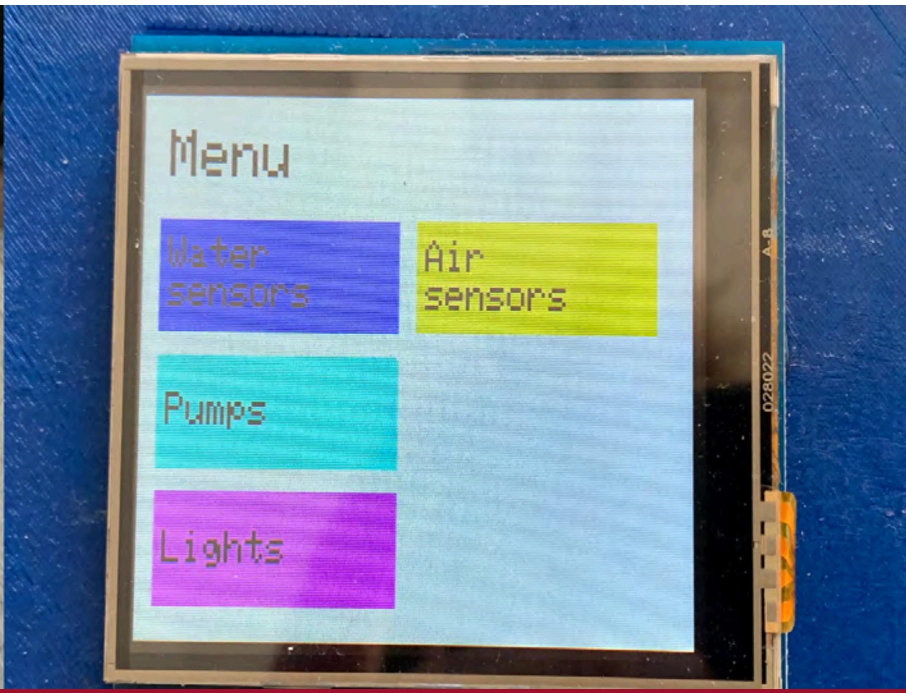
DEGREE PROJECT IN TECHNOLOGY,
FIRST CYCLE, 15 CREDITS
STOCKHOLM, SWEDEN 2021

Optimizable Hydroponic Plant Incubator

Building a hydroponic plant incubator with a highly optimizable environment

JONATHAN LING

GUSTAV LINDSTRAND



KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF INDUSTRIAL ENGINEERING AND MANAGEMENT

Optimizable Hydroponic Plant Incubator

Building a hydroponic plant incubator with a highly optimizable environment

JONATHAN LING
GUSTAV LINDSTRAND

Bachelors Thesis at ITM
Supervisor: Nihad Subasic
Examiner: Nihad Subasic

TRITA-ITM-EX 2021:48

Abstract

This report investigates how to build a compact, optimizable and at the same time user friendly hydroponic system for growing plants as efficiently as possible. Hydroponics grows plants using water with dissolved nutrients instead of soil, allowing faster and more efficient growth.

The focus has been on the implementation and the usability of such a system, centering around monitoring and to some degree controlling important parameters for growth such as humidity, air temperature, nutrition concentration and light intensity, colour and exposure time. The plant is enclosed in a confined space with artificial lighting which allows thorough control of the light environment.

In order to achieve the desired level of control over the growth parameters, several sensors along with a microcontroller were used. A touch screen with a custom built graphical user interface was also connected to allow the user to control and monitor important aspects of growth conditions.

The conclusion drawn from this project is that there is a high order of optimizability within the boundaries of this project. The measured factors are easily read on an intuitive, easily navigated touch screen for direct feedback. Regarding the lights' effect on plant growth, the conclusion is drawn that plants grow well with many types of lights, but more time is needed to thoroughly investigate different light exposure times, colour and intensity.

Keywords: Hydroponic systems, Mechatronics, Plant growth, Optimization

Referat

Optimizable Hydroponic Plant Incubator

I denna rapport undersöks hur man kan bygga ett kompakt, optimerat och samtidigt användarvänligt hydroponiskt system för att odla växter så effektivt som möjligt. Hydroponiska system får plantor att växa genom att använda vatten med näring istället för jord, vilket tillåter snabbare och mer effektiv tillväxt.

Fokuset i denna rapport har varit på implementeringen och användbarheten av ett sådant system, med tyngpunkt på övervakning och till viss grad styrning av viktiga faktorer i en plantas tillväxt såsom luftfuktighet, temperatur, näringskoncentration och ljusintensitet, färg och exponeringstid. Växten är innesluten i ett begränsat utrymme med artificiellt ljus, vilket tillåter genomgående kontroll av ljusmiljön.

För att uppnå önskad kontroll av tillväxtparametrarna, användes ett flertal sensorer tillsammans med en mikrokontroller. Till detta kopplades en pekskärm med ett egentillverkat användargränssnitt, som tillåter användaren att kontrollera och övervaka viktiga aspekter i tillväxten.

Slutsatsen från detta projekt är att det finns en hög grad av optimerbarhet inom denna konstruktion. De uppmätta parametrarna kan enkelt avläsas från en intuitiv, lättnavigerad pekskärm för direkt återkoppling. Avseende ljusets inverkan på tillväxt, dras slutsatsen att plantor växer väl med många typer av lampor, men det krävs mycket tid för att undersöka tillväxt med olika ljusexponering, ljusintesitet och färg.

Nyckelord: Hydroponiska system, Mekatronik, Odling, Optimering

Acknowledgements

We would like to thank Nihad Subasic for the informative lessons and quick responses whenever a problem has occured. We would also like to thank both assistants Amir Avdic and Malin Lundvall for their help as well as Staffan Qvarnström for his help with choosing components and support. We would also like to thank the company Botanium AB for the initial interview where a lot of questions were answered. Lastly, we would like to thank the Lindstrand family for lending out facilites to work in.

Jonathan Ling & Gustav Lindstrand
Stockholm, May 2021

Table of Contents

Nomenclature

Abbreviations

CAD	Computer Aided Design
EC	Electric Conductivity
GUI	Graphical User Interface
I2C	Inter-Integrated Circuit
ID	Identification
LECA	Lightweight Expanded Clay Aggregate
LED	Light Emitting Diode
PAR	Photosynthetic active radiation
PSU	Power Supply Unit
RGB	Red Green Blue
SPI	Serial Peripheral Interface
TDS	Total Dissolved Solids
TFT	Thin Film Transistor
USB	Universal Serial Bus
VAC	Volts Alternating Current
VDC	Volts Direct Current

Units

A	Ampere
Lm	Lumen

nm	Nanometer
ppm	Parts Per Million
V	Volt
W	Watt

List of Figures

List of Tables

Introduction

This chapter will cover the introduction of the project, presenting a background around the subject, why this particular project was chosen, the scope and method.

1.1 Background

With a growing world population comes a higher demand for food. Conventional farming on land has been around for many years, but the efficiency and outcome of this type of farming is soon not enough, the need for more effective methods is inevitable. One such method is hydroponic systems, which uses a fraction of the land conventional farming uses, improves growth rate, consumes less water and reduces the need for pesticides[1]. The growth parameters of a hydroponic system are also much easier to control, as the growing takes place in an artificial environment and not in nature.

1.2 Purpose

This project was chosen based on both authors' interest in plants and cultivation. Since hydroponic systems are still relatively young compared to traditional cultivating, trying new methods and changing parameters is in the interest of both home users and larger hydroponic farms. A study from MIT (Massachusetts Institute of Technology) shows that controlling the different aspects of growing basil can give plants better taste than traditionally grown basil[2]. This study was a great inspiration for this project and will influence the forthcoming work. The aim of this thesis is to make such a system, where focus is at making different parameters controllable for the best possible outcome. This product is well suited for industrial companies growing hydroponically, seeking to optimize their growth speed or plant quality. The research questions studied in this project are:

- How does light affect growth in a hydroponic system?
- Which factors needs to be controlled in order to ensure a high order of optimizability?

1.3 Scope

In order to ensure a product which is highly optimizable, nutrition concentration, light colour, light intensity and light exposure time should be able to be controlled. Humidity, air temperature, water level and nutrition concentration should be able to be monitored.

It was however decided that a few factors should be left out. These were both monitoring and controlling of pH-value, control of humidity and control of temperature. The reason for leaving out these factors is partly because of budget, but also that the complexity level compared to the added value for the product was deemed too high. Another reason was that these factors had been controlled and monitored in earlier projects at KTH, which would make it more interesting to try to control other parameters.

1.4 Method

The project that is chosen is a hydroponic plant system which lets the users optimize the growth conditions. A hydroponic system grows plants without the use of soil - instead, a growth medium is used to hold the plants and the roots, while a pump waters the plants with water containing dissolved nutrients. The water will then drip down back into the water tank. Nutrients will be stored in a separated small container beside the water tank, with a small pump that will be able to dose small amounts (a few ml) of nutrients in the water tank. The design is to be closed off, i.e. it will be enclosed, trapping light, heat and vapour from the outside environment. Such a system requires artificial light in order to grow the plants, which will be supplied with a LED-strip along the sides of the cylindrical plant incubator.

The incubator requires monitoring of several values, as well as the ability to regulate these conditions. An electric conductivity sensor will be used to measure the concentration of nutrients. This sensor also measures water temperature, as the degree of conductivity is highly dependent on the water temperature. Temperature and humidity should be measured in the plant incubator.

The user should be able to control exposure time, colour and intensity of the lights. All this customization will be done using a touch screen display with a GUI (graphical user interface), mounted on the top.

Theory

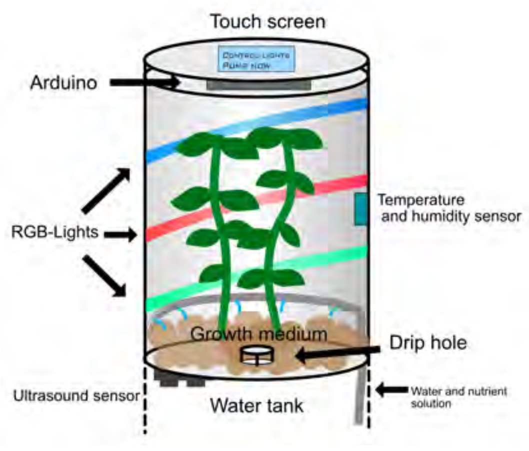
This chapter presents theory about the hydroponic system and the parameters that will be controlled and optimized. This is presented to give insight in how much impact each parameter has on plant growth and in what way they affect different aspects of the plant growth.

2.1 Hydroponic systems

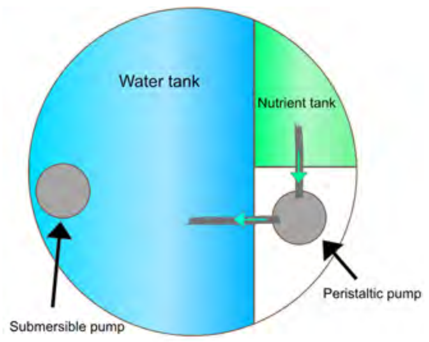
A hydroponic system grows plants without the use of soil - instead, a growth medium such as LECA (lightweight expanded clay aggregate) is used to hold the plants and the roots. When growing plants on land or in pots, water and nutrients are absorbed by the roots from the soil in which the plants grow. In hydroponic systems, all needed nutrients are available in the water that flows through the growth medium and is absorbed by the roots[3].

There are many variations of hydroponic systems, where the different hydroponic systems ranges from manual to fully automatic. The main difference comes from the method of watering and adding nutrients to the plants [3].

The method used in this project is a flow-system. Water with dissolved nutrients is pumped to the root of the plants from a tank just below the growing medium, for example every third hour. The water solution then runs down in the growth medium, allowing the plants to use as much or as little water and nutrients as they need. Any excess water drips down back into the tank, and is reused, thus requiring less water than conventional growing[1]. A sketch of this projects setup is presented in Figure 2.1.



(a) See-through of the system with components.



(b) Water tank seen from above.

Figure 2.1: Sketch of the system made in Affinity Designer [4].

2.2 Lighting

Common in all methods of growing is the need of light, either sunlight or artificial light such as LED (Light Emitting Diode). For plants to grow strong and produce leaves, flowers and other outcome, they must perform photosynthesis. Light in conventional farming comes from the sun, which contains all wavelengths in the photosynthetic spectrum, where PAR (photosynthetic active radiation) occurs in wavelengths between 400 - 700nm[5].

There are several factors in lighting that influences plant growth, one such is light intensity. Light intensity is usually measured in Lm (Lumen) and is defined as the quantity of light flowing over one second[6]. Different plants need different intensity, for example, growing fruits gives best possible outcome with higher light intensity[7]. Plants grown in low light intensity are usually longer with smaller leaves, plants grown in high light intensity is shorter with a stronger stem, more branches and bigger leaves[8].

Another factor is light exposure i.e, the total amount of light the plant is exposed to during a certain period of time. Light exposure differs in different parts of the world, and certain plants have different light exposure needs[8]. What light colour used is also an important parameter. The main colours used in growing lights are red (600 - 700nm) and blue (400 - 500nm) but plants make use of all visible light, and all different colours are vital for plant growth. Different stages of plant growth requires different lights, as some parts of the spectrum are of more important in stem growth, others in the flowering stage and so on. For example, flowering is triggered more by wavelengths in the red and orange spectra[9].

2.3 Nutrition

The nutrients added in the water play a major role in the outcome of growing plants. There are two major groups of nutrients needed for plants, macro and micro, the difference between these two groups are the amount needed. Macro nutrients are needed in larger amounts. Both groups are essential for normal plant growth. Some nutrients can be added as a separate liquid, while some other occurs naturally in air and water[10]. Presented in Table 2.1 are the essential nutrients listed in each group.

Table 2.1: Macro & micro nutrients with their respective chemical symbols.

Macro nutrients	Micro nutrients
Carbon (C)	Iron (Fe)
Hydrogen (H)	Manganese (Mn)
Oxygen (O)	Zinc (Zn)
Nitrogen (N)	Boron (Bo)
Phosphorus (P)	Molybdenum (Mo)
Potassium (K)	Chlorine (Cl)
Sulfur (S)	Copper (Cu)
Calcium (Ca)	Nickel (Ni)
Magnesium (Mg)	

Each nutrient has a specific role in plant growth, for example carbon, hydrogen and oxygen make up most of the organic compound[11].

The nutrient concentration is measured by the EC (electric conductivity) in the water, as nutrients affects the liquids ability to conduct electric charge. EC is also affected by the water temperature, as ion activity in nutrients increases with the temperature, which increases the ability to conduct electric charge[12].

Too much or too little of any nutrient can leave the leaves and stems with deficiencies. In this project a suitable nutrition made by the company Botanium will be used, it contains all the necessary nutrients as it is made for hydroponic systems[13]. According to Botanium, 5 ml of nutrients should be added to their water tank, this will be used to get the appropriate amount of nutrients in this projects water tank[14].

2.4 Air temperature and humidity

As different plants around the world grow in a wide variety of temperatures and different levels of humidity, these two factors play a major role in plant growth. For example, a study of spinach grown in 5°C and 16°C shows a significant difference in growth time and plant attributes. The plants grown in 16°C were fully grown in 32 days, compared to 5°C which took 92 days. Even though plants in warmer temperatures grew faster, dry weight and leaf size was larger for the plants grown in lower temperature[15]. Higher water temperatures directly affects the EC, as discussed in Chapter 2.3.

Humidity influences photosynthesis, plant growth and development. As with temperature, different plants and herbs have different optimal humidity levels. One major factor is the change of transpiration levels in different levels of humidity. Transpiration is the release of water vapor, for example, lower levels of humidity increase water loss from leaves. Transpiration also increases with higher temperatures, which makes temperature and humidity strongly connected[16].

Methodology

The hydroponic system consists of three parts - hardware, software and chassis. These three parts will be explained in this chapter.

3.1 Equipment

In this section, all parts are presented.

3.1.1 Microcontroller

The microcontroller used was an Arduino UNO, shown in Figure 3.1. It is based on the ATmega328P 8-bit microcontroller. It has 14 digital input/output pins, 6 analog inputs, I2C and SPI [17].

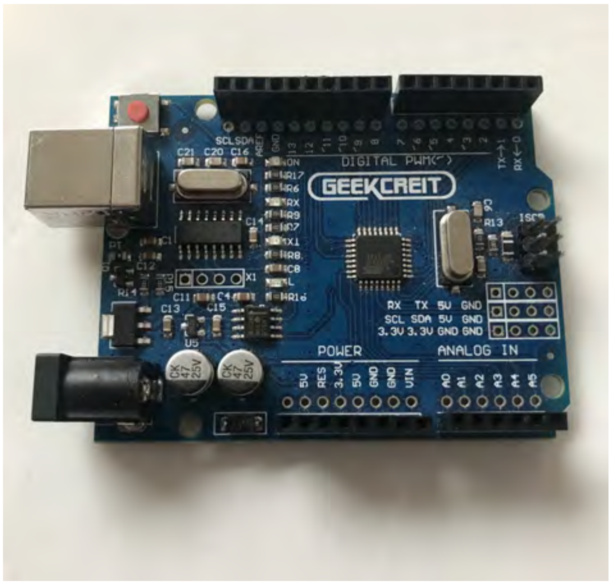


Figure 3.1: The Arduino Uno microcontroller, photo taken by Gustav Lindstrand.

3.1.2 Sensors

Several sensors are used in order to monitor the different factors affecting growth. They were all connected to the Arduino with the possibility of viewing the sensor values on a touch screen GUI.

TDS sensor

The TDS, or Total Dissolved Solids, sensor - shown in Figure 3.2 - is used to measure the concentration of nutrition in the water tank. It works by measuring the EC of the water. Since the EC increases with a higher concentration of nutrition, measuring the conductivity allows translation to nutrition levels.

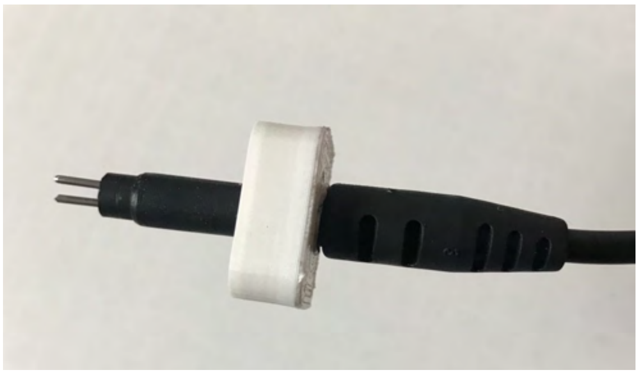


Figure 3.2: The TDS-sensor with 3D-printed holder, photo taken by Jonathan Ling.

Air temperature and humidity sensor

The temperature and humidity sensor used is of the type DHT11, see Figure 3.3. It has a digital serial interface and is supplied by 5VDC, with a temperature accuracy of $\pm 2°C$ between $0 - 50°C$. The relative humidity has an accuracy of $\pm 5\%$ in the range $20 - 90\%$[18]. It was installed on the inside wall of the chassis, right next to the plant. It is used to simultaneously measure humidity and temperature.

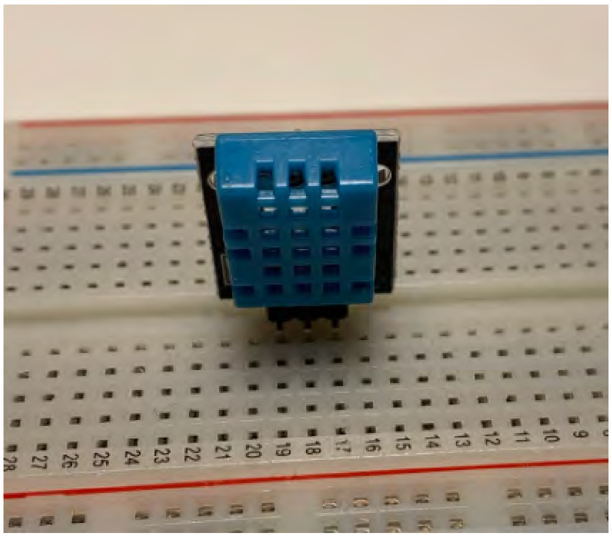


Figure 3.3: The temperature and humidity sensor, photo taken by Gustav Lindstrand.

Water lever sensor

The water level needs to be monitored closely in order to calculate the water volume, which is needed for nutrition dispensing. To measure the water level, an ultrasonic sensor, shown in Figure 3.4, was used. This is however not a water proof sensor. Therefore, it was placed above the water level but under the drip plate, which is where the water will drip back into the water tank. For this reason, a protective, dry holder for the sensor was added to the chassis, from which the sensor can measure the water level without getting wet.

The type of ultrasonic sensor which was used is the HC-SR04 with a measurement resolution of 0.3 cm and detection distance of 2 to 450 cm [19]. In order to calculate the volume, the area of the water tank is multiplied with the water level, which is calculated by subtracting the height of the water tank from the reading of the ultrasonic sensor.

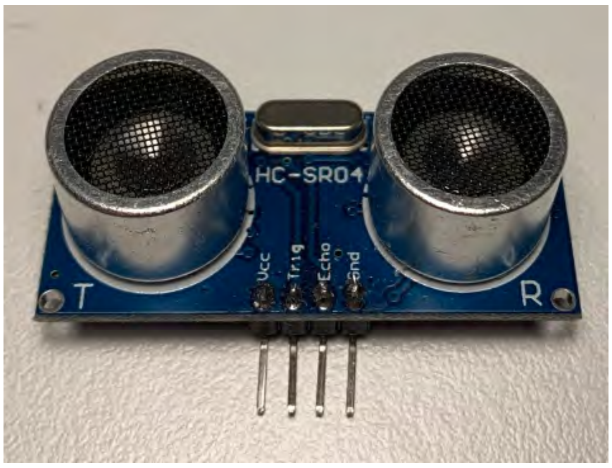


Figure 3.4: An ultrasonic distance sensor which will be used to measure water level, photo taken by Gustav Lindstrand.

3.1.3 Lamps

The lamps should preferably be LED-lights, as other types of lights are less effective, unable to provide the correct light spectrum and/or heat up too much to be used all the time [20]. The light that was used is a RGB (Red Green Blue) LED lighting strip, shown in Figure 3.5. The LED strip used is a one meter long addressable 5V RGB light strip, made by Luxorparts[21]. Every diode uses a maximum of 60mA and there are a total of 60 diodes which sums up to 3.6A. Addressable means that every single LED can be individually controlled in both intensity and colour. This gives many options in how the light can be controlled, for example, when seeds are planted only light in the lower regions of the tube is required. The strip was installed inside the tube, around the plant to provide lighting from all directions in the incubator. The light is addressable via digital outputs from the Arduino[22].

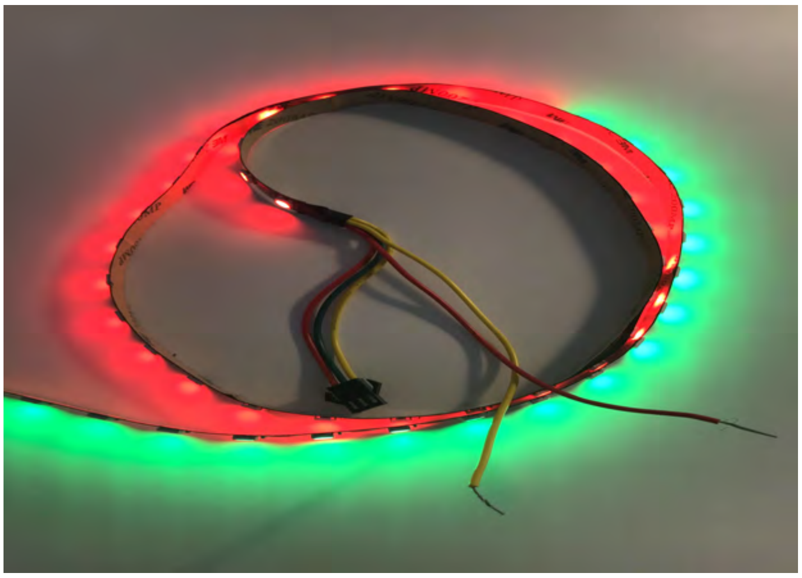


Figure 3.5: Addressable RGB Light strip, photo taken by Jonathan Ling.

3.1.4 Pumps

The pump, shown in Figure 3.6, which was used to pump up water to the plant, is a small submersible 5V 1A pump. It has an approximate water flow rate of 100 l/h [23].

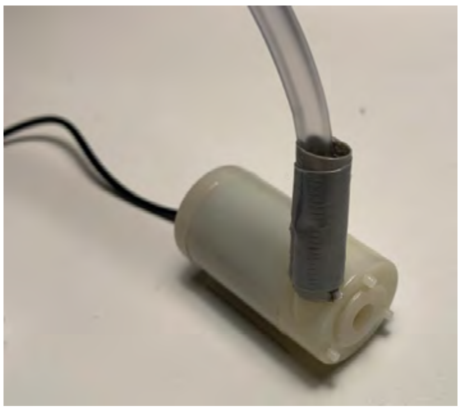


Figure 3.6: A 5V 1A water pump with 100 l/h flow rate, photo taken by Gustav Lindstrand.

For dosing of nutrients, a pump with a lower flow rate was needed. About 5 ml of nutrients is needed per litre of water. For this, a 6V 0.8A peristaltic pump with 39 ml/min flow rate was chosen, seen in Figure 3.7.

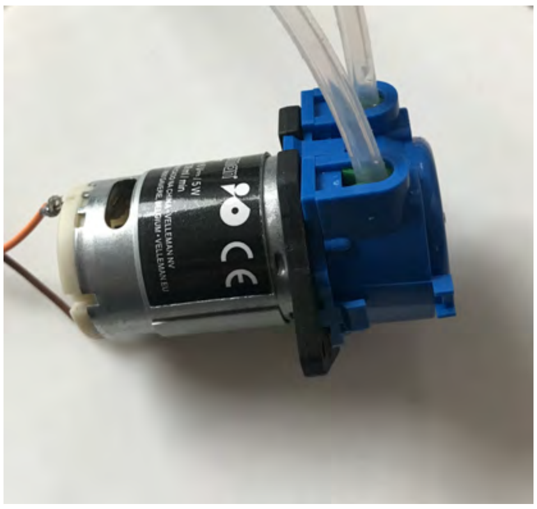


Figure 3.7: Peristaltic pump with 39 ml/min flowrate[24], photo taken by Gustav Lindstrand.

However, the pump was supplied with 5V instead as a 6V supply was not available. Tests showed that the pump works well with 5V, even retaining the pump rate of 39 ml/min.

3.1.5 Power supply

The power supply needs to power the Arduino and the rest of the electrical components. For this project, two standard 230VAC-5VDC (USB A) chargers were used. For the Arduino and the touch screen, a 5W one was used, and for the rest of the components a 10W charger was utilized.

Table 3.1: Peak power consumption for all components.

Component	Peak power usage	Which PSU is used?
Water pump	5W	10W
Lights	18W	10W
2.8” touch screen	1W	5W
Arduino Uno	0.2W	5W
Dosing pump	4.8W	10W
DHT11 Temp & humidity sensor	0.14W	10W
TDS sensor	0.03W	10W
Ultrasonic sensor	0.075W	10W

From Table 3.1 it is clear that the 5W charger was sufficient for powering the Uno and the touch screen, even at peak power consumption. For the other components, the total power consumption exceeds the supplied power. The sensors draw only a little power, less than 0.25W together. The main power consumers are the pumps and the lights. The solution to this was to program the lights to turn off as soon as the pumps are active. The only issue then would be that when both pumps are active, they might - along with the sensors - draw too much current together. However, the pumps were programmed to never be active at the same time - the nutrients are pumped right after the water. The lights, while they do require 18W to run on maximum intensity, only used as much current as is available and thus ran on about 9.5W.

3.1.6 Touch screen

The touch screen used was a 2.8 inch TFT touch screen with a built in shield. It is supplied with 3.3V and requires 300mA at most, but normally around 100mA[25].

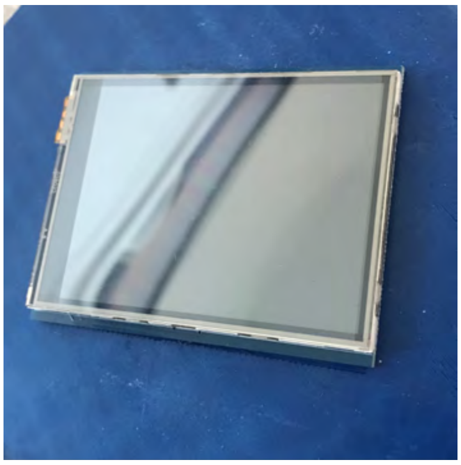


Figure 3.8: The 2.8” touch screen, photo taken by Gustav Lindstrand.

3.2 Hardware setup

In this section, installation, mounting and construction of all hardware will be described.

3.2.1 Cable management

The sensors, pumps, lights and touch screen is controlled by the Arduino Uno. This means that cables needed to be routed from each component, of which many are spread out all over the product. For this reason, cable containment is needed on the outside of the chassis, on the back. From here, the cables go from the very bottom of the water tank where the water pump is, to the top of the product where the Arduino and the touch screen are located.

3.2.2 Touch screen

The touch screen was mounted on a top lid, with jumper wires between the screen and the Arduino, which is placed below on a disc. The touch screen uses pins 8-13, two ground pins, the 3.3V pin and a 5V pin. It uses a resistive touch screen, which means that harder objects than a finger facilitates the usage of the screen, although fingers still work albeit less reliably.

3.3 Circuit diagram

In order to facilitate planning and installation of components, a circuit diagram was made, as seen in Figure 3.9. It shows which pins should be used for which function and which components should go to which place.

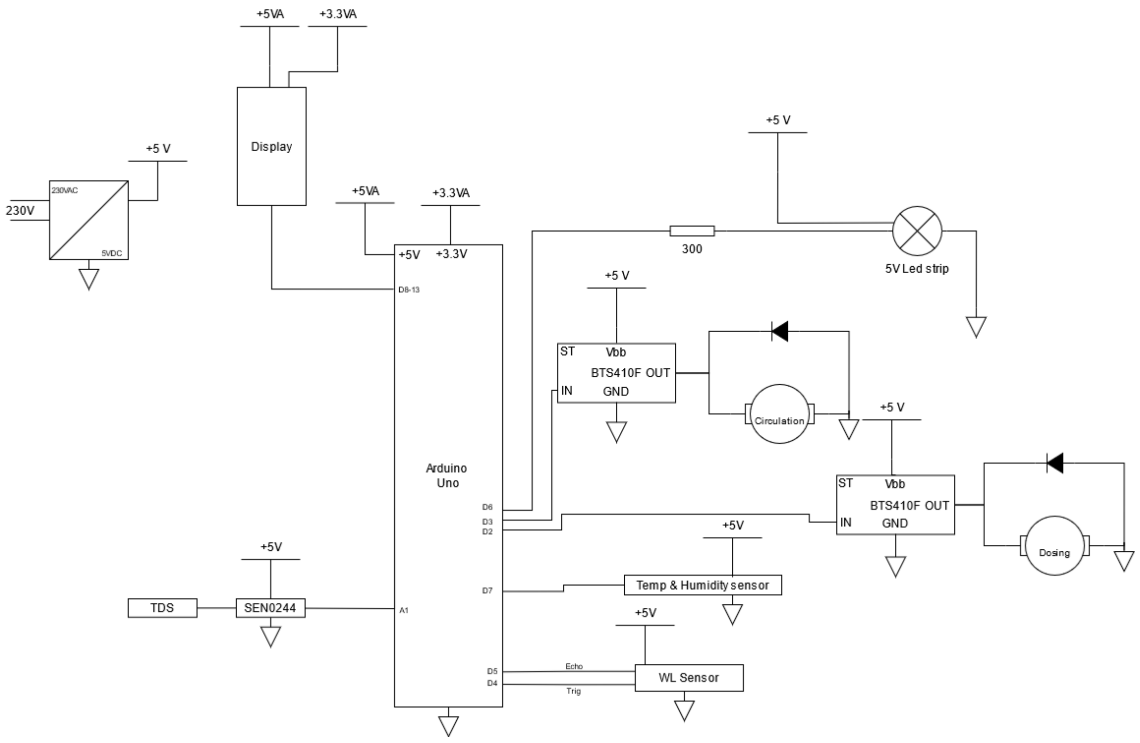


Figure 3.9: The wiring diagram for the project, made in draw.io[26].

The pumps and LED strip are controlled by a highside power switch (BTS410F). The switches are controlled by digital outputs from the processor board. Parallel with the inductive element - the pump - a flyback diode was installed. This protects the power switch from dangerously high voltages generated when the pumps are switched off. The TDS-sensor requires a driver board, SEN0244 [27]. It provides an interface for the signal.

3.4 Software setup

This section covers the methodology of the software, i.e. planning and structuring of the user interface and Arduino code.

3.4.1 Graphical user interface

The user interface on the touch screen consists of a menu of buttons, each linking to a sub-menu. The sub-menus that are presented as options are as follows:

- Water sensors
- Air sensors
- Pumps
- Lights

Each of these shows the current setting and/or value of sensor reading, as well as controls for the components that are controllable. The sub-menu ”Water sensors”, shows water volume and nutrient concentration. ”Air sensors” shows the temperature and humidity in the growth environment. ”Pumps” includes a button to pump water if needed. On ”Lights”, colours could be controlled as well scheduled hours per day that the lights should be on.

3.4.2 Programming the menu

While the screen uses libraries in order to draw shapes, text and lines, the menu was built from scratch. This allows for a higher degree of control of the menu and its structure.

The menu was built around sub-menus, each with an ID-number assigned to them. A sub-menu was, for example, the air sensors sub-menu, where the humidity and air temperature was shown. The main menu has 4 buttons, each linking to a certain sub-menu. Some sub-menus only have text, and some have text and buttons. Each button has its own function, which draws a rectangle and prints text on the correct place on the screen. The loop in the code then continuously looks at the touch input data to see if the x and y coordinates of the touch falls within the button’s area. If it does, and if the correct sub-menu is chosen (checked by comparing the ID of the current sub-menu with the ID of the button’s sub-menu), then the button’s ”hit”-function is executed. This function tells the program what to do when the button is hit, such as go to a new sub-menu, go back to the main menu or change light colour. When a new sub-menu is entered, the screen is first cleared by printing a white square on the whole screen. Then, the button functions for the sub-menu are executed.

3.4.3 Control systems

In order to keep a stable level of nutrition in the water tank, a simple feedback system was implemented, where readings from the TDS-sensor are compared to the water level to keep the nutrition in a certain range, decided in the TDS experiment described in ?? For example, if the nutrition concentration goes below 430 ppm, the nutrition pump is on for 0.5 seconds, if it goes way below the desired minimum level, it is activated for 2 s. The time that the nutrition pump is active is scaled according to the water volume - when 100% full, it pumps for 2 s while at 50% it pumps for 1 s.

3.5 Chassis

Parts of the chassis was 3D-printed using the material Polylactic acid, made completely out of renewable sources (bioplastic). It was made in CAD (Computer Aided Design) using Solid Edge[28]. An exploded version of a prototype of the chassis, with some electrical components added, can be seen in Figure 3.10.

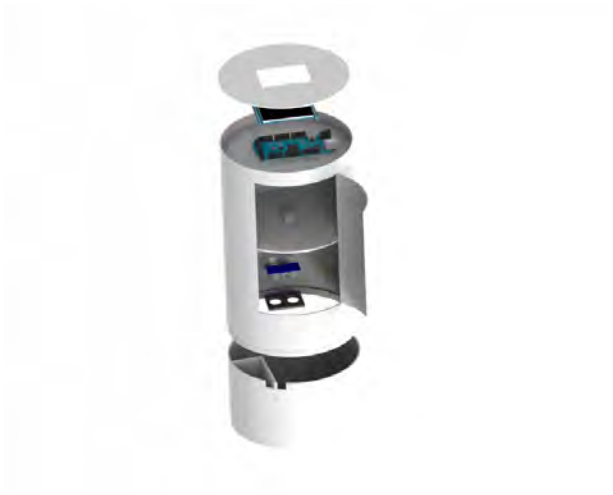


Figure 3.10: An exploded view of the prototype chassis along with some added components, created in Solid Edge [28].

The chassis consists of the parts shown in Table 3.2.

Table 3.2: Chassis parts for the whole construction.

Component	Source
Outer tube	Bought
Drip plate	3D-printed
Bottom plate with inner walls	3D-printed
TDS-holder	3D-printed
Holder for ultrasound sensor	3D-printed
Top lid for touch screen	3D-printed

3.5.1 Outer tube

The outer tube is made out of polyvinyl chloride (PVC), it is 500 mm long, has a outer diameter of 153 mm and inner diameter of 150 mm[29]. From this tube a door was cut out and screwed on to the tube to be able to enclose the growing area.

3.5.2 Drip plate

A drip plate was placed inside the tube, a growth medium was placed on top of the drip plate and from here, the plants will grow. In the middle of the drip plate there is a hole for the water to run through and be reused.

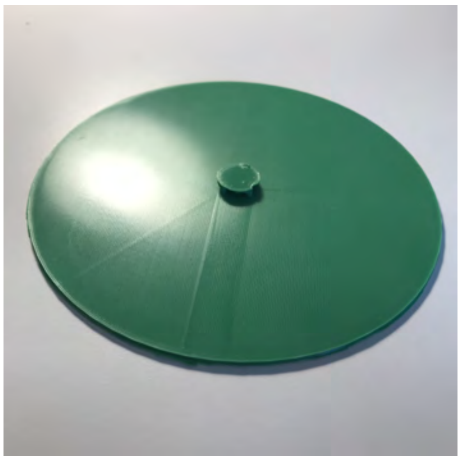


Figure 3.11: Top view of the 3D-printed drip plate, photo taken by Jonathan Ling.

3.5.3 Bottom plate with inner walls

This part was glued on to the bottom of the tube. It is divided into three parts; water tank, nutrient tank and a space for the peristaltic pump. On the inside wall, the TDS-sensor was placed to be able to measure the concentration of nutrients in the water.

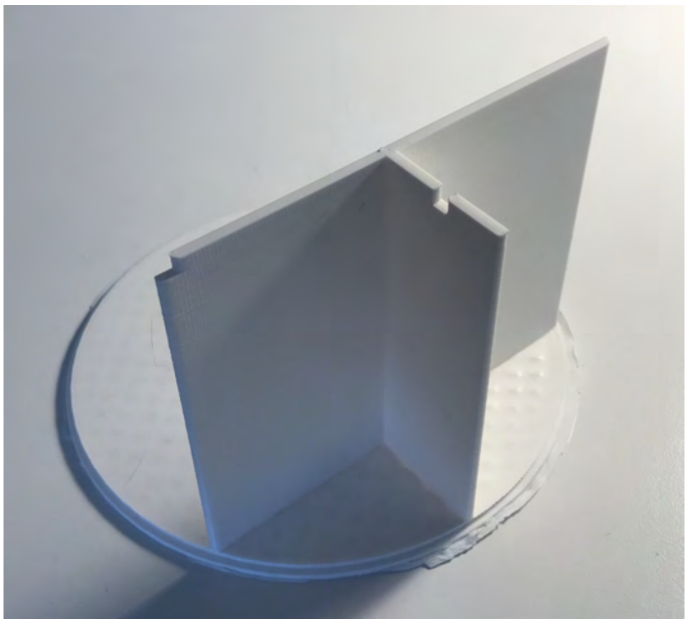


Figure 3.12: 3D-printed bottom plate, top view, photo taken by Jonathan Ling.

3.5.4 TDS-holder and holder for ultrasonic sensor

These parts were 3D-printed and holds the respective sensor, both sensors was placed in the water tank area. The TDS-holder can be seen in Figure 3.2 and the ultrasonic sensor mount can be seen in Figure A.3.1.

3.5.5 Top lid for touch screen

On top of the tube, a lid - seen in Figure 3.13 - was placed with a hole cut out for the touch screen. The Arduino was placed below this lid, since the touch screen was directly connected to the Arduino.

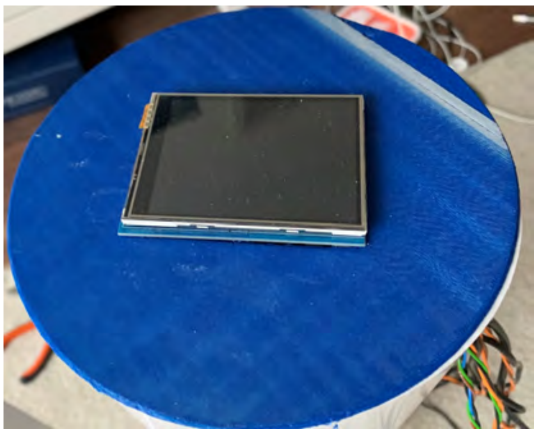


Figure 3.13: 3D-printed top lid, photo taken by Gustav Lindstrand.

3.6 Experiments

In this chapter, experiments and simulations are presented. These provide valuable information for designing the incubator, as certain parameters in this project can be initially investigated, and thus determine how these parameters will be designed.

3.6.1 Growth experiment - proof of concept

As a first test of just how effective an hydroponic system is, a first iteration of the product was made using a hydroponic system sold by Botanium[14]. The product pumps water every third hour and the nutrient is added manually to the water tank. The product was modified with a cardboard shell and a lamp to mimic the growth conditions of this project. In addition, the humidity and temperature sensor DHT11 was placed inside the shell. The results of this initial experiment can be found in Chapter 4.1.1.

3.6.2 Simulation in simulink

In order to test the stability of the nutrient concentration, a simulation model was made in Simulink[30]. The simulation is centered around two integrator blocks: mg of nutrition in the water and the volume of water. These two can be divided in order to calculate nutrition concentration. Inputs to the system are evaporation rate and signals of refilling the water tank. In the system model, the water tank is refilled at a random point, as well as when the water volume goes below a certain volume. Evaporation rate is dependent on the room temperature, which is generated stochastically, as well as the plant height. The goal of the model is to simulate the pumping of nutrients in the water at this point, as the concentration will decrease when new water is added. The results of this experiment can be found in chapter 4.1.2 and the image of the simulation structure can be found in Appendix A.2.

3.6.3 TDS experiment

In this experiment, the TDS sensor was connected to the Arduino and the probe was put in a container of water. The amount of nutrients in the glass of water was varied and measured, both before and after stirring, to see how the amount of nutrients and stirring change the TDS readings. Readings were also made in different places in the container to see how this affects the readings. Finally, the container was left over night to see how the nutrients were dissolved over a longer period of time. The purpose of the experiment is mainly to get a desired value or a desired range of values for the TDS sensor.

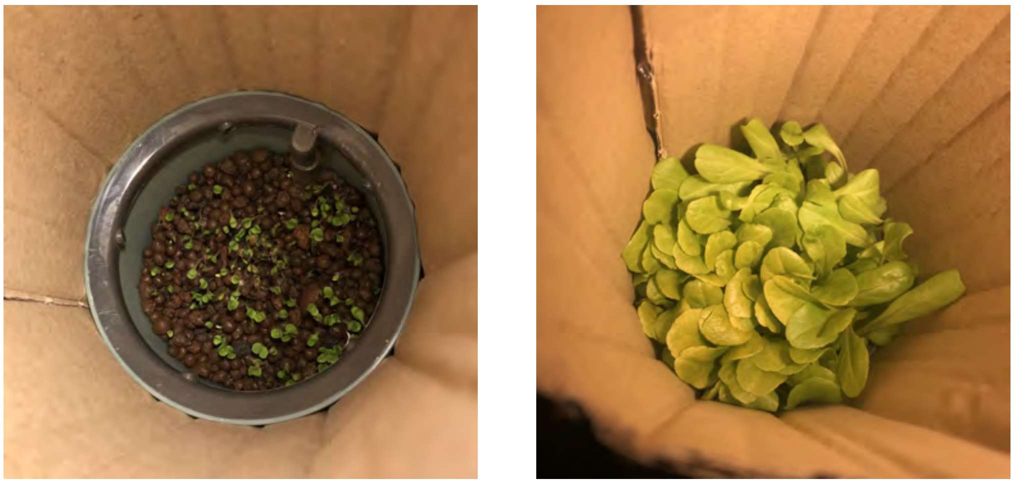
Results

4.1 Results of experiments

This section presents the results of the experiments conducted in Section 3.6.

4.1.1 Result of growing experiment

In this test, the salad *Baby leaf*[31] was grown from seed. The 2.6W lamp was on 24 hours a day. According to the producer of the seeds, the growth time takes up to 55 days from planted seed to harvested plant, in this test it took 23 days until leaves were harvested. The results during the first 13 days are shown in Figure 4.1.



(a) 2nd of february, 2021.

(b) 11th of february, 2021.

Figure 4.1: Initial growth experiment, photos taken by Jonathan Ling.

The humidity and temperature sensor showed a temperature of around 19°C and 15% moisture in the surrounding room, and around 26°C and 45% moisture inside the shell, which tells that the enclosed area is effective in keeping heat and moisture.

4.1.2 Results of simulation

As seen in Figure 4.2, the system responds well to the disturbance in the form of refilling the water tank. The nutrient concentration reaches the desired concentration level quickly.

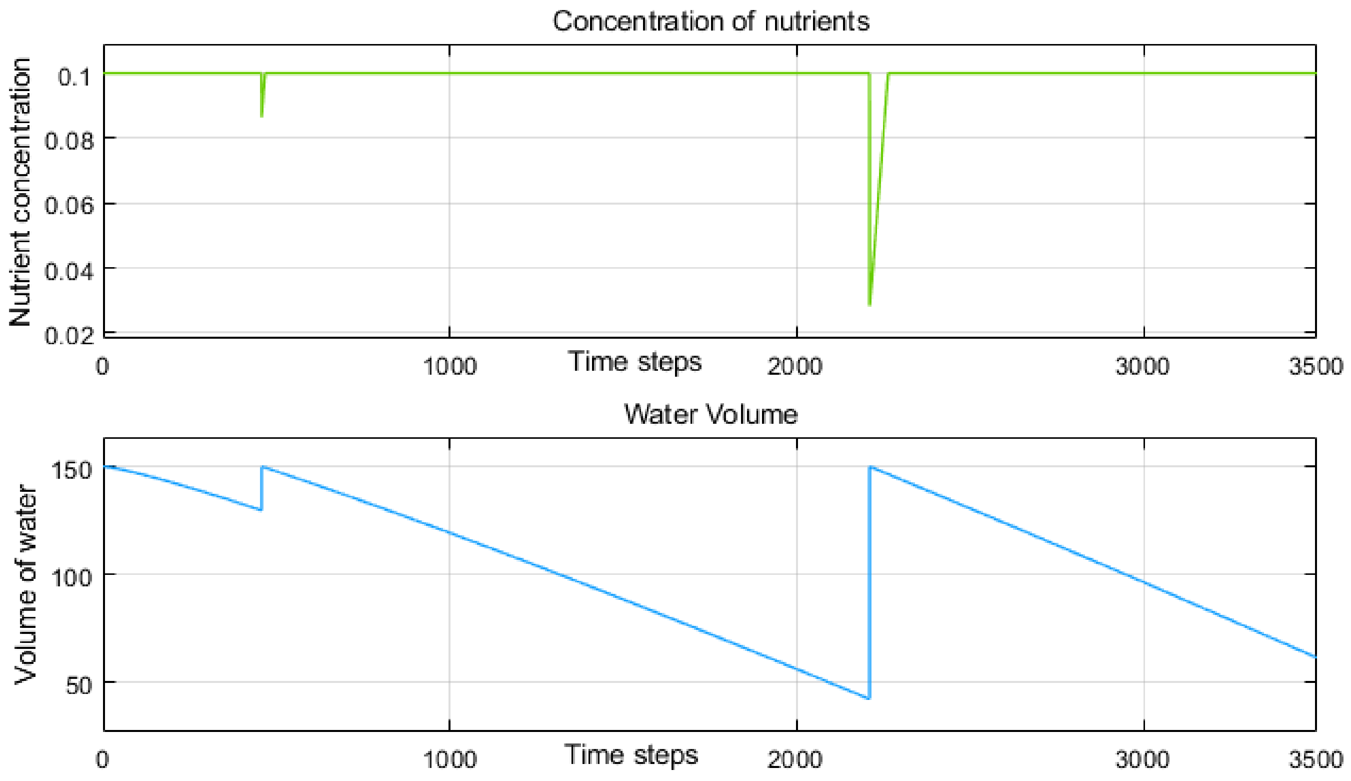


Figure 4.2: Graph over nutrient concentration (green) and water level (blue) in the simulation, created in Simulink [30].

4.1.3 Result of TDS experiment

As seen in Figure 4.3, the TDS value increased as more nutrients were added, and at least within these values the increase seems to be linear.

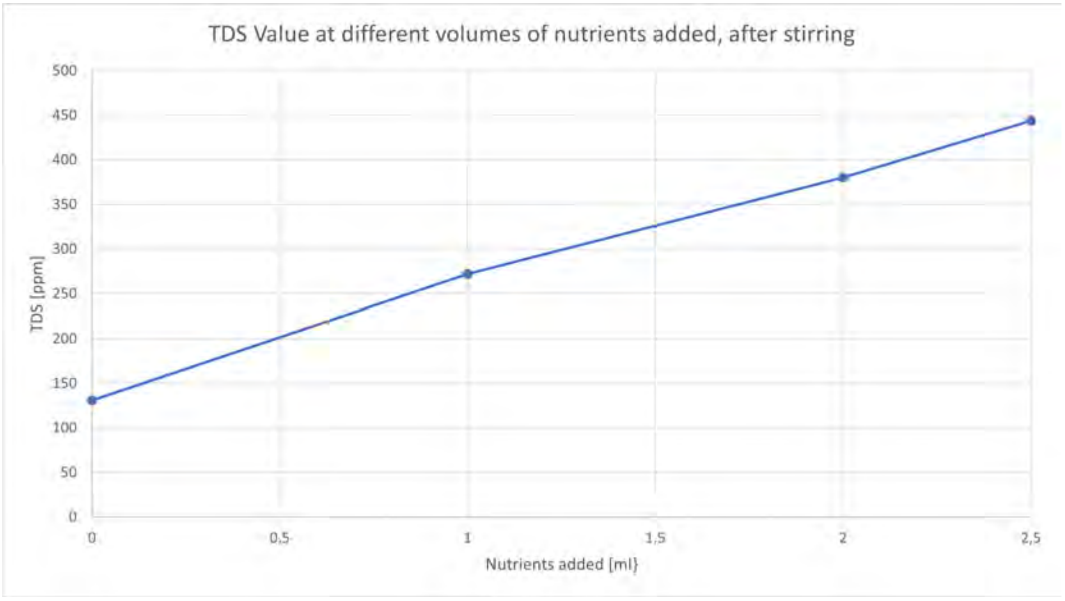


Figure 4.3: Graph showing the TDS readings at different volumes of nutrients added, after stirring. Made in Microsoft Excel.

For the 500 ml of water, the desired volume of nutrients is 2.5 ml according to the manufacturer. In this experiment, this seems to correspond to between 430-520 ppm, depending on where the probe was located and the degree of stirring that took place. As a desired value, 430 ppm will be chosen, as it might be relatively easy to overshoot this with the nutrient pump since achieving a detailed enough control of the exact volume of nutrients might be hard. Furthermore, it is always possible to increase the concentration by simply pumping more nutrients into the water, but decreasing the concentration cannot be done by the system autonomously which is why the desired value is put at the bottom of the allowed range.

The detailed log of the experiment can be seen in Appendix A.4.

4.2 Nutrients stability in the incubator

When 500 ml of tap water was added to the tank, the TDS-sensor showed 130 ppm, which triggered the pumps to run for 2 seconds, adding 1.2 ml of nutrition. Since the nutrition pump is only active after watering, which occurs every third hour, it cannot stabilize directly. After the first pump run, the nutrition level stabilized at around 370 ppm which still is not a desired value. For the second pump run, the nutrition pump ran for 0.5 seconds, adding 0.3 ml of nutrition. After around 10 minutes, the nutrition concentration was stabilized at 460 ppm. For the third pump run, the nutrition pump did not run as the concentration was above the desired level.

4.3 System integration test

This section covers the way which the user interface increases the ability to optimize the incubator.

4.3.1 Optimizability using the screen

The touch screen user interface, shown in Figure 4.4 provides an easy way for the user to customize the settings of the plant incubator, as well as get information about the status.

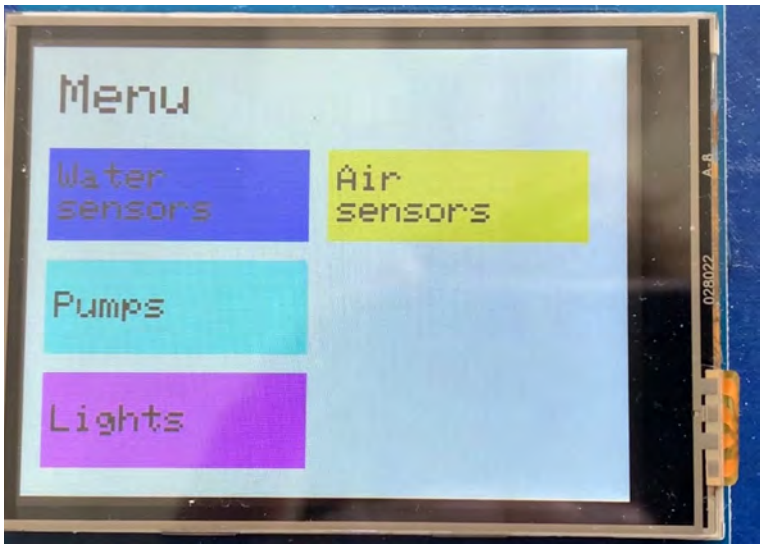


Figure 4.4: The main menu of the GUI used in the project, photo taken by Gustav Lindstrand.

The lights page allows the user to set the colour of the lights, as well as the schedule, i.e. the amount of hours per day that the lights should be turned on. The pump page allows for manual activation of the pumps, in case it is deemed necessary. The air sensors page shows data from the DHT11 sensor, i.e air temperature and humidity. If this is a growth factor that is being studied, it is highly useful to be able to know these data points. The water sensor page shows the TDS and remaining water volume. This shows if the water has the correct concentration of nutrients as well as the remaining water volume, indicating that refilling of the water tank might be necessary.

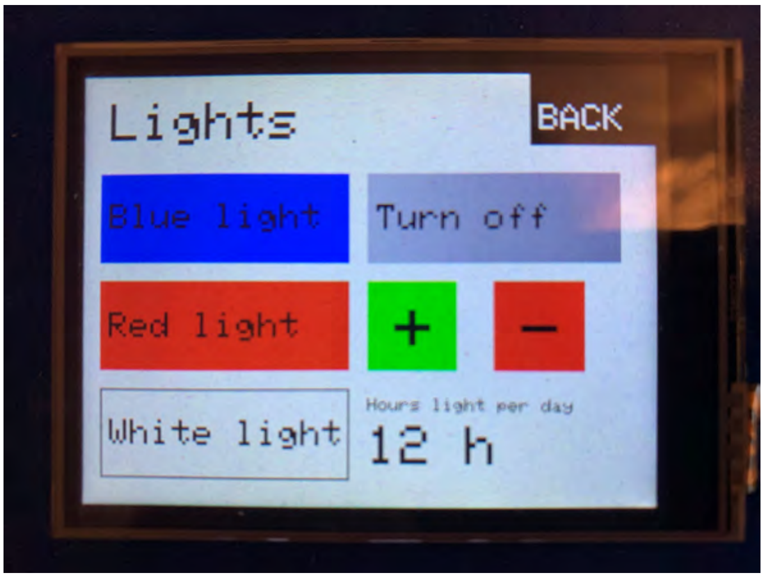


Figure 4.5: The sub-menu for the control of the lights, photo taken by Gustav Lindstrand.

In Figure 4.5 the GUI for a sub-menu is shown. The back button returns to the main menu, the light button changes the light of the LEDs and the plus and minus buttons increase or decrease the hours of light per day that the plant will get. The other sub-menus can be seen in Appendix A.3.3.

Discussion

5.1 Analysis of nutrient stability

As mentioned previously, the nutrition level is measured with the TDS-sensor. During an interview with the company Botanium[14], we recieved a TDS-sensor as a gift. This sensor had a built in temperature sensor which is needed to calculate the amount of nutrients in the water. However, they did not have a manual for this and nothing could be found by searching the internet. Because of this, a new TDS-sensor without temperature sensor was ordered. This made nutrient readings less reliable. Despite this, the implemented system with nutrition readings are accepted as sufficient to keep a somewhat good level of nutrition concentration, since the temperature will be kept at room temperature for the majority of the growth time.

5.2 Efficiency of lights

Plants take time to grow, and with a limited amount of time for this project, testing of different light intensities, colours and exposure has been hard to thoroughly examine. In order to test the lights thoroughly, many months would be needed as not only light affects growth. Seeds, nutrition and temperature, to mention some, have impact on the results, but with the results from the initial experiment along with the presented theory, it is concluded that plants grow well with many different types of lights.

5.3 Construction

This incubator was constructed to be easily handled and have likeliness with a finished product, which is why a compact, integrated system was built. As mentioned in Chapter 3.5, many parts were 3D-printed. This has been vital for this project's time management. Some 3D-printed parts hold water and assumptions were made they would not leak, this was not the case. Because of this, a silicone seal was needed to keep the water in place. This was made in many iterations and was a time consuming part of this project. Figuring out component placement in a small containment has also been a relatively difficult problem to solve. The tube used as the outer layer was not perfectly round, this made the assembly of the bottom plate much harder and more sealant was needed to make sure leakage did not occur.

In general, the construction is working well. Making a compact product is achieved without excluding vital parts. What could be a possible problem in the long-term is moisture damages to some components, as well as further leakage.

5.4 Possible error sources

The ultrasonic sensor has a low resolution. For this project's system, a higher resolution or another way to measure the water level would be more suitable since the water tank is small and need to be precisely measured to be able to measure the water volume more accurately.

The air and humidity sensor, DHT11, is difficult to calibrate since using another thermometer or humidity sensor to compare requires high standard instruments, something that was not possible in this project. The high degree of error is also a problem which no calibration can solve. For hobby use, it is likely good enough, but for industrial research a better temperature and humidity sensor would be needed.

As mentioned in Chapter 5.1, the TDS-sensor used does not have a temperature sensor, making readings less reliable as temperature affects EC. Monitoring and controlling of pH-value was left out, the influence of pH-value on the nutrient concentration is thereby unknown, and is a possible error source.

Conclusion

6.1 Addressing purpose

The conclusion of this study is that the majority of the parameters controlled to ensure a high order of optimizability is present within the boundaries of this study but could always be built upon further. It is a matter of priorities, budget and time whether additional factors are needed or possible. With the system present today, light intensity, exposure time and colour could be controlled. In addition to that, humidity, temperature and water level can be closely monitored and somewhat controlled. These parameters are deemed necessary to control to keep a high order of optimizability.

With the system built in this study, it is possible to experiment with light intensity, exposure time and different colours. As mentioned, plants take some time to grow and the experiments are only limited by time. The initial experiment presented in Chapter 4.1.1 showed that salad could be grown with an ordinary 2.6W bed lamp. To test, for example, three different settings of lights thoroughly would take a minimum of three months, a time this project did not have.

6.2 Future studies

As mentioned in the discussion, there are many different parameters in light that affects plant growth. For future studies longer tests, different light intensities, exposure time and so on could be tested.

Regarding the major research question, the optimizability, there is more or less no end to how optimized a system could be. There are multiple factors that have been left out, such as control and measurement of pH-value. Temperature and humidity could be controlled to some extent in this project, this could be extended to fully controllable temperature and humidity by adding a fan and a heat source. Some things that would be relatively easy to add would be the ability to customize the desired nutrient concentration, pump schedule and uploading the sensor data to an SD-card for later analysis.

For a future project with a similar design as this, it would be recommended to use a different method of creating the water tank area, preferably molded in one piece e.g by 3D-printing or other methods, to avoid extra work with sealing and the risk of leakage. Cable handling is also an area where improvements can be made, since some interference was discovered at some points of time in the process of testing the incubator.

Bibliography

[1] G. Barbosa et al. “Comparison of Land, Water, and Energy Requirements of Lettuce Grown Using Hydroponic vs. Conventional Agricultural Methods”. In: *International journal of environmental research and public health* 12 (June 2015), pp. 6879–91.

[2] A. Trafton. *The future of agriculture is computerized.* 2019, [cited 16 Feb. 2021]. URL: `https://news.mit.edu/2019/algorithm-growing-agriculture-0403`.

[3] Jr. J.W. Bartok. *Hydroponic Systems.* 2009, [cited 16 Feb, 2021]. URL: `https://ag.umass.edu/greenhouse-floriculture/fact-sheets/hydroponic-systems`.

[4] Serif. *Affinity Designer 2020.* URL: `https://affinity.serif.com/en-gb/designer/`.

[5] Inc. Fondriest Environmental. *Solar Radiation and Photosynethically Active Radiation.* 21 Mar. 2014. URL: `https://www.fondriest.com/environmental-measurements/parameters/weather/photosynthetically-active-radiation/`.

[6] The editors of Encyclopaedia Britannica. *Luminous intensity. Encyclopedia Britannica.* 2017, [cited 18 Feb 2021. URL: `https://www.britannica.com/science/luminous-intensity`.

[7] H. Shirley. “The Influence of Light Intensity and Light Quality Upon the Growth of Plants.” In: *American Journal of Botany* 16 (1929), pp. 353–355.

[8] Drs. D. Wilkerson and C. Hall. *Light, Temperature and Humidity.* Year: Unknown, [cited 20 Feb, 2021]. URL: `https://aggie-horticulture.tamu.edu/ornamental/a-reference-guide-to-plant-care-handling-and-merchandising/light-temperature-and-humidity/`.

[9] A. Smestad. *The Effect of Light on Plant Growth.* 2017, [cited 20 Feb, 2021]. URL: `https://sciencing.com/the-effect-of-light-on-plant-growth-12201478.html`.

[10] *Essential Nutrients for Plants.* 6 Mar. 2021, [cited 20 May. 2021]. URL: `https://bio.libretexts.org/@go/page/13781`.

[11] E. Sanchez et al. *Hydroponics Systems and Principles Of Plant Nutrition: Essential Nutrients, Function, Deficiency, and Excess.* 2020, [cited 20 feb, 2021]. URL: https://extension.psu.edu/hydroponics-systems-and-principles-of-plant-nutrition-essential-nutrients-function-deficiency-and-excess.

[12] A. Shoukat, M. Hussain, and A. Shoukat. *Efects of Temperature on Total dissolved Solid in water.* Conference: Water quality study at Mehran University Sindh Feb. 2020, [cited 16 Feb. 2021].

[13] Botanium. *Nutrients.* 2021. URL: https://botanium.se/collections/frontpage/products/nutrients-300-ml.

[14] Botanium. *Botanium.* 2021. URL: https://botanium.se/collections/frontpage/products/botanium-kit.

[15] S.R. Boese and N.P.A. Huner. "Efect of Growth Temperature and Temperature Shifts on Spinach Leaf Morphology and Photosynthesis". In: 94.4 (1990), pp. 1830–1836. URL: http://www.plantphysiol.org/content/94/4/1830.

[16] T.W. Tibbitts. "Humidity and Plants". In: *BioScience* 29.6 (June 1979), pp. 358–363. eprint: https://academic.oup.com/bioscience/article-pdf/29/6/358/594530/29-6-358.pdf. URL: https://doi.org/10.2307/1307692.

[17] Arduino.cc. *Arduino Uno.* URL: https://store.arduino.cc/arduino-uno-rev3.

[18] Velleman nv. *DHT11 DIGITAL TEMPERATURE HUMIDITY SENSOR MODULE FOR ARDUINO®.* 2017. URL: https://www.kjell.com/globalassets/mediaassets/745018_87086_manual_en.pdf?ref=A5605F6320.

[19] Velleman nv. *HC-SR05 ULTRASONIC SENSOR.* 2017. URL: https://www.kjell.com/globalassets/mediaassets/745226_87059_manual_en.pdf?ref=71FE6689C4.

[20] *Advantages of LED Lights.* 27 Oct. 2020, [cited 20 May. 2021]. URL: https://www.thelightbulb.co.uk/resources/ultimate-guide-led-lights-advantages-leds/.

[21] Kjell Company. *Luxorparts Adresserbar RGB LED-list 1 m.* 2021. URL: https://www.kjell.com/se/produkter/el-verktyg/arduino/arduino-tillbehor/luxorparts-adresserbar-rgb-led-list-1-m-p87998.

[22] LTD DONGGUANG OPSCO OPTOELECTRONICS CO. *Intelligent Control LED Integrated Light Source.* URL: http://www.farnell.com/datasheets/2329775.pdf.

[23] Electrokit Sweden AB. *Vattenpump mini 5V.* URL: https://www.electrokit.com/produkt/vattenpump-mini-5v/.

[24] Electrokit Sweden AB. *Vätskepump peristalisk 6V.* URL: https://www.electrokit.com/produkt/vatskepump-peristalisk-12v-silikonslang/.

[25] lady ada. *Adafruit 2.8" TFT Touch Shield v2 - Capacitive or Resistive*. 2018. URL: `https://www.elfa.se/Web/Downloads/_t/ds/1651_eng_tds.pdf`.

[26] Seibert Media. *Draw.io*. URL: `https://drawio-app.com/`.

[27] DFRobot. *SEN0244 Datasheet*. URL: `https://www.digikey.se/htmldatasheets/production/2799469/0/0/1/sen0244.html`.

[28] Siemens. *Solid Edge 2020*. URL: `https://solidedge.siemens.com/en/`.

[29] Amazon. *Ventilationsrörplatta ø 150 0,5 M Awenta*. 2021. URL: `https://www.amazon.se/gp/product/B012VRR570/ref=ppx_yo_dt_b_asin_title_o00_s00?ie=UTF8&psc=1`.

[30] The Math Works. *Matlab® & Simulink®*. URL: `https://se.mathworks.com/products/simulink.html`.

[31] Nelson Garden. *Sallat, Plock- 'Salad Bowl'*. URL: `https://www.nelsongarden.se/swe/sek/p/gronsaker_120/sallat-plock-_91321`.

Appendix

A.1 Arduino Code

```cpp
//Code for the Optimizable Hydroponic Plant Incubator
//Names: Gustav Lindstrand and Jonathan Ling
//TRITA-ITM-EX 2021:48
//Bachelor's thesis at ITM in Mechatronics
//The Royal Institute of Technology in Stockholm
//2021-05-20
//This code creates the graphical user interface with all sub menus. The
/////////////////////////////////
//////File 1 - Main file/////////
/////////////////////////////////

#include <DHT.h>
#include <DHT_U.h>
#include <Adafruit_GFX.h>
#include <SPI.h>
#include <Wire.h>
#include <Adafruit_ILI9341.h>
#include <Adafruit_STMPE610.h>
#include <Adafruit_NeoPixel.h>
// This is calibration data for the raw touch data to the screen coordinates
#define TS_MINX 150
#define TS_MINY 130
#define TS_MAXX 3800
#define TS_MAXY 4000

#define FRAME_X 10
#define FRAME_Y 10
#define FRAME_W 200
#define FRAME_H 50

#define TEXT_X 10
#define TEXT_Y 50
//These are coordinates and dimensions for different buttons in the menu
#define BACKBUTTON_X 250
#define BACKBUTTON_Y 0
```

```
#define BACKBUTTON_W 90
#define BACKBUTTON_H 40

#define WSENSBUTTON_X 10
#define WSENSBUTTON_Y 55
#define WSENSBUTTON_W 140
#define WSENSBUTTON_H 50

#define ASENSBUTTON_X 160
#define ASENSBUTTON_Y 55
#define ASENSBUTTON_W 140
#define ASENSBUTTON_H 50

#define INCREASEBUTTON_X 160
#define INCREASEBUTTON_Y 115
#define INCREASEBUTTON_W 50
#define INCREASEBUTTON_H 50

#define DECREASEBUTTON_X 230
#define DECREASEBUTTON_Y 115
#define DECREASEBUTTON_W 50
#define DECREASEBUTTON_H 50

#define PUMPBUTTON_X 10
#define PUMPBUTTON_Y 115
#define PUMPBUTTON_W 140
#define PUMPBUTTON_H 50

#define LIGHTBUTTON_X 10
#define LIGHTBUTTON_Y 175
#define LIGHTBUTTON_W 140
#define LIGHTBUTTON_H 50

#define PUMPTEST_X 30
#define PUMPTEST_Y 80
#define PUMPTEST_W 200
#define PUMPTEST_H 80

#define DHTTYPE DHT11 //Specifying the type of DHT temperature & humidity sensor
#define STMPE_CS 8 //Pin information for the touch screen
Adafruit_STMPE610 ts = Adafruit_STMPE610(STMPE_CS); //Pin information for the touch screen
#define TFT_CS 10 //Pin information for the touch screen
#define TFT_DC 9 //Pin information for the touch screen
Adafruit_ILI9341 tft = Adafruit_ILI9341(TFT_CS, TFT_DC); //Pin information for the touch screen
//Input pins
#define TdsSensorPin A1 //defining pin for reading analog data from the TDS Sensor
#define DHTPIN 7 //Pin for temp and humidity sensor
#define LEDPIN 6 //Pin for controlling the light colour
#define ECHOPIN 5 //Echo pin for ultrasonic sensor
#define TRIGPIN 4 //Trig pin for ultrasonic sensor
```

```
#define WPUMPPIN 3 //Pin which controls the switch to the water pump
#define NPUMPPIN 2 //Pin which controls the switch to the nutrient pump
#define N_LEDS 60 //Amount of leds in the light strip
#define VREF 5.0 //analog reference voltage(Volt) of the ADC
#define SCOUNT  30 //sum of sample point
long duration; //variable for the duration of sound wave travel
float distance; //variable for the distance measurement
int analogBuffer[SCOUNT]; //store the analog value in the array, read from ADC
int analogBufferTemp[SCOUNT]; //temp buffer for TDS calculations
int analogBufferIndex = 0,copyIndex = 0,temperature = 25; //Starting variables for TDS calculations
float averageVoltage = 0,tdsValue = 0;
DHT dht(DHTPIN, DHTTYPE); //Initializing the Temp and humidity sensor
Adafruit_NeoPixel strip = Adafruit_NeoPixel(N_LEDS, LEDPIN, NEO_GRB + NEO_KHZ800); //Initializing the led strip
int hoursLightPerDay = 12; //default value for hours of light per day
int lightColor = 3; //0 = lights off, 1=blue, 2=red, 3=white
bool lightsOn = false; //True if lights are on, false if off
unsigned long wPumpPreviousMillis = 0; //variable for counting milliseconds for water pump schedule
unsigned long lightPreviousMillis = 0; //variable for counting milliseconds for light schedule
bool measureTDS = false; //TDS should only be measured directly after pumping
bool wPumpOn = false; //True if water pump is on
bool nPumpOn = false; //True if nutrient pump is on
bool unableToTurnOnLights = false; //Turns on if scheduled lights were not activated due to pumps being on
int minTDSValue = 430; //Minimum PPM value which should be achieved in the water mix
unsigned long pumpIntervals = 3*3.6e6; //Intervals between water pumping. 3.6e6ms=1h
int pumpTime = 20E3; //Time that the water pumps should pump
int timeToWaitForTDSAfterPump = 60E3; //Time to measure TDS after pumps have started to pump
//Variables from sensors
float waterVolume = 0; //Water volume
short menuMode = 0; //Number which tells which menu you're in

void setup(void)
{
  Serial.begin(9600); //Setting baud rate for serial communication
  dht.begin(); //Starting dht measurement
  tft.begin(); //Starting tft
  ts.begin(); //Starting touch screen
  strip.begin(); //Start the led strip
  tft.fillScreen(ILI9341_WHITE); //Filling the screen with a white square
  tft.setRotation(1); //Setting the rotation of the screen
  wSensBtn(); //These function initializes the buttons for the main menu
  pumpBtn();
  lightBtn();
  aSensBtn();
  changeMenuText("Menu"); //Sets menu text
  pinMode(TdsSensorPin,INPUT); //Sets the tds sensor pin as input
  pinMode(TRIGPIN, OUTPUT); //Sets the trigPin as an OUTPUT
  pinMode(ECHOPIN, INPUT); //Sets the echoPin as an INPUT
  pinMode(WPUMPPIN, OUTPUT); //Sets the water pump pin as OUTPUT
  pinMode(NPUMPPIN, OUTPUT); //Sets the nutrient pump pin as OUTPUT
  Serial.println("Started");
}
void loop()
{
    touchInputHandler(); //Runs the touch input handler function which checks for touch data and activates buttons, if pressed.
    unsigned long currentMillis = millis(); //Set currentMillis to millis(), millis() is simply the amount of ms passed since arduino was initialized
    if((currentMillis - wPumpPreviousMillis >= pumpIntervals ||
currentMillis<1000) && measureTDS==false && wPumpOn == false) //If pumpInterval ms has passed and water pump is off and TDS is not measured OR it's the first time running, execute
    {
      Serial.println("Starting pumps");
      wPumpPreviousMillis = currentMillis; //reset the millisecond counter for the pumps
      startWPump(); //start pumping with the water pump
      measureTDS=true; //start measuring tds
      wPumpOn = true; //bool for indicating that the pumps are on
    }
    if (currentMillis - wPumpPreviousMillis >= pumpTime && measureTDS==true && wPumpOn==true)//If pumpTime ms has passed and water pump is on and TDS is measured, execute
    {
      stopWPump(); //Stop water pumps
      Serial.println("Stopping pumps");
      wPumpPreviousMillis = currentMillis; //Reset ms counter
      wPumpOn=false; //indicate that pumps are off
    }
    if (currentMillis - wPumpPreviousMillis >=
timeToWaitForTDSAfterPump && measureTDS==true && wPumpOn==false) //If timeToWaitForTDSAfterPump ms has passed and water pump is off and TDS is measured, execute
    {
      measureTDS=false; //Stop measuring tds
      Serial.println("Stopping measurement of TDS");
      wPumpPreviousMillis = currentMillis; //reset ms counter
      //If the desired TDS is not high enough in the water, pump more nutrients. 2s is about equal to 1.2 ml, so 4s of pumping is what is needed for the full water tank to reach the correct concentration of nutrients.
      //However sometimes the first seconds or two it is possible that not enough nutrients gets pumped which would then be resolved at the next pump interval.
      if(tdsValue<minTDSValue && tdsValue>minTDSValue-130)
      {
```

```
      feedNutrients(waterVolume*50); //function for pumping nutrients a little less, since the tdsValue is close to be reached. At full water tank, this correspons to 500 ms.
    }
    else if(tdsValue<minTDSValue-250) //If measured TDS value is well below desired value, pump a little longer. At full water tank, this corresponds to 2.5s (2500ms)
    {
      feedNutrients(waterVolume*250);
    }
  }
  if (currentMillis - lightPreviousMillis >= hoursLightPerDay*60e3 && lightsOn) //Scheduling for the lights. On for hoursLightPerDay h, off 24-hoursLightPerDay h
  {
    lightPreviousMillis = currentMillis; //reset ms counter
    lightsOn = false; //Bool for indicating that lights are off
    Serial.println("Lights turned off");
    turnOffScheduledLights(); //turns off the lights
  }
  else if (currentMillis - lightPreviousMillis >= (24-hoursLightPerDay)*60e3 && lightsOn==false) //Scheduling for the lights. On for hoursLightPerDay h, off 24-hoursLightPerDay h.
  {
    lightPreviousMillis = currentMillis; //reset ms counter
    lightsOn = true; //Bool for indicating that lights are on
    Serial.println("Lights turned on");
    turnOnScheduledLights(); //turns on the lights
  }
  if(unableToTurnOnLights==true) //If lights were unable to be turned on due to pumps running, execute
  {
    unableToTurnOnLights=false; //Reset the unable to turn on lights bool (if pumps are still on, this will return to true in the function below
    turnOnScheduledLights(); //Tries to turn on lights again, if it does not work then "unableToTurnOnLights" will return to being true
  }

  if(measureTDS==true)
  {
    TDSMeasurements(); //Function for measuring the TDS
  }
  if(menuMode==1)
  {
    updateVolume(getWVolume()); //Updates the water level
    updateNT(tdsValue); //Updates the nutrient concentration
  }
  if(menuMode==4)
  {
    updateAT(dht.readTemperature()); //Updates air temp
    updateH(dht.readHumidity()); //Updates humidity
  }
}
```

```
/////////////////////////////////
////File 2 - Air sensor page/////
/////////////////////////////////
//menuMode = 4
void updateAT(float AT)
{
  tft.setCursor(TEXT_X+5, TEXT_Y); //Sets the text cursor at the text position
  tft.setTextSize(2); //Sets text size
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE); //Set text color and background color
  tft.print("Current air temperature"); //Prints text
  tft.setCursor(TEXT_X+5, TEXT_Y+20); //Sets the text cursor at the text position
  tft.print("is:"); //Prints text
  tft.setTextColor(ILI9341_BLUE,ILI9341_WHITE); //Set text color and background color
  tft.print(AT); //Print air temperature
  tft.print(" C"); //Prints text
}
void updateH(float H)
{
  tft.setCursor(TEXT_X+5, TEXT_Y*2);
  tft.setTextSize(2);
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE);
  tft.print("Current humidity level");
  tft.setCursor(TEXT_X+5, TEXT_Y*2+20);
  tft.print("is:");
  tft.setTextColor(ILI9341_BLUE,ILI9341_WHITE);
  tft.print(H);
  tft.print(" %");
}
```

```
/////////////////////////////////
//////File 3 - Lights page///////
/////////////////////////////////
//menuMode=3
void redBtn()
{
  tft.fillRect(PUMPBUTTON_X, PUMPBUTTON_Y, PUMPBUTTON_W, PUMPBUTTON_H, ILI9341_RED); //Fills the buttons rectangle
  tft.setCursor(PUMPBUTTON_X+5, PUMPBUTTON_Y+17); //Sets text cursor
  tft.setTextColor(ILI9341_BLACK); //Set text color
  tft.setTextSize(2); //Set text size
  tft.println("Red light"); //Prints text
}
void blueBtn()
{
  tft.fillRect(WSENSBUTTON_X, WSENSBUTTON_Y, WSENSBUTTON_W, WSENSBUTTON_H, ILI9341_BLUE);
  tft.setCursor(WSENSBUTTON_X+5, WSENSBUTTON_Y+17);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("Blue light");
```

```
}
void whiteBtn()
{
  tft.drawRect(LIGHTBUTTON_X, LIGHTBUTTON_Y, LIGHTBUTTON_W,
    LIGHTBUTTON_H, ILI9341_BLACK);
  tft.setCursor(LIGHTBUTTON_X+5, LIGHTBUTTON_Y+17);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("White light");
}
void turnOffBtn()
{
  tft.fillRect(ASENSBUTTON_X, ASENSBUTTON_Y, ASENSBUTTON_W,
    ASENSBUTTON_H, ILI9341_LIGHTGREY);
  tft.setCursor(ASENSBUTTON_X+5, ASENSBUTTON_Y+17);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("Turn off");
}
void increaseBtn()
{
  tft.fillRect(INCREASEBUTTON_X, INCREASEBUTTON_Y, INCREASEBUTTON_W,
    INCREASEBUTTON_H, ILI9341_GREEN);
  tft.setCursor(INCREASEBUTTON_X+15, INCREASEBUTTON_Y+12);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(4);
  tft.println("+");
}
void decreaseBtn()
{
  tft.fillRect(DECREASEBUTTON_X, DECREASEBUTTON_Y, DECREASEBUTTON_W,
    DECREASEBUTTON_H, ILI9341_RED);
  tft.setCursor(DECREASEBUTTON_X+15, DECREASEBUTTON_Y+12);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(4);
  tft.println("-");
}
void redBtnHit()
{
  Serial.println("Lights switched color to red");
  chase(strip.Color(255, 0, 0)); //Sets light color to red
  lightColor=2; //Sets light color, as the chosen light is supposed to
    be turned on when scheduled
}
void blueBtnHit()
{
  Serial.println("Lights switched color to blue");
  chase(strip.Color(0, 0, 255)); //Sets light color to blue
  lightColor=1; //Sets light color, as the chosen light is supposed to
    be turned on when scheduled
}
void whiteBtnHit()
{
  Serial.println("Lights switched color to white");
  chase(strip.Color(100, 100, 100)); //Sets light color to white
  lightColor=3; //Sets light color, as the chosen light is supposed to be turned on when scheduled
}
void turnOffBtnHit()
{
  Serial.println("Lights switched off");
  chase(strip.Color(0, 0, 0)); //Turn off lights
}
void decreaseBtnHit()
{
  Serial.println("Decrease button hit");
  if(hoursLightPerDay>1) //Hours of lights per day must be at least 1
  {
    hoursLightPerDay--; //If decrease button is pressed, decrease hoursOfLightPerDay by one
    timeText(hoursLightPerDay); //Sets the text with the new number of hours
  }
}
void increaseBtnHit()
{
  Serial.println("Increase button hit");
  if(hoursLightPerDay<24) //Hours of lights per day must be at most 24
  {
    hoursLightPerDay++; //Increase the number of hours of light per day when pressed
    timeText(hoursLightPerDay); //Sets the text with the new number of hours
  }
}
void turnOnScheduledLights() //Function for turning on the lights with the last decided color. Default is white
{
  if(nPumpOn==false && wPumpOn==false) //Lights cannot be lit at the same time as pumps are on
  {
    switch(lightColor)//0 = lights off, 1=blue, 2=red, 3=white
    {
      case 1:
        chase(strip.Color(0, 0, 255)); //Blue
      break;
      case 2:
        chase(strip.Color(255, 0, 0)); //Red
      break;
      case 3:
        chase(strip.Color(100, 100, 100)); //White
      break;
    }
  }
  else
  {
    unableToTurnOnLights = true; //Turns on if lights were unable to be turned on due to pumps being on
  }
}
void turnOffScheduledLights() //Turns off lights
{
  chase(strip.Color(0, 0, 0)); //Turn off
}
void timeText(int) //Function for setting the text of the hour setting
    text
{
  tft.setCursor(INCREASEBUTTON_X, INCREASEBUTTON_Y+65);
  tft.setTextSize(1);
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE);
  tft.print("Hours light per day");
  tft.setTextSize(3);
  tft.setCursor(INCREASEBUTTON_X, INCREASEBUTTON_Y+80);
  tft.print("      ");
  tft.setCursor(INCREASEBUTTON_X, INCREASEBUTTON_Y+80);
  tft.print(hoursLightPerDay);
  tft.print(" h");
}
static void chase(uint32_t c) //Function for changing light color using
     RGB (0-255,0-255,0-255)
  {
      strip.fill(c, 0, strip.numPixels());
      strip.show();
  }
```

```
//////////////////////////////////////
//File 4 - Main menu button functions//
//////////////////////////////////////
//Sets all the buttons for the main menu
void wSensBtn()
{
  tft.fillRect(WSENSBUTTON_X, WSENSBUTTON_Y, WSENSBUTTON_W,
    WSENSBUTTON_H, ILI9341_BLUE); //Fills the buttons rectangle
  tft.setCursor(WSENSBUTTON_X+5, WSENSBUTTON_Y+8); //Sets text cursor
  tft.setTextColor(ILI9341_BLACK); //Set text color
  tft.setTextSize(2); //Set text size
  tft.println("Water"); //Prints text
  tft.setCursor(WSENSBUTTON_X+5, WSENSBUTTON_Y+25); //Set cursor
  tft.print("sensors"); //Prints text
}
void wSensBtnHit()
{
  menuMode = 1; //Sets menu page
  clearScreen(); //Clears the screen before page change
  backBtn(); //Prints back button
  changeMenuText("Water sensors");
}
void aSensBtn()
{
  tft.fillRect(ASENSBUTTON_X, ASENSBUTTON_Y, ASENSBUTTON_W,
    ASENSBUTTON_H, ILI9341_YELLOW);
  tft.setCursor(ASENSBUTTON_X+5, ASENSBUTTON_Y+8);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println(”Air”);
  tft.setCursor(ASENSBUTTON_X+5, ASENSBUTTON_Y+25);
  tft.print(”sensors”);
}
void aSensBtnHit()
{
  menuMode = 4; //Sets menu page
  clearScreen(); //Clears the screen before page change
  backBtn(); A//Prints back button
  changeMenuText(”Air sensors”);
}
void pumpBtn()
{
  tft.fillRect(PUMPBUTTON_X, PUMPBUTTON_Y, PUMPBUTTON_W, PUMPBUTTON_H,
    ILI9341_CYAN);
  tft.setCursor(PUMPBUTTON_X+5, PUMPBUTTON_Y+17);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println(”Pumps”);
}
void pumpBtnHit()
{
  clearScreen(); //Clears the screen before page change
  pumpTestBtn(); //Prints the pump test button
  backBtn(); //Prints back button
  changeMenuText(”Pumps”);
  menuMode = 2; //Sets menu page
}
void lightBtn()
{
  tft.fillRect(LIGHTBUTTON_X, LIGHTBUTTON_Y, LIGHTBUTTON_W,
    LIGHTBUTTON_H, ILI9341_MAGENTA);
  tft.setCursor(LIGHTBUTTON_X+5, LIGHTBUTTON_Y+17);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println(”Lights”);
}
void lightBtnHit()
{
  clearScreen(); //Clears the screen before page change
  redBtn(); //Prints red button
  backBtn(); //Prints backbutton
  whiteBtn(); //Prints white button
  blueBtn(); //Prints blue button
  turnOffBtn(); //Prints turn off-button
  increaseBtn(); //Prints increase-button
  decreaseBtn(); //Prints decrease-button
  timeText(hoursLightPerDay); //Prints the time text
  changeMenuText(”Lights”); //Sets menu text
  delay(200); //Delay 200ms in order to prevent double-clicks
  menuMode = 3; //Sets menu mode
}
```

```
////////////////////////////////
//File 5 - Nutrition functions//
////////////////////////////////
//The code below measures the TDS. It is taken from the datasheet of the SEN0244.
//Source: https://www.digikey.se/product-detail/sv/dfrobot/SEN0244/1738-1368-ND/8019062
int getMedianNum(int bArray[], int iFilterLen)
{
  int bTab[iFilterLen];
  for (byte i = 0; i<iFilterLen; i++)
  bTab[i] = bArray[i];
  int i, j, bTemp;
  for (j = 0; j < iFilterLen - 1; j++)
  {
  for (i = 0; i < iFilterLen - j - 1; i++)
    {
    if (bTab[i] > bTab[i + 1])
      {
        bTemp = bTab[i];
        bTab[i] = bTab[i + 1];
        bTab[i + 1] = bTemp;
      }
    }
  }
  if ((iFilterLen & 1) > 0)
  {
    bTemp = bTab[(iFilterLen - 1) / 2];
  }
  else
  {
    bTemp = (bTab[iFilterLen / 2] + bTab[iFilterLen / 2 - 1]) / 2;
  }
  return bTemp;
}
void TDSMeasurements()
{
  static unsigned long analogSampleTimepoint = millis();
  if(millis()-analogSampleTimepoint > 40U) //every 40 milliseconds, read the analog value from the ADC
  {
    analogSampleTimepoint = millis();
    analogBuffer[analogBufferIndex] = analogRead(TdsSensorPin); //read the analog value and store into the buffer
    analogBufferIndex++;
    if(analogBufferIndex == SCOUNT)
    {
      analogBufferIndex = 0;
    }
  }
  static unsigned long printTimepoint = millis();
  if(millis()-printTimepoint > 800U)
  {
    printTimepoint = millis();
    for(copyIndex=0;copyIndex<SCOUNT;copyIndex++)
    analogBufferTemp[copyIndex]= analogBuffer[copyIndex];
    averageVoltage = getMedianNum(analogBufferTemp,SCOUNT) * (float)
   VREF / 1024.0; //read the analog value more stable by the median
   filtering algorithm, and convert to voltage value
    float compensationCoefficient=1.0+0.02*(temperature-25.0); //
   temperature compensation formula: fFinalResult(25^C) = fFinalResult
   (current)/(1.0+0.02*(fTP-25.0));
    float compensationVoltage=averageVoltage/compensationCoefficient;
   //temperature compensation
    tdsValue=(133.42*compensationVoltage*compensationVoltage*
   compensationVoltage - 255.86*compensationVoltage*
   compensationVoltage + 857.39*compensationVoltage)*0.5; //convert
   voltage value to tds value
    Serial.print("TDS Value:");
    Serial.print(tdsValue,0);
    Serial.println("ppm");
  }//End of code taken from the SEN0244 datasheet
}
```

```
/////////////////////////////////
//////File 6 - Pump page/////////
/////////////////////////////////
//menuMode = 2
void pumpTestBtn()
{
  tft.fillRect(PUMPTEST_X, PUMPTEST_Y, PUMPTEST_W, PUMPTEST_H,
   ILI9341_RED);
  tft.setCursor(PUMPTEST_X+5, PUMPTEST_Y+30);
  tft.setTextColor(ILI9341_BLACK);
  tft.setTextSize(2);
  tft.println("Water pump test");
}
void resetPumpTestStatus()
{
  tft.setCursor(PUMPTEST_X+5, PUMPTEST_Y+100);
  tft.println("                         ");
}
void pumpTestBtnHit()
{
  tft.setCursor(PUMPTEST_X+5, PUMPTEST_Y+100); //Set cursor
  tft.setTextSize(2); //Set text size
  tft.println("Pump test starting..."); //Prints text
  startWPump(); //Starts water pump
  delay(10000); //Delay 10s (ie run pumps 10s)
  stopWPump(); //Stops water pump
  feedNutrients(2000); //Run nutrient pump 2s
  resetPumpTestStatus(); //Resets the text of pump status
  tft.setCursor(PUMPTEST_X+5, PUMPTEST_Y+100); //Set cursor
  tft.println("Pumps tested!"); //Print text
  delay(2000); //Delay 2s
  resetPumpTestStatus(); //Resets the text of pump status
}
void startWPump()
```

```
{
  turnOffScheduledLights(); //Turns off lights when pump starts in order to provide enough current for the pumps
  Serial.println("PUMPPIN HIGH");
  digitalWrite(WPUMPPIN, HIGH); //Setting the pump pin high starts the pump
}
void stopWPump()
{
  digitalWrite(WPUMPPIN, LOW); //Setting the pump pin to low stops the pump
  Serial.println("PUMPPIN LOW");
  turnOnScheduledLights(); //Turn on lights again after stopping the pump
}
void feedNutrients(int nutrientPumpTime)
{
  turnOffScheduledLights();
  nPumpOn = true; //Set pump status to off
  digitalWrite(NPUMPPIN, HIGH); //Setting the pump pin high starts the pump
  delay(nutrientPumpTime); //Delay nutrientPumpTime ms
  digitalWrite(NPUMPPIN, LOW); //Setting the pump pin to low stops the pump
  nPumpOn = false; //Set pump status to off
  turnOnScheduledLights();
}
```

```
///////////////////////////////////////
///File 7 - General system functions///
///////////////////////////////////////
void clearScreen() //The clear screen func simply draws a white rectangle over all of the screen
{
  tft.fillRect(0, 0, 340, 240, ILI9341_WHITE);
}
void changeMenuText(const String& menuText) //This function is made to facilitate changing the menu text on top of each page
{
  tft.setCursor(FRAME_X+5, FRAME_Y+5);
  tft.setTextSize(3);
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE);
  tft.print(menuText);
}
```

```
/////////////////////////////////
/File 8 - Back button functions//
/////////////////////////////////
void backBtn()
{
  tft.fillRect(BACKBUTTON_X, BACKBUTTON_Y, BACKBUTTON_W, BACKBUTTON_H, ILI9341_BLACK);
  tft.setCursor(BACKBUTTON_X+5, BACKBUTTON_Y+17);
  tft.setTextColor(ILI9341_WHITE);
  tft.setTextSize(2);
  tft.println("BACK");
}
void backBtnHit() //Goes back to main menu, prints all buttons of the main menu
{
  menuMode = 0;
  clearScreen();
  wSensBtn();
  pumpBtn();
  lightBtn();
  aSensBtn();
  changeMenuText("Menu");
}
```

```
///////////////////////////////////////
//////File 9 - Touch Input Handler/////
///////////////////////////////////////
//This function handles all the touch data, and executes button hit functions
void touchInputHandler()
{
  // See if there's any  touch data for us
  if (!ts.bufferEmpty())
  {
    // Retrieve a point
    TS_Point p = ts.getPoint();
    // Scale using the calibration #'s
    // and rotate coordinate system
    p.x = map(p.x, TS_MINY, TS_MAXY, 0, tft.height()); //gets coordinate data
    p.y = map(p.y, TS_MINX, TS_MAXX, 0, tft.width());
    int y = tft.height() - p.x; //calculates y and x touch data
    int x = p.y;
    //The code below looks to see if the x and y coordinates fall within each button.
    //If it does, the code checks which menu is chosen and if the correct one is chosen, the corresponding "button hit" function is executed.
    if((x > BACKBUTTON_X) && (x < (BACKBUTTON_X + BACKBUTTON_W)))
    {
      if ((y > BACKBUTTON_Y) && (y <= (BACKBUTTON_Y + BACKBUTTON_H)))
      {
        Serial.println("Backbtn hit");
        x=0; y=0;
        if(menuMode!=0)
        {
          backBtnHit();
        }
      }
    }
    if((x > PUMPBUTTON_X) && (x < (PUMPBUTTON_X + PUMPBUTTON_W)))
    {
      if ((y > PUMPBUTTON_Y) && (y <= (PUMPBUTTON_Y + PUMPBUTTON_H)))
      {
        Serial.println("PUMPBUTTON HIT");
        x=0; y=0;
        switch(menuMode)
        {
          case 0:
            pumpBtnHit();
          break;
          case 3:
            redBtnHit();
          break;
        }
      }
    }

    if((x > WSENSBUTTON_X) && (x < (WSENSBUTTON_X + WSENSBUTTON_W)))
    {
      if ((y > WSENSBUTTON_Y) && (y <= (WSENSBUTTON_Y + WSENSBUTTON_H))
    )
      {
        Serial.println("WSENSBUTTON HIT");
        x=0; y=0;
        switch(menuMode)
        {
          case 0:
            wSensBtnHit();
          break;
          case 3:
            blueBtnHit();
          break;
        }
      }
    }
    if((x > ASENSBUTTON_X) && (x < (ASENSBUTTON_X + ASENSBUTTON_W)))
    {
      if ((y > ASENSBUTTON_Y) && (y <= (ASENSBUTTON_Y + ASENSBUTTON_H))
    )
      {
        Serial.println("ASENSBUTTON HIT");
        x=0; y=0;
        switch(menuMode)
        {
          case 0:
            aSensBtnHit();
          break;
          case 3:
            turnOffBtnHit();
          break;
        }
      }
    }
    if((x > LIGHTBUTTON_X) && (x < (LIGHTBUTTON_X + LIGHTBUTTON_W)))
    {
      if ((y > LIGHTBUTTON_Y) && (y <= (LIGHTBUTTON_Y + LIGHTBUTTON_H))
    )
       {
          Serial.println("LIGHTBUTTON HIT");
          x=0; y=0;
          switch(menuMode)
          {
             case 0:
                lightBtnHit();
             break;
             case 3:
                whiteBtnHit();
             break;
          }
       }
    }
    if(((x > PUMPTEST_X) && (x < (PUMPTEST_X + PUMPTEST_W)))&& menuMode
    ==2)
       {
       if ((y > PUMPTEST_Y) && (y <= (PUMPTEST_Y + PUMPTEST_H)))
          {
             Serial.println("PUMPTEST HIT");
             x=0; y=0;
             pumpTestBtnHit();
          }
       }
    if(((x > DECREASEBUTTON_X) && (x < (DECREASEBUTTON_X +
    DECREASEBUTTON_W)))&& menuMode==3)
       {
       if ((y > DECREASEBUTTON_Y) && (y <= (DECREASEBUTTON_Y +
    DECREASEBUTTON_H)))
          {
             Serial.println("DECREASEBUTTON HIT");
             x=0; y=0;
             decreaseBtnHit();
             delay(150);
          }
       }
    if(((x > INCREASEBUTTON_X) && (x < (INCREASEBUTTON_X +
    INCREASEBUTTON_W)))&& menuMode==3)
       {
       if ((y > INCREASEBUTTON_Y) && (y <= (INCREASEBUTTON_Y +
    INCREASEBUTTON_H)))
          {
             Serial.println("INCREASEBUTTON HIT");
             x=0; y=0;
             increaseBtnHit();
             delay(150);
          }
       }
  }
}
```

```
/////////////////////////////////
///File 10 - Water sensor page////
/////////////////////////////////
//menuMode = 1
void updateVolume(float wVolume)
{
  tft.setCursor(TEXT_X+5, TEXT_Y); //Set cursor location
  tft.setTextSize(2); //Set text size
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE); //Set text color and background
  tft.print("Current remaining"); //Print text
  tft.setCursor(TEXT_X+5, TEXT_Y+20); //Set cursor location
  tft.print("water volume is:"); //Print text
  tft.setTextColor(ILI9341_BLUE,ILI9341_WHITE); //Set text color
  tft.print(wVolume); //Print water volume
  tft.print(" dL"); //print text
}
void updateNT(int NT)
{
  tft.setCursor(TEXT_X+5, TEXT_Y*2);
  tft.setTextSize(2);
  tft.setTextColor(ILI9341_BLACK,ILI9341_WHITE);
  tft.print("Current nutrient");
  tft.setCursor(TEXT_X+5, TEXT_Y*2+20);
  tft.print("concentration is:");
  tft.setTextColor(ILI9341_BLUE,ILI9341_WHITE);
  tft.print(NT);
  tft.print(" ppm");
}
int getWVolume()
{
  digitalWrite(TRIGPIN, LOW);
  delayMicroseconds(2);
  // Sets the trigPin HIGH (ACTIVE) for 10 microseconds
  digitalWrite(TRIGPIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIGPIN, LOW);
  // Reads the echoPin, returns the sound wave travel time in microseconds
  duration = pulseIn(ECHOPIN, HIGH);
  // Calculating the distance
  distance = duration * 0.034 / 2; // Speed of sound wave divided by 2 (go and back)
  waterVolume = (10-distance)*95*0.01; //95 cm2 is the area of the water tank, multiplied by the water level in cm. 10cm is the height of the water tank, so to get the height the distance from the ultrasonic sensor is subtracted
  //The volume is displayed in dL, and 1 cm3 = 0.01 dL (hence the multiplication by 0.01).
  return waterVolume;
}
```

A.2 Simulink

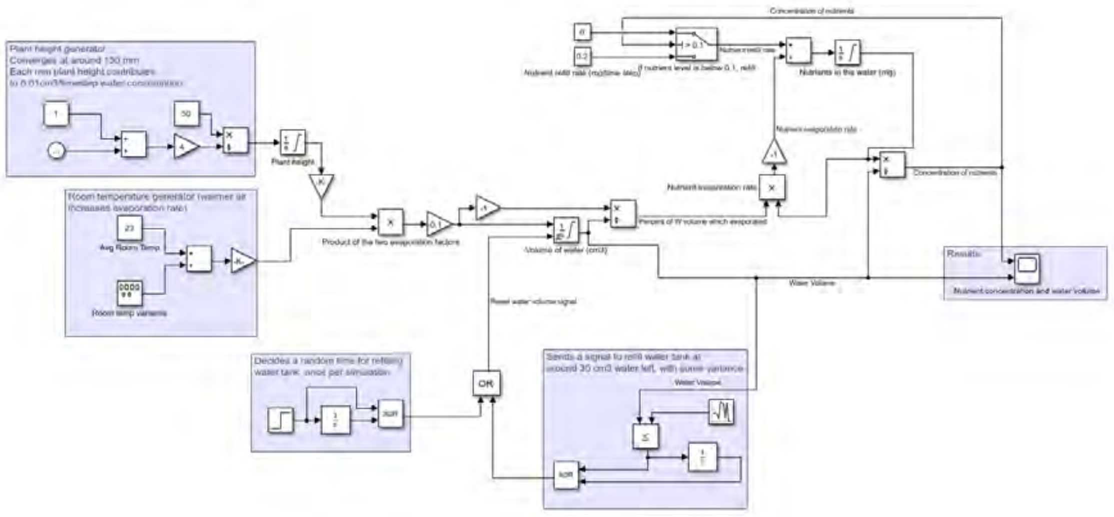

Plant height generator
Converges at around 130 mm
Each mm plant height contributes
to 0.01cm3/timestep water consumption
1
50
4
Plant height
Room temperature generator (warmer air increases evaporation rate)
23
Avg Room Temp
Room temp variance
-K-
Product of the two evaporation factors
0.1
-1
Volume of water (cm3)
Percent of W volume which evaporated
0
0.2
> 0.1
Nutrient refill rate (mg/time step)
If nutrient level is below 0.1, refill
Nutrient refill rate
Nutrients in the water (mg)
Concentration of nutrients
Nutrient evaporation rate
Nutrient evaporation rate
-1
Concentration of nutrients
Results
Nutrient concentration and water volume
Water Volume
Reset water volume signal
OR
Decides a random time for refilling water tank, once per simulation
XOR
Sends a signal to refill water tank at around 30 cm3 water left, with some variance
Water Volume
≤
XOR

Figure A.2.1: Picture of the simulated system, made in Simulink [30].

Figure A.2.1 is split up into two below, for clearer view of the simulated system.

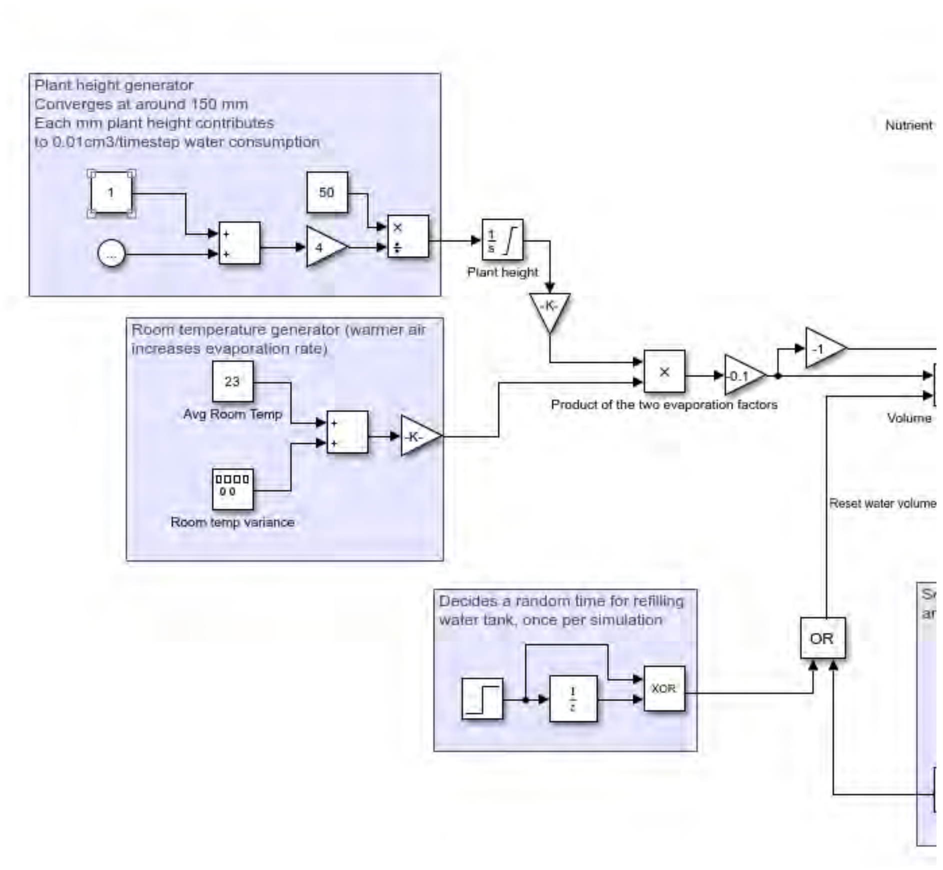

Plant height generator
Converges at around 150 mm
Each mm plant height contributes
to 0.01cm3/timestep water consumption
1
...
50
4
Plant height
-K-
Room temperature generator (warmer air increases evaporation rate)
23
Avg Room Temp
Room temp variance
-K-
Product of the two evaporation factors
-0.1
-1
Nutrient
Volume
Reset water volume
Decides a random time for refilling water tank, once per simulation
XOR
OR

Figure A.2.2: Divided picture of the simulated system, first half.

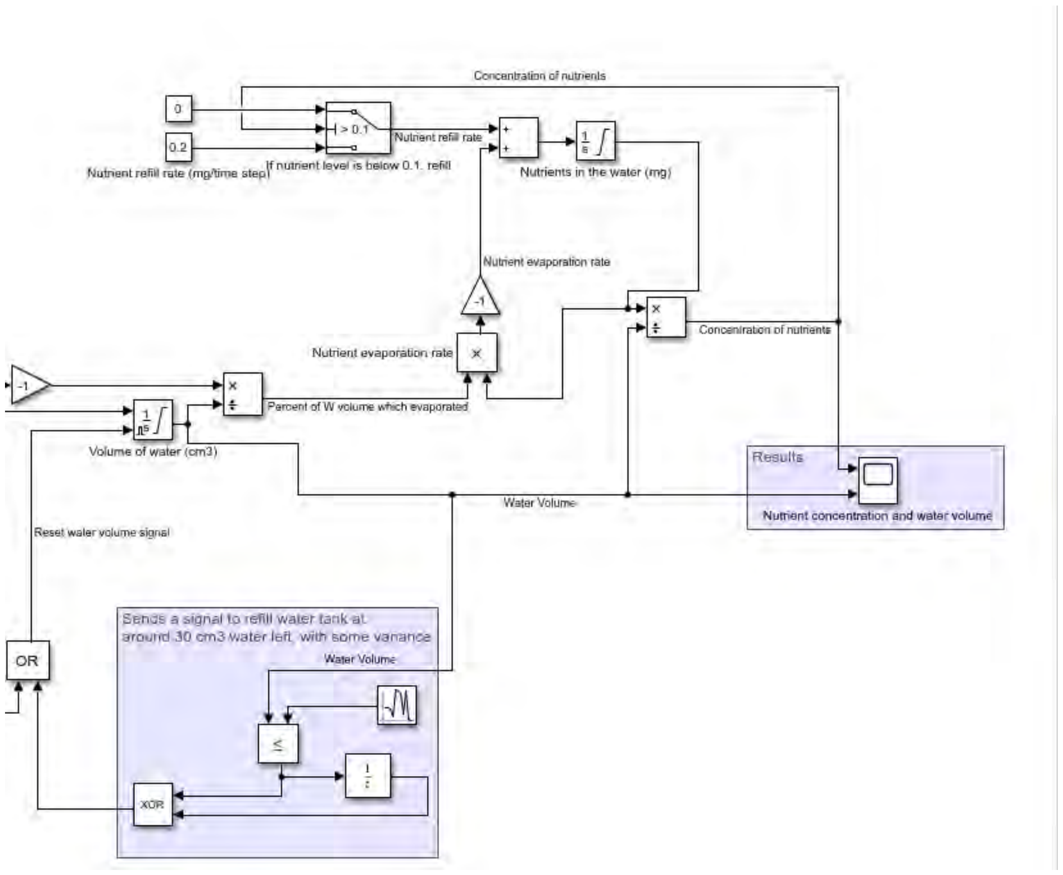

Concentration of nutrients
0
0.2
> 0.1
Nutrient refill rate
Nutrient refill rate (mg/time step)
If nutrient level is below 0.1, refill
Nutrients in the water (mg)
Nutrient evaporation rate
-1
Nutrient evaporation rate
Concentration of nutrients
Percent of W volume which evaporated
Volume of water (cm3)
Water Volume
Results
Nutrient concentration and water volume
Reset water volume signal
OR
Sends a signal to refill water tank at around 30 cm3 water left, with some variance
Water Volume
XOR

Figure A.2.3: Divided picture of the simulated system, second half.

A.3 Finished incubator

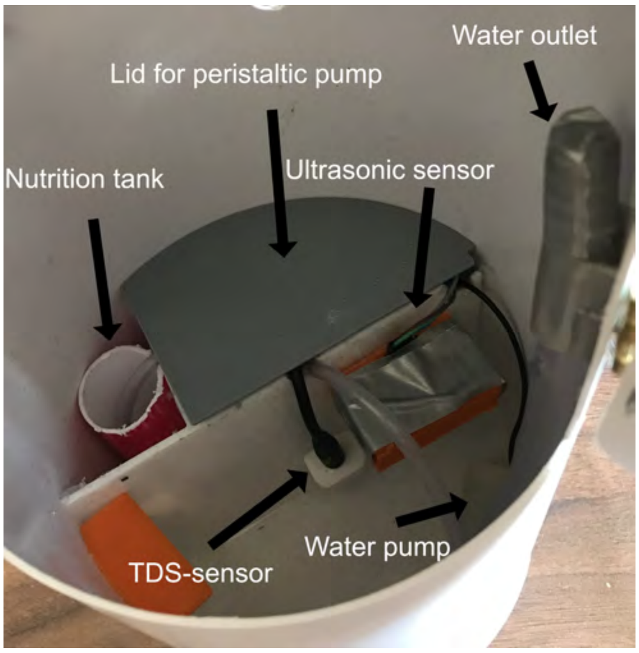
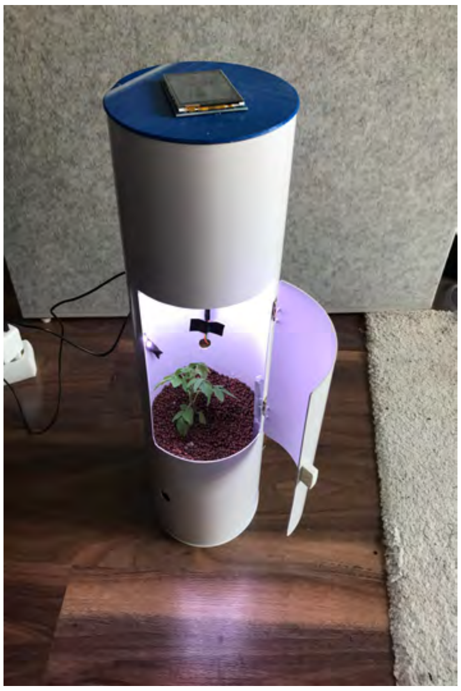
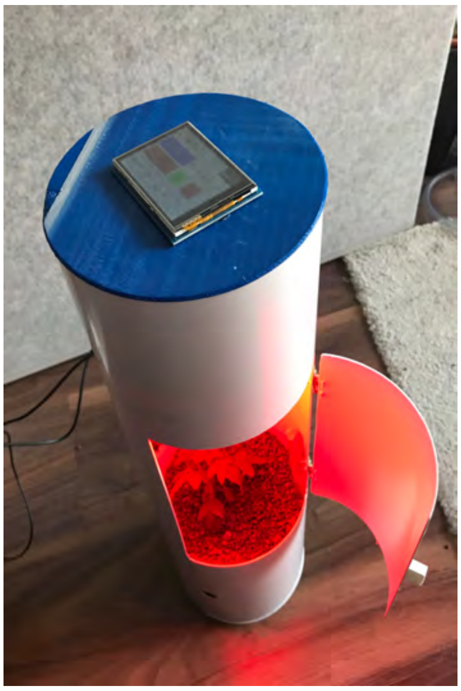


Figure A.3.1: Water tank area without the covering drop disc, photo taken by Jonathan Ling, edited in Affinity Designer [4].

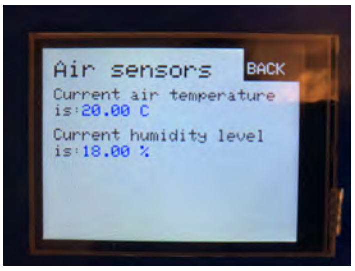


(a) Finished project with white lights.



(b) Finished project with red lights on.

Figure A.3.2: Pictures of different light settings, photos taken by Jonathan Ling.

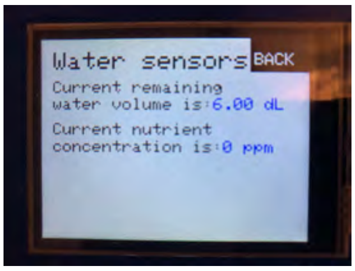


Air sensors BACK

Current air temperature is:20.00 C

Current humidity level is:18.00 %

(a) Air sensors menu.

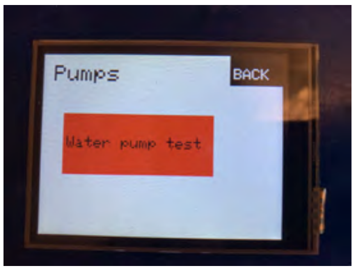


Water sensors BACK

Current remaining water volume is:6.00 dL

Current nutrient concentration is:0 ppm

(b) Water sensors menu.



Pumps BACK

Water pump test

(c) Pumps menu.

Figure A.3.3: Images of different sub-menus in the GUI, photos taken by Gustav Lindstrand.

A.4 TDS experiment

TDS experiment with a container of water.

500 ml of water. Measurements made at the bottom of the container

Measurement 1: No nutrient added, 21 C
Result: 131 ppm, No variation

Measurement 2: 1 ml nutrient added, not stirred, 21 C
Result: Quickly increases to about 800 ppm. Not a very stable reading

Measurement 3: 1 ml nutrient added, stirred abt 1 s, 21 C
Result: Stabilizes quickly at 272 ppm

Measurement 4: 1 ml nutrient added, stirred abt 4 s, the probed was moved to the other side of the bottom, 21 C
Result: Changes just a little, to 276 ppm

Measurement 5: Another 1 ml nutrient added (2 ml in total), no stirring. 22 C
Time: 21:24:40
Result: After a few seconds PPM increases to abt 780 ppm. Unstable, varies between 740-800

Measurement 6: 2 ml nutrient added in total, stirring for abt 1 s. 22 C
Time: 21:26:40
Result: After a few seconds the value is stabilized at 380 ppm +- 4 ppm

Measurement 7: Another 0.5 ml nutrient added (2.5 ml in total, this is how much that should be added according to nutrient instructions), no stirring. 22 C
Time: 21:28:40
Result: After a few seconds it reaches abt 790ppm. Unstable, varies between 740-800 ppm

Measurement 8: 2.5 ml nutrient added in total, stirring for abt 1 s. 22 C
Time: 21:30:20
Result: After a few seconds it reaches 444 ppm +- 4 ppm

Measurement 9: 2.5 ml nutrient added in total. Stirred water. Measurement at the bottom of the container.
Time: 21:33:30
Result: 509 ppm

Measurement 10: 2.5 ml nutrient added in total. Stirred water. Measurement at the top of the container
Time: 21:34:30
Result: 520 ppm

Measurement 11: 2.5 ml nutrient added in total. Stirred water. Measurement at the top of the container, on opposite side of Measurement 10
Time: 21:34:30
Result: 450 ppm

Measurement 12: 2.5 ml nutrient added in total. Heavily stirred water. Measurement at the bottom of the container
Time: 21:39:40
Result: 436 ppm

Measurement 13: 2.5 ml nutrient added in total. Heavily stirred water. Measurement in the middle of the container
Time: 21:40:48
Result: 458 ppm

Measurement 14: 2.5 ml nutrient added in total. Heavily stirred water. Measurement at the top of the container
Time: 21:41:04
Result: 517 ppm

Measurement 15: 2.5 ml nutrient added in total. Heavily stirred water. Measurement at the bottom of the container
Time: 21:42:23
Result: 481ppm

Conclusion: The desired TDS value with a water temperature of 22 C seems to vary between 430-520 ppm. Stirring has a large effect on the readings, stabilizing the values.

Day 2:

Water temperature 20 C. Same water as in Measurement 15, but has been left untouched during the night without stirring or shaking.

Measurement 16: At the bottom of the container, stable at på 409 ppm.
Measurement 17: In the middle of the container, stable at 422 ppm.
Measurement 18: Top of the container, stable at 481 ppm.

TRITA -ITM-EX 2021:48