. G. Allen, J. W. Jones. "Solar and Net Radiation-Based Equations to Estimate Reference Evapotranspiration in Humid Climates". In: *Journal of Irrigation and Drainage Engineering* 129.5 (2003), pp. 336–347. DOI: 10.1061/(ASCE)0733-9437(2003)129:5(336). eprint: https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%290733-9437%282003%29129%3A5%28336%29. URL: https://ascelibrary.org/doi/abs/10.1061/%28ASCE%290733-9437%282003%29129%3A5%28336%29 (cit. on p. 37).

[JNS08] S. Jain, P. C. Nayak, K. Sudheer. "Models for estimating evapotranspiration using artificial neural networks, and their physical interpretation". In: *Hydrological Processes* 22 (June 2008), pp. 2225–2234. DOI: 10.1002/hyp.6819. URL: https://www.researchgate.net/publication/227517362_Models_for_estimating_evapotranspiration_using_artificial_neural_networks_and_their_physical_interpretation (cit. on pp. 26, 37, 38).

[Jua] R. Juaron. *When is the best time to water the garden?* Accessed on 2021-07-30. Iowa State University. URL: https://hortnews.extension.iastate.edu/faq/when-best-time-water-garden (cit. on p. 70).

[KB17] D. P. Kingma, J. Ba. "Adam: A Method for Stochastic Optimization". In: (2017). arXiv: 1412.6980 [cs.LG] (cit. on p. 42).

[KWZ+14] D. A. Karras, W.-g. Wang, S. Zou, D. Chen, J. Kong. "Prediction of the Reference Evapotranspiration Using a Chaotic Approach". In: (July 2014). DOI: 10.1155/2014/347625 (cit. on p. 26).

[Mah17] J. Mahanta. *Introduction to Neural Networks, Advantages and Applications*. Accessed on 2021-07-30. 2017. URL: https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207 (cit. on p. 38).

[MHN13] A. L. Maas, A. Y. Hannun, A. Y. Ng. "Rectifier nonlinearities improve neural network acoustic models". In: *in ICML Workshop on Deep Learning for Audio, Speech and Language Processing*. Stanford University. 2013. DOI: 10.1.1.693.1422. URL: https://ai.stanford.edu/~amaas/papers/relu_hybrid_icml2013_final.pdf (cit. on p. 41).

[Mic19] P. Michael. *A Conversion Guide: Solar Irradiance and Lux Illuminance*. 2019. DOI: 10.21227/mxr7-p365. URL: https://dx.doi.org/10.21227/mxr7-p365 (cit. on p. 44).

[Mij18] M. Mijwil. "Artificial Neural Networks Advantages and Disadvantages". In: (Jan. 2018). Accessed on 2021-07-30. URL: https://www.researchgate.net/publication/323665827_Artificial_Neural_Networks_Advantages_and_Disadvantages (cit. on p. 38).

[Nau] R. Nau. *Moving average and exponential smoothing models*. Accessed on 2021-07-29. Duke University. URL: http://people.duke.edu/~rnau/411avg.htm (cit. on p. 32).

[NIGM18] C. Nwankpa, W. Ijomah, A. Gachagan, S. Marshall. "Activation Functions: Comparison of trends in Practice and Research for Deep Learning". In: *CoRR* abs/1811.03378 (2018). arXiv: 1811.03378. URL: http://arxiv.org/abs/1811.03378 (cit. on p. 40).

[npa] n.p. *Chapter 2 - FAO Penman-Monteith equation*. Accessed on 2021-07-30. Food and Agriculture Organization (FAO). URL: http://www.fao.org/3/x0490e/x0490e06.htm (cit. on p. 37).

[npb] n.p. *Chapter 6 - ETc - Single crop coefficient (Kc)*. Accessed on 2021-07-30. Food and Agriculture Organization (FAO). URL: http://www.fao.org/3/x0490e/x0490e0b.htm (cit. on p. 37).

[npc] n.p. *Diseases of Greenhouse Ornamental Crops*. Accessed on 2021-07-29. TEXAS A&M UNIVERSITY. URL: http://hortipm.tamu.edu/ipmguide/path/diseases.html (cit. on p. 25).

[npd] n.p. *Managing water sustainably is key to the future of food and agriculture*. Accessed on 2021-07-30. URL: https://www.oecd.org/agriculture/topics/water-and-agriculture/ (cit. on p. 84).

[npe] n.p. *Soil Water Evaporation*. Accessed on 2021-07-30. U.S. DEPARTMENT OF AGRICULTURE. URL: https://hrsl.ba.ars.usda.gov/SPAW/SPAW%20Reference%20Manual/SoilWaterEvaporation.htm (cit. on p. 38).

[Org] W. M. Organization. *HAPTER 8. MEASUREMENT OF SUNSHINE DURATION*. Accessed on 2021-07-29. URL: https://library.wmo.int/doc_num.php?explnum_id=3154 (cit. on p. 36).

[Org20] W. M. Organization. *World Meteorological Day – 23 March*. Accessed on 2021-07-30. 2020. URL: https://library.wmo.int/doc_num.php?explnum_id=10227 (cit. on p. 69).

[PD17] M. Poudel, B. Dunn. *Greenhouse Carbon Dioxide Supplementation*. Accessed on 2021-07-29. OKLAHOMA STATE UNIVERSITY. 2017. URL: https://extension.okstate.edu/fact-sheets/greenhouse-carbon-dioxide-supplementation.html (cit. on p. 25).

[Roj96] R. Rojas. *Neural Networks - A Systematic Introduction*. Springer-Verlag Berlin Heidelberg, 1996. ISBN: 978-3-540-60505-8. DOI: 10.1007/978-3-642-61068-4 (cit. on p. 41).

[Sha18] V. Sharma. *METHODS AND TECHNIQUES FOR SOIL MOISTURE MONITORING*. Accessed on 2021-07-29. Department of Plant Sciences, University of Wyoming. 2018. URL: http://wyoextension.org/publications/html/B1331/ (cit. on p. 33).

[Tem08] R. Tempo. "Randomized Algorithms for Systems and Control: Theory and Applications". In: (2008). URL: https://apps.dtic.mil/dtic/tr/fulltext/u2/a514074.pdf (cit. on p. 18).

[TLL95] I. V. Tetko, D. J. Livingstone, A. I. Luik. "Neural network studies. 1. Comparison of overfitting and overtraining". In: (1995). DOI: 10.1021/ci00027a006 (cit. on p. 48).

[Von10] C. Von Zabeltitz. *Integrated greenhouse systems for mild climates: climate conditions, design, construction, maintenance, climate control*. Springer Science & Business Media, 2010. ISBN: 978-3-642-14581-0. DOI: 10.1007/978-3-642-14582-7 (cit. on p. 25).

[ZRM+13] M. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, G. Hinton. "On rectified linear units for speech processing". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2013, pp. 3517–3521. DOI: 10.1109/ICASSP.2013.6638312 (cit. on p. 41).

Declaration

I hereby declare that the work presented in this thesis is entirely my own and that I did not use any other sources and references than the listed ones. I have marked all direct or indirect statements from other sources contained therein as quotations. Neither this work nor significant parts of it were part of another examination procedure. I have not published this work in whole or in part before. The electronic copy is consistent with all submitted copies.

---

place, date, signature