University of South-Eastern Norway

IT and information systems

Bachelor thesis

Multifunctional Advanced Yield Accumulator, an automated hydroponic system with data collection

Author:
Marius Norheim

Spring, 2021

Preface

This is a thesis that summarizes my academic endeavours having gone through the bachelor program IT and information systems at University of South-East Norway. In my ventures through the education program I have gained valuable insights into development methodology, computer security, data visualization and automation processes, while expanding my knowledge of working hands-on with code.

The culmination being this project, an automated hydroponic system with data collection where I get to utilize a wide range of the skill set I have learned. A further motivation for making this project is a strong personal belief in green technology to enable sustainability for everyone regardless of resource availability, and to have a positive impact on the world, however small it may be.

The cost of this project ended up at just over €200, with the biggest single cost being a Raspberry Pi and it is easily expanded by simply getting a bigger plastic container for the water reservoir. It is a true privilege living in an age where we can build automated systems that can grow food and take care of the plants at such low budgets if we just apply some creativity and effort into developing a working solution.

Dedication

The thesis is notably dedicated to my mom who sadly was taken far too early by cancer, my dad and Swami Rajinder Ji. For being my spiritual guides, and in honor of raising me to be a free rebellious spirit, to always do my best and to work hard while persevering through adversity and staying positive. I am eternally grateful for the privilege of including all of you in my life and being a part of yours.

Acknowledgements

I want to give a special thanks to Katerina Mangaroska at University of South-Eastern Norway for superb guidance during the project, and being the best educator I have ever had the pleasure of learning from. Always a motivating force for me to do better, supporting and believing in me and inspiring my critical thinking to the next level.

Contents

List of Figures

List of Tables

1 Project charter

The idea behind the project charter is to introduce the topic and also identify the benefits, constraints, milestones and stakeholders to base the work upon.

1.1 Introduction

The goal for this project is to make a working prototype of an automated hydroponic system, by way of an application that collects data from sensors and performs regulating actions on the climate and water quality based on the data. The application will also function as a database logger and visualize data to an end user, along with a web API that the user can query for real time and historical data from the database.

To meet the requirements and do appropriate testing a grow box and hydroponics system will be designed that can be used indoors to grow leafy greens, tomatoes, peppers and cucumbers and other model plants. I have decided to use one of the simplest hydroponic systems to set up - Deep Water Culture. Because the project is a proof of concept, the aim is to minimize potential design complications and also reduce the amount of risk factors.

The application responsible for automating and data logging is not discriminatory towards any type of hydroponic system, and the underlying principles for optimal conditions remain the same.

This thesis is part of the project and functions as documentation for the organizing, how progress has been made in the project and to substantiate my process of justifications and critical thinking applied to the work involved (Fazaeli et al., 2012).

1.2 Benefits

The benefits of realizing this project are mostly related to time or effort spent micromanaging the environment a plant needs in order to thrive. Automating climate and water control based on sensor data saves the end user from spending time making sure the temperature and humidity is correct, and that the water quality and nutrient levels are at an optimal level.

Detailed monitoring presented to the user in a visually appealing manner is also beneficial for looking at changes to the environment over time. The data can also be used if a user has a noticeable positive development for plant growth or yield between different growth cycles. The web API can be queried for real time monitoring, and further developed for alerts or loading historical data into custom visualization or analysis techniques.

1.3 Constraints

The project has several constraints that need to be factored into the planning and execution of the project, some with a higher priority than others.

1.3.1 Time

There is a fixed, non-negotiable delivery date set at 21.05.2021, 14:00. Thus, time is the absolute highest priority in this project.

1.3.2 Scope

As long as the minimum viable product criteria are met, requirements ranking less than high in the priority are considered less valuable than delivering on time for this project.

1.3.3 Budget

The budget for this project is capped at €250. While there is an ultimate goal of improving cost effect to make this project more accessible, the delivery date and scope of the project takes precedence over budget issues up to this cap until a later development iteration.

1.3.4 Space

As a student living on a small budget, the physical space occupied by the system is a limitation. This is not deemed as a critical factor in the final product as hydroponic systems are very scalable, but for a successful implementation into production the electronic devices used in this project should be reassessed and properly dimensioned for the occupying space.

1.4 Milestones

Milestone	**Due date**	**Completion date**
Preliminary grow box	01.02.2021	01.02.2021
Hydroponic water reservoir	01.02.2021	01.02.2021
Component fitting in grow box	05.05.2021	03.05.2021
Sensor data polling	15.02.2021	10.02.2021
Sensor data saving	01.03.2021	01.03.2021
Air climate regulating actions	10.04.2021	09.04.2021
Water quality regulating actions	15.04.2021	15.04.2021
Visualizations	01.05.2021	01.05.2021
API	18.05.2021	17.05.2021
Electrical box	10.05.2021	10.05.2021
Testing	15.05.2021	14.05.2021

Table 1: Project milestones

1.5 Stakeholders

The product has a wide range of potential use depending on scaling, advanced functionality and success in implementing what end users want and need. Identifying stakeholders is an important first step in an attempt to recruit the stakeholders as part of the effort for the product to reach its potential and to envision the future of the product.

1.5.1 Primary

1. Recreational gardeners
2. Industrial farmers
3. Industrial workforce
4. Growers in areas with water shortage

Recreational gardeners have an interest in this product because it saves a significant amount of effort and time into producing homemade crops, a trend that is rising with the COVID pandemic ongoing.

The same reasons apply for industrial farmers, and implicitly their need for labour. Because the system reduces the efforts and time needed to ensure optimal conditions a lesser workforce is required for all combined tasks per growing cycle.

Growers in areas with water shortage are affected because they can gain increased effectiveness in produce per liter of water spent.

1.5.2 Secondary

1. Agriculture equipment producers
2. Semiconductor producers
3. Fertilizer producers
4. Crop exporters
5. Grocery stores

Businesses that produce equipment needed for hydroponic systems and agriculture could be positively impacted with increased sales if a strong trend is established, this includes agriculture equipment, semiconductors and fertilizer.

Crop exporters have the possibility of being both positively and negatively affected depending on which market the product succeeds in. If it succeeds in the private market and home growing has a strong increased trend, crop exports could be lowered. If it succeeds in the industrial market, crop exports could be rising.

Grocery stores could be negatively impacted with a loss in sales if the private market become more self sufficient.

1.5.3 Key stakeholders

1. Investment companies
2. Private equity firms

Key stakeholder that are not directly or indirectly affected by the product could be potential investment companies or private equity firms that invest into the product or parent company and assists in getting to market.

2 Prestudy

The goal of the prestudy is to establish general research on the topic and gain knowledge that is needed later in the project outside the scope of application and system development.

2.1 Hydroponics

Hydroponics is a technique for growing plants in a nutrient solution containing all the micro and macro elements that a plant needs for optimal growth. It's common to support the root system and base of the stem using an inert medium such as rockwool, perlite or gravel. Hydroponics is derived from the ancient Greek words 'hydro' that translates to water, and 'ponos' that translates to hardships, or labor.

The ever changing global climate can be a challenge to deal with for farmers, and the predictions for increased greenhouse gas emissions is that the climate will get more extreme. The implications are that weather deviations like heat waves, extreme cold and storms will become more intensified. Heat waves can significantly impact growing of crops because a higher than optimal temperature will restrain the plants from growing, and lakes and water sources can dry up, providing less irrigation for the plants.

On the inverse of the extremes, intense cold weather has similar effects on plants and in places where climate reaches freezing temperatures the crops will die. Heavy storms with an abundance of rainfall can also impact plant growth and create problems with root rot and fungus growth, because soil retains the water and it actually is possible to over water plants. It sounds contradictory to use hydroponics as a solution for this, but the key is in oxygenating the water reservoir sufficiently to prevent this. Greenhouses provide a controlled environment for the crops where they can thrive in optimal conditions year round uninhibited by the extreme weather changes, but greenhouses are traditionally more expensive to operate than a field.

The benefits of using hydroponics versus soil are abundant. For starters there is a significantly lower water usage and greater control over the growing environment, but using hydroponic systems also requires the grower to be more meticulous in the process because of the lack of a buffer element. There are risk factors we need to assess and it is absolutely crucial that water has a satisfactory level of oxygenation and the plants stay hydrated at all times, as it will die if left without water supply for a significantly shorter time than if grown in a buffer medium such as soil.

Growing 1 kilogram of tomatoes using industrial agriculture in soil requires 400 liters of water, while the same quantity using hydroponics requires 70 liters of water. This can be further reduced to an even lower level of 20 liters by using a specialized technique within hydroponics dubbed aeroponics.

2.2 Types of hydroponic systems

Hydroponics is a general term for all growing done directly in a nutrient solution, and there are an abundance of specific techniques and systems one can utilize within hydroponics. The different techniques each have their own benefits and inconveniences, some with special risk factors that need to be addressed for a successful implementation.

2.2.1 Deep Water Culture

Deep Water Culture is among the most simple hydroponic systems in use, only requiring a water reservoir where the plants root system is submerged in nutrient solution and an air pump/stone to keep the water oxygenated.

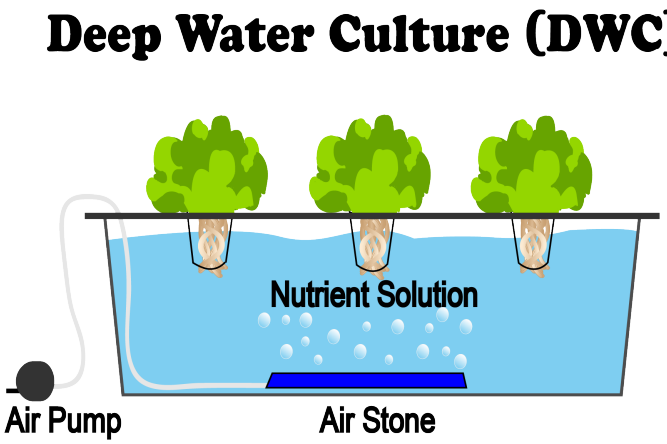

Figure 1: Concept of deep water culture hydroponic system

2.2.2 Nutrient Film Technique

Nutrient Film Technique (NFT) is a method where the nutrient solution is stored in a water reservoir with a pump connected to circulate a small amount of water into the highest pointing end of a separate container that holds the plants. The container that holds the plants is placed at

a slightly downward angle with a drain system connected back to the water reservoir, in order for gravity to assist the circulation of nutrient solution.

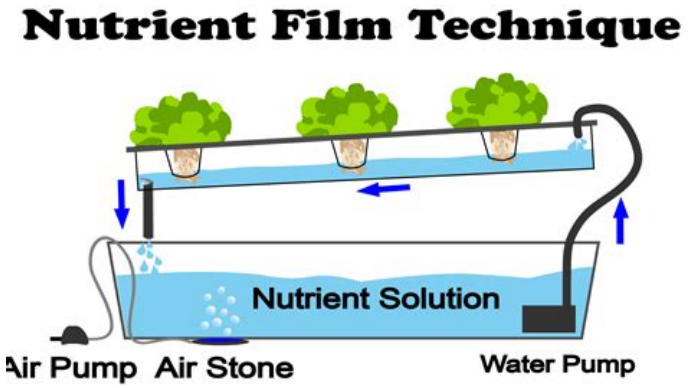

Figure 2: Concept of nutrient film technique hydroponic system

2.2.3 Ebb and flow

Ebb and flow is a method that simulates the natural tidal forces of nature. The ebb is the outgoing phase, when the tide drains away from the shore; and the flow is the incoming phase when water rises again. A water reservoir holds the nutrient solution while a pump controlled by a timer floods a second container that holds the plants on a regular interval. The second container has a drain to ensure the nutrient solution goes back into the water reservoir after the flooding.

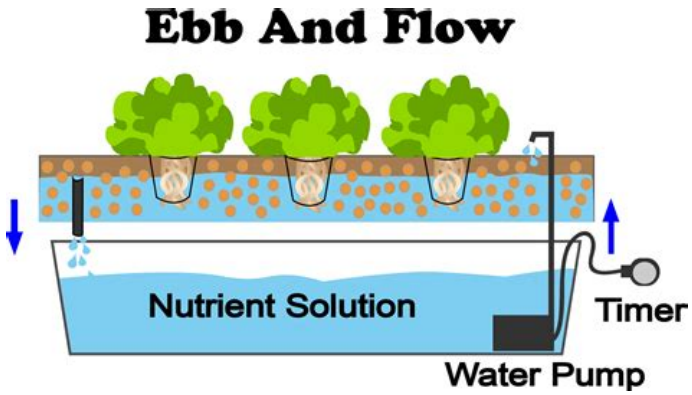

Figure 3: Concept of ebb and flow hydroponic system

2.2.4 Aeroponics

Aeroponics is a method where the root system of the plants are suspended in air and spray nozzles are utilized to feed the nutrient solution to the plants, usually by the way of a PVC pipe structure connected to a water pump submerged in the nutrient solution. Aeroponics has been used to grow plants in space and is recognised in terms of water usage as one of the most effective methods to grow plants.

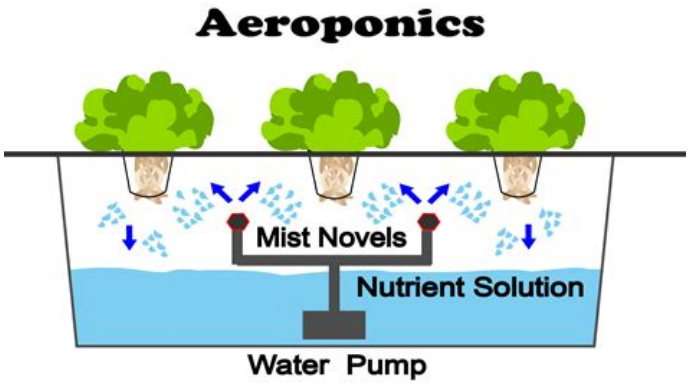

Figure 4: Concept of aeroponics system

2.3 Essentials for a thriving plant

Building on the research for different hydroponic systems available, there is also a need to find out what environmental factors affect growth in order to build a system that automates the growing process.

2.3.1 Light

There are several factors that determine if the light is appropriate for growing plants, and depending on what kind of plant is grown some will produce flowers (fruits) based on the duration of exposure. What is known as 'short day plants' (i.e, rice, soybean and onion) will only produce flowers when the day length is less than 12 hours, and inversely 'long day plants' (i.e, lettuce, radish and spinach) will only produce flowers when the exposure to sun is more than 12 hours. Some plants are neutral (i,e, tomato, cucumber and peas) and will produce flowers regardless of light exposure, but rather at a specific age.

The intensity or concentration of the light is also a critical factor, and is measured in a unit called lumen. Intensity of light increases the plants ability to produce food through photosynthesis.

Light has different wavelengths, and for a plant to thrive it needs light from both the blue and the red wavelength spectrum. Blue wavelength is primarily responsible for leaf growth, and mixed with red wavelength it encourages flowering of the plants.

2.3.2 Temperature

Temperature affects most of the processes in a growing plant, and up to a point increases the plants photosynthesis and respiration. Some plants are considered 'cool season crops' (i.e, lettuce, radish and spinach) while others are labeled 'warm season crops' (i.e tomato, cucumber and peppers).

Crop quality can also be affected by temperature. A lower temperature will reduce energy consumption by the plant and increase sugar storage, and a higher temperature can cause lettuce to taste bitter. However if the temperature falls too low or too rises too high, it will stagnate the growth of the plants.

2.3.3 Humidity

Relative humidity affects the transpiration of plants, but should be balanced as to not promote fungal growth. Also for this parameter different plants will have different needs depending on their natural climate.

2.3.4 Water quality

Like most other environmental conditions water temperatures that are too low or too high can adversely affect plant growth.

PH levels are important for nutrient uptake and most plants require a slightly acidic environment to thrive, between ph 5-6.5. Once again the exact levels differ between plants.

2.3.5 Nutrients

Optimal nutrient levels for different plants can vary widely. Some plants (i.e, tomato) requires a very high level of nutrients to thrive, and will have stagnated growth if the requirements are not met. Other plants such as lettuce and peppers are more moderated in their nutrient levels, and would be miserable if placed in a environment that is optimized for tomatoes.

2.4 Sensor equipment for data

The Johnny Five API has to be consulted to check for compatibility with the sensor equipment.

For reading temperature and humidity there are several choices available through the Hygrometer and Multi APIs such as HTU21D, SI7021, HIH6130 and SHT31D. Their margin for error are listed as fairly similar, so SHT31D has been selected for its low cost. Selection in waterproof temperature sensors are limited, with DS18B20 the only option.

To analyze the water quality in terms of ph and nutrient levels in terms of electrical conductivity there are a few options. DFRobot and Atlas Scientific both have viable options for the measurements, but for this project the Atlas Scientific sensors are excluded because of the combined price of $288 before shipping and import tax costs. The DFRobot EC sensor will also push the budget above the requirements at $70 plus shipping and import tax, so a more budget friendly EC sensor from Keystudio has been selected for the project. It has a limitation of 1000ppm, but for a prototype this has been deemed as an acceptable compromise. There is no API to read ph and electrical conductivity values, but its possible to read an analog value with the Sensor API and convert this into useful information in the application code.

Had budget not been a concern, some choices would have been different. The atlas scientific sensory equipment for ph and electrical conductivity seem a step above and beyond the rest, with a expected lifetime of 1,5years and the ability to stay submerged without re-calibration for a year - but so is the price.

2.5 Minimum viable product

Plants are living things, and similar to us humans there is no detailed 'one size fits all' miracle recipe to ensure ideal growth for all the different kinds of plants. They have different needs, and to reflect that our system should have a configuration module where it is easy to specify the needs for any given plant. This also means that different plants should be in separate growing environments to cater to their needs, e.g a tomato plant will have different optimal conditions than lettuce on most of the parameters that our system controls.

A minimum viable product in this project will demonstrate automatic regulatory actions on a hydroponic system and save data points to a database. This implies an enclosed environment where its possible to set desired parameters such as temperature and humidity, and collecting sensor data as regulatory actions can not be performed without knowing the state for the climate and water quality. At a minimum the system needs to display the aforementioned data in some capacity for testing purposes. The product also needs to be tested before delivery to ensure the system requirements are met.

Certain aspects of the water quality regulation in the major requirements can be sacrificed if the time schedule is at risk of being broken, as time constraints is the highest priority in the project. This is justified because the system inherently is a closed and thus highly controlled environment, and the risk of unexpected fluctuations of ph or electrical conductivity over a short time span is low without some external influence on the system.

3 Product

A detailed plan for the product is needed to ensure an organized development schedule with clear goals and objective that have to be met in the project.

3.1 User interviews

To determine the system requirements interviews with gardeners has been conducted. The interview objects were all gardening at a recreational level, but its reasonable to extrapolate these requirements to also somewhat account for professional gardeners, as the effects will be time saving and require less effort from the gardener to maintain the environment for optimal growth.

Respondents

- Per Skov
- Elisabeth Uggeland
- Raymond Thomassen

Questions asked

- Do you have any experience with hydroponics?
- How much time per week do you spend on gardening?
- What task is the most repetitive in your gardening activities?
- What task is the most time consuming in your gardening activities?
- What is the most mundane task in your gardening activities?
- What would you like to spend more time doing in your gardening activities?
- What sort of tasks would an ideal automated system do for your gardening activities?

Respondent - Per Skov

Yes, I switched to using a simple deep water culture system for my bigger plants such as tomatoes last year and I am currently considering investing in a system for smaller plants and herbs.

On average I spend over an hour in my garden every day from May to October, depending on season variations.

There's always something that needs more water or fertilizer. The working positions are increasingly straining on my back, and this is a big part in why I'm experimenting with hydroponics.

It's hard to pinpoint the activity that takes the most time in my garden. It's an overall effort to be honest, and you can't really skip out on doing the chores that are required. I just like to get going with activities, so fertilizing and doing detailed work like measuring and mixing nutrients sometimes feel like more effort than it really is.

The most mundane stuff to me is the micro managing of conditions. I do keep a hygrometer to check for temperatures and humidity but at this point I rarely act upon it unless it gets too cold either early or late in the season.

I would like to spend a bit more time on taking care of the individual plants with pruning and building supports that allow all the plants to get more surface area of sunlight.

An ideal automated system would take care of all the micro management for me and also allow for me to let the system go unmanaged for a week at the time.

Respondent - Elisabeth Jemtland

No, I keep my plants in a pot with soil.

Maybe a little over an hour per week.

Watering the plants is mostly what I do once the plants are seeded and potted over to bigger pots.

As I said, watering the plants is mostly what I do. If I see changes in the leafs of the plants I also give some fertilizer in the water to keep them happy, but I don't have the time or energy outside of work and kids to take a more scientific approach.

Time constraints don't allow for the mundane activities. I have mixed results with the plants, but that is to be expected with the time I put into it, which is intentionally at a minimum.

I'd like to do as little as possible but still have some fresh produce.

An ideal system would take care of all the things I don't have time for. The more, the better.

Respondent - Raymond Thomassen

Yes, I recently bought a small hydroponic system for four plants. This is my first venture into planting things and nothing has died on me yet, so I'm really satisfied!

Thirty minutes per week, I simply set up the system on my kitchen workbench and planted the seeds.

I make sure the water reservoir is filled and has appropriate nutrient levels with a PPM measurement device. [Editorial note, PPM is a measurement of electrical conductivity.]

Same answer as the last question.

My plants are still young and hasn't required much more care as of yet.

I'm very satisfied with spending the least amount possible on these activities, which is why I started growing in a hydroponic system. It's great!

An ideal system would take care of as much as possible, I'm not experienced enough yet to know all the details but what appealed to me with hydroponics was that I didn't need to do repetitive tasks like watering every few days.

Conclusion from interviews

The respondents requirements were all harmoniously in line with spending less time on repetitive tasks for optimizing the growing conditions for the individual plants. For respondents that used soil as medium, watering and fertilizing seems to be the most time consuming activity. The respondents that had experience with hydroponics had seemingly positive experiences with using these systems and spent less time on watering at the expense of spending a bit more time with making sure nutrients were at a correct level.

An automated system that takes care of the smaller details of the growing environment to ensure optimal growth should have an appeal based on these interviews. Follow up questions in regards to climate and water quality were unappealing to this level of recreational gardening users, mostly for time constraint reasons - but they all seemingly wanted an automated system that would take care of these things for them in a 'set and forget' type of manner with minimal effort required on

their part.

3.2 User stories

Building upon the findings from the interviews, I developed a list with short description of features (i.e., user stories) that a potential user would require form an automatic hydroponic system. The user stories later were used to define the functions (i.e., functional requirements) and the behaviors (i.e., non-functional requirements) of the system I have been developing for this project.

1. As a user, I want to spend less time watering my plants for increased productivity.
2. As a user, I want the water temperature to not get too cold so my plants will stay alive even during winter.
3. As a user, I want the nutrient levels to be at a optimal level level so my plants will thrive and produce maximum yield.
4. As a user, I want the ph levels to be within optimal levels so my plants will thrive and stay healthy.
5. As a user, I want the air temperature to not get too cold so my plants will stay alive even during cold winter days.
6. As a user, I want the air temperature to not get too warm so my plants will stay alive even during hot summer days.
7. As a user, I want the humidity to not be too low so my plants don't dry out.
8. As a user, I want the humidity to not be too high so my plants get infected by mold.
9. As a user, I want to be able to see what the data parameters are for monitoring the plants over time.
10. As a user, I want to be able to easily define and change the ranges for temperatures, humidity, ph and nutrient levels to ensure optimal growth for different plants.

3.3 Risk analysis

For a successful implementation and in an attempt to get ahead of any potential problems, a risk analysis has been conducted along with possible steps to mitigate said risks in the project.

Risk	**Probability**	**Consequences**
Power outage	Low	Dead plants
Electrical fire	Low	Personal injury, property damage, dead plants
Water leak	Low	Property damage, electric shock, dead plants
Component failure	Medium	Sick plants
Component inaccuracy	High	Sick plants
Data breach	Low	Information disclosure
Sabotage	Low	Dead plants
Salmonella	Low	Personal injury
Pathogens	Low	Property damage, sick plants
Pests	Low	Sick plants
Empty water reservoir	High	Dead plants

Table 2: Risk analysis

3.3.1 Mitigation

Electrical fire

To minimize the risk of electrical fire a waterproof protective casing will be built, containing all the electrical outlets, relay module and microcontroller.

Water leak

For this particular type of hydroponic system the chances of a water leak is very small as the water remains in the same container at all times. A slightly larger tub or container than the water reservoir should suffice to prevent water leaks.

Water damage can lead to permanent and serious property damage and mold growth, not necessarily just a mess to clean up. Water is also highly damaging to electronics and electrically conductive, so it's imperative that the user keep electronics devices away from areas that can be affected by a leak.

Component failure

Unplugging the signal cables for the sensors will cause them to output nonsensical values (i.e, -45 degrees celsius and 0% humidity) that should give an indication to the user that something is wrong. Using the visualization dashboard can confirm if a sensor has failed, and for long it has been out of operation.

There are a number of reasons why a component can fail, sometimes as undramatic as poorly connected wires. The fix can potentially be as easy as restarting the application on the Raspberry Pi, which comes with a HDMI connector that works with any modern monitor. To ensure wires are less likely to be disconnected we will use electrical tape on connecting joints.

The sensory equipment for this project is relatively cheap, so having spares can prove beneficial.

Component inaccuracy

Over time components can become less accurate, this especially applies to the ph (i.e., water quality) and ec (i.e., electrical conductivity) sensors used in this project. It's recommended to do a health check for the components at intervals using calibration solutions.

As is the case for component failure, the sensory equipment for this project is relatively cheap and higher accuracy can be obtained by buying more expensive sensory equipment. However, the Johnny-Five framework documentation should be consulted in regards to support for the replacement sensor.

http://johnny-five.io/api/multi/

http://johnny-five.io/api/thermometer/

http://johnny-five.io/api/hygrometer/

Data breach

Optimal growth values for plants is well documented and is likely not worth investing a significant cost or effort into protecting. However, both Raspberry Pi OS and Grafana deploys with default users and passwords that are well known, and its beneficial to create unique users and passwords for good practice. It is also recommended to keep the Raspberry Pi in a secure trust boundary with restricted physical access. In this project, the electronic box will double as a physical boundary.

For the purposes of this hydroponic system, there is no compelling reason to have the Raspberry Pi connected to the internet. Iptables will be used to block network access to the Raspberry Pi outside of a private network.

Sabotage

Hostile actors could potentially want to sabotage the crops, and the hydroponic system relies on

all the components to function as intended. Many of the aforementioned points that were made in regards to data breaches also apply to sabotage.

Pathogens

As a preventive measure to avoid pathogens such as algae which can harm the root system of the plants, the water reservoir will be spray painted and pot sponges will be utilized to reduce levels of light reaching the nutrient water.

Empty water reservoir

Left unattended the water reservoir will eventually dry up, and the crops will die. Some manual intervention to make sure the reservoir is adequately filled is needed for this problem.

3.4 System requirements

3.4.1 Functional

Requirement	FURPS category	Priority
The system must collect sensor data	Functional	High
The system must save sensor data to a database	Functional	High
The system must display sensor data to an end user	Functional	High
The system should visualize graphs from sensor data	Functional	Medium
The system must perform air climate regulation	Functional	High
The system must perform water quality regulation	Functional	High
The system should be able to relay data queries	Functional	Low
The system must send readable error messages	Functional	High

Table 3: Functional requirements

3.4.2 Nonfunctional

Requirement	FURPS category	Priority
The system must be able to successfully grow plants as stated in the sub-section 2.2	Usability	High
The system must have an enclosed environment suited to perform regulatory actions	Usability	High
The system must have a configurable range of parameters for the desired environment as stated in the sub-section 2.3	Usability	High
The system must be able to run on a Raspberry Pi	Usability	High
The system should automatically start when Raspberry Pi is powered	Reliability	Medium
The system must only allow access to administrators	Security	High
The system should fit in a protected casing to avoid the risks identified in sub-section 3.3	Reliability	Medium

Table 4: Nonfunctional requirements

3.5 Hardware

3.5.1 Arduino Uno

Arduino Uno is a prototyping microcontroller that provides 6 analog pins and 13 digital pins to connect sensory equipment. The data from these sensors are then used to control relays connec-

ted to electrical devices that assist in regulating the environment the plants grow in for optimal conditions.

To facilitate communication between the Raspberry Pi and Arduino through a USB port, the latter will be loaded with a sketch that includes the Firmata protocol. The existing documentation breadth of projects built on Arduino residing on the internet is considered favorable for creating a minimum viable product.

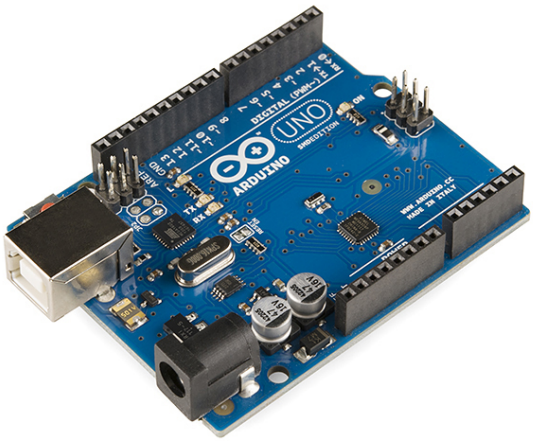

Figure 5: Arduino Uno

3.5.2 Raspberry Pi

Raspberry Pi is a tiny computer built on a single printed circuit board. It fits this project perfectly with the specifically built Raspberry Pi OS implementation of GNU/Linux. This will facilitate storing data in a database and running a wide range of development environments locally with assistance of the package manager.

Using a host computer will enable this project to use a simple microcontroller without wifi support, and to exclude the costs of cloud solutions is beneficial for this project in regards to the budget cap, as well as making security less complicated to manage.

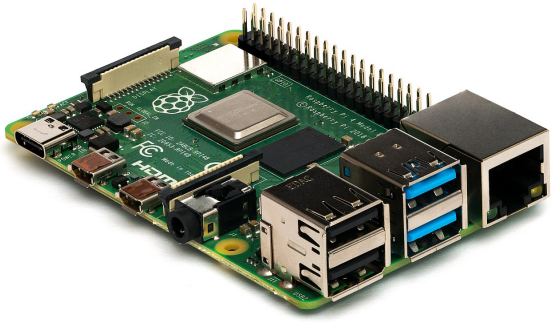

Figure 6: Raspberry Pi 4 Model B

3.6 Components

Hydroponic system

- Reservoir
- Net pots
- Pot sponge
- Air pump
- Air stone
- Silicon tubing

Sensors for climate

- Temperature and relative humidity - SHT31D
- Light - KY-018 Photoresistor

Sensors for water quality

- Temperature, waterproof - DS18B20
- Nutrient levels - Keystudio KS0429 TDS meter
- PH - DFRobot ph sensor

Electrical devices for regulation

- Arduino relay module
- Air heater
- Air fan
- Heating pad
- Peristaltic pumps

Other electrical utilities

- Electronics protective casing
- Voltage transformer 230/12V
- Plug outlet
- Plug extension cord

Prices are listed per unit excluding delivery cost.

Component	Count	Price
Ikea Samla plastic container - Reservoir 39x28x14cm	1	€2,9
LED Grow light full spectrum	1	€6,25
Net pot - 48,5mm X 70mm	9	€0,16
Pot sponge - 45mm X 30mm	9	€0,05
Air pump	1	€2,9
Air stone	1	€0,25
Raspberry Pi 4 Starter kit - 2GB Ram	1	€61,55
Arduino Uno	1	€2,7
Arduino Uno protective casing	1	€0,75
Breadboard	x	€0,25
8 relay module	1	€3,7
KY-011 LED module	2	€0,25
KY-018 Photoresistor module	1	€0,25
SHT31D - temperature and humidity sensor	1	€3,4
DS18B20 - waterproof temperature sensor	1	€1,45
Keystudio KS0429 TDS meter	1	€6,50
DFRobot ph sensor	1	€26
INTLLAB 12v 5w peristaltic pump	4	€4,1
Hamron voltage transformer 230/12V	1	€30
Air heater	1	€20
Fan	1	€3,15
Heating mat	1	€5,8
Ultrasonic mister	1	€8,05

Table 5: Component list with cost

3.7 Development environment

3.7.1 Node.js - Javascript runtime

This project uses Node.js runtime as a development environment on a Raspberry Pi for governing the microcontroller over coding directly on the Arduino Uno, because it gives a wide range of extra potential in regards to storing the data locally and using a third-party solution such as Grafana for visualizing this data. Node.js works asynchronously which fits perfectly for the nature of this application. This allows for easily running actions like data collection and regulatory actions on timed intervals.

Furthermore, Node.js also offer a substantial ecosystem in terms of frameworks and libraries, which will assist in the projects goal of making a minimum viable product in the allotted amount of time.

3.7.2 Express.js - Web API

Express.js is a web API framework within the Node.js ecosystem that is used in this project to fetch data from sensory equipment and the database, and then serve the query results to a user in JSON format.

3.7.3 Johnny-Five - IoT framework

Johnny-Five is a IoT framework within the Node.js ecosystem that enables the application to connect to a microcontroller to read sensor data and perform actions with relays. The framework is favorable for this project because it has integrated support for general categories of components that is needed to meet the system requirements. The documentation has been referenced prior to

purchase of components to make sure the specific components that are used will integrate smoothly with our application.

3.7.4 Grafana - Data visualization

Grafana is a analytic and interactive visualization web application that supports connecting directly to a data source and creating custom dashboards. This will help the project achieve professional looking graphs for our data and will serve as the applications client layer for the end user. The documentation has been referenced to make sure support for the database used in the application is set up.

3.7.5 InfluxDB - Time series database

InfluxDB is an open source time series database that supports both Node.js and Grafana and fits our use case. Using a time series databases provides the project with a system that is optimized for storing and serving timestamped key-value pairs that contain our sensory equipment data. The database can be used as a direct source for Grafana which will help creating a very satisfactory minimum viable product in the time frame allowed by the project. InfluxDB provides a client API for using the database with Node.js.

3.8 Architecture

The architecture of this system is client-server where the server code is run on a Raspberry Pi server with Node.js runtime. The server application controls the arduino microcontroller through Johnny-Five framework and serves a queryable web API with Express.js. It also polls the sensory equipment for data on 5 minute intervals, and the sensor measurements are saved in InfluxDB.

On the client side of the app, an end user can connect to the Raspberry Pi using a web client and display visualization graphs of the measurements with the assistance of Grafana.

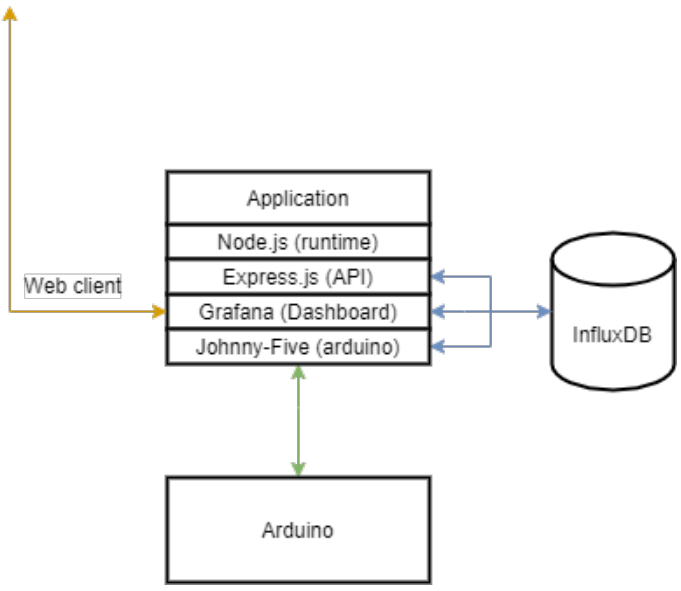

Figure 7: Application layer workflow

The system uses a relay module to start or stop electrical devices and pumps that control air climate and water quality. Threshold values are defined in the configuration and if a sensor exceeds this,

the appropriate electrical device or pump will take corrective action. In addition, a LED light will also pulse to indicate to the user that a value is exceeded.

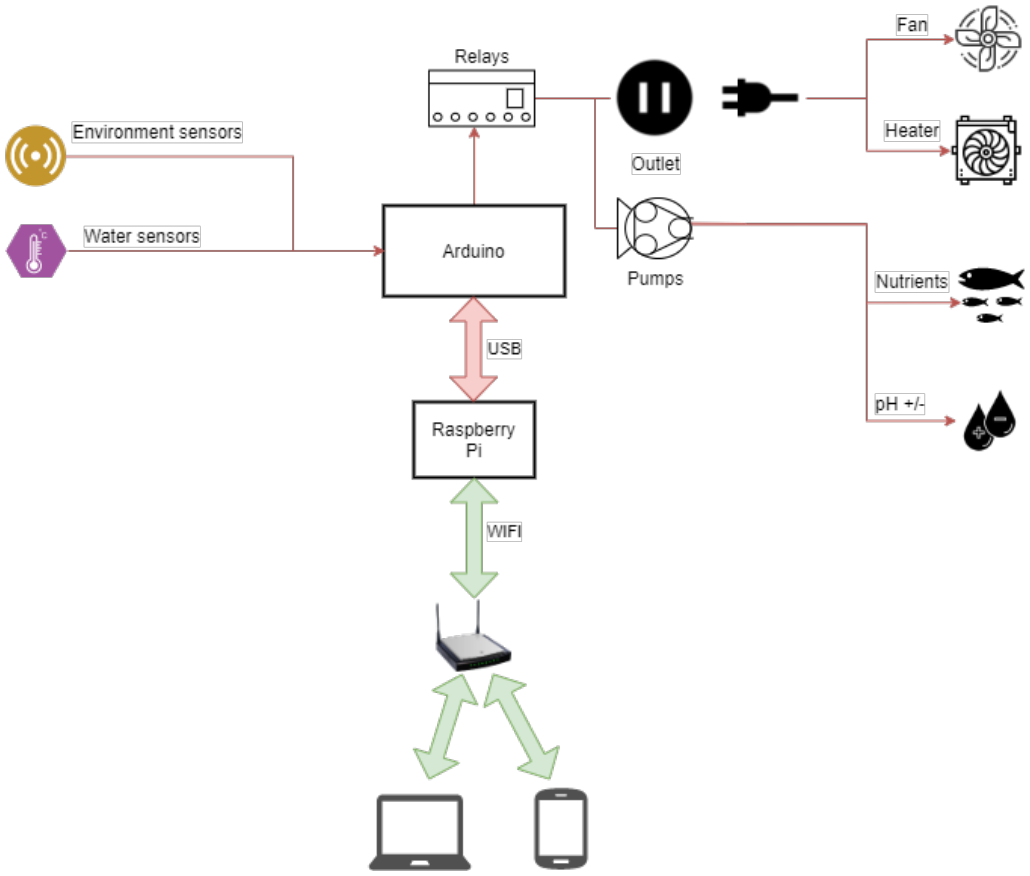

Figure 8: Device communication workflow

Green arrows indicate workflow between a user's web client and the server. Red arrows indicates workflow controlled by the application.

3.9 Modules

3.9.1 Configuration

—config.js

The configuration consolidates important parameters for the application, such as database url and access token, what pins are used to connect the different sensory equipment and threshold values to enable regulatory devices for optimal growth. If there is a need to change any of these parameters its easy to find them without going through all the code in the application.

3.9.2 Setup

—setup.js

The setup file initializes InfluxDB with the necessary bucket, username, password and an access token.

3.9.3 Server

—server.js

The server module is the main part of the application, where it connects to the micro controller, set up sensory equipment, describe functions for getting data from the sensors, save measurements to the database and publish data to the web API. This is also where the logic for regulating the electrical devices is executed.

3.9.4 Client dashboard

—dashboard.json

Imported into the web interface of Grafana after a InfluxDB data source has been set up in the web UI.

3.10 Wire frames

A wire frame for the client UI has been created to turn the abstract idea of a user interface into something tangible to base the work upon.

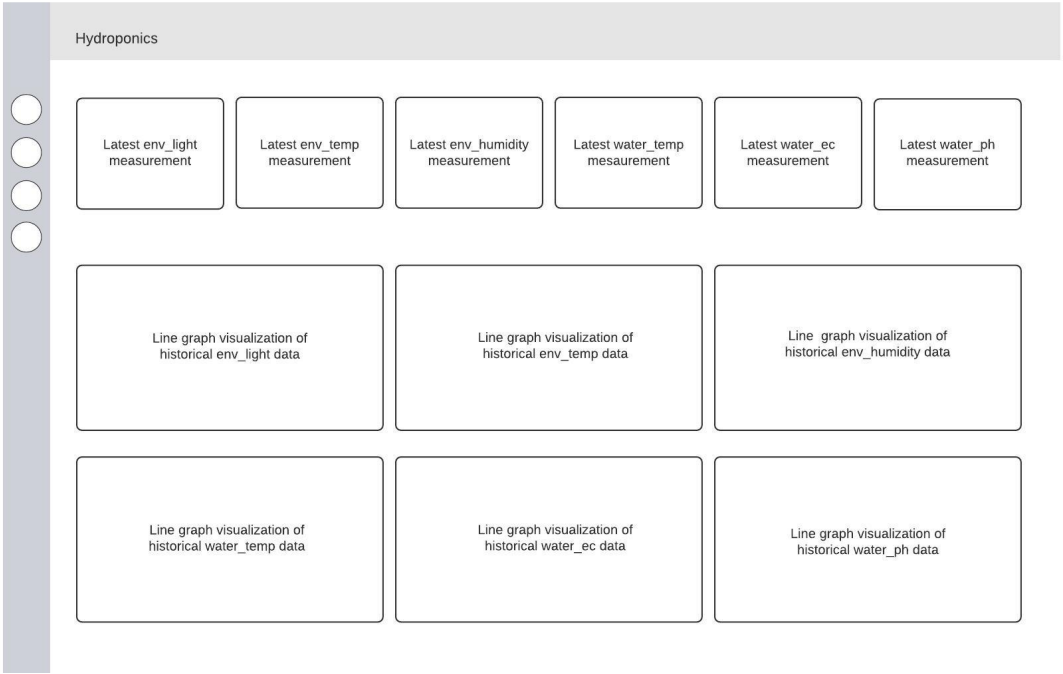

Figure 9: Wire frame for the user interface

3.11 Wiring diagram

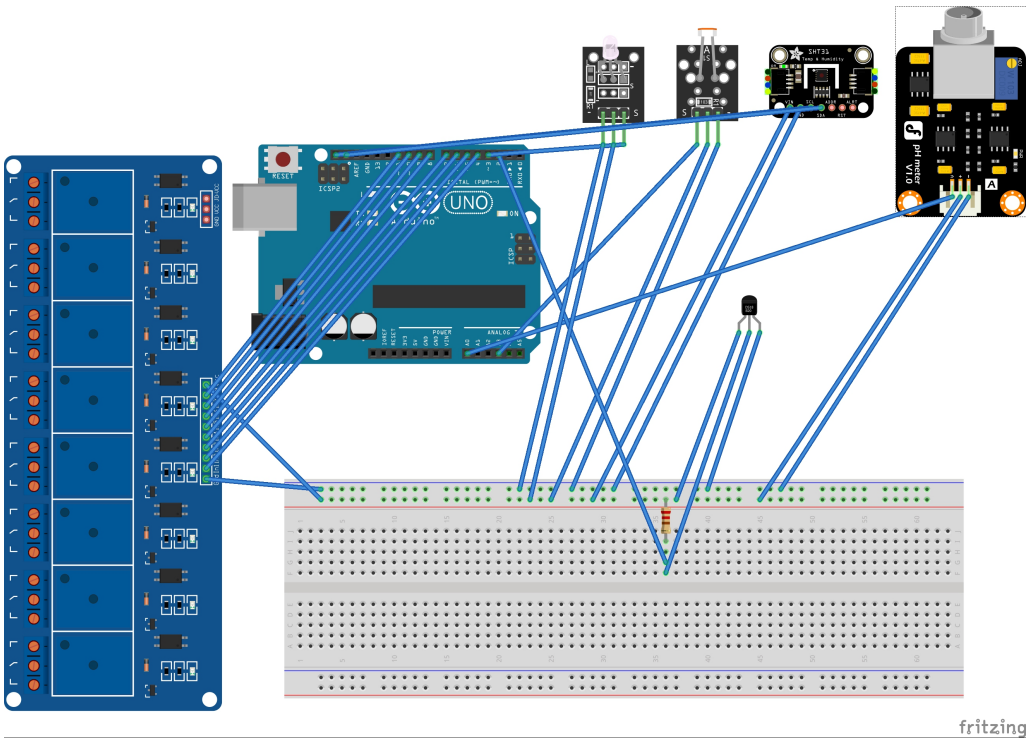

Figure 10: Wiring diagram for the microcontroller and components

To connect the microcontroller with the components a breadboard is used. First the Arduino is connected to the breadboard through the 5v PIN from Arduino to + on the breadboard, and the GND PIN from Arduino to - on the bredboard.

The relay module is then connected to the power circuit by wiring VCC to + on the breadboard and GND to - on the breadboard. The PIN numbers that send signals to the relay module can be defined in config.js, and is set at PIN 4 to 11 by default. Wires are connected from the respective PIN numbers on the Arduino to IN1-8 on the relay module. In this case the IN1-4 are 12V powered pumps, and IN5-8 are 230V powered electrical devices.

The sensor modules are all connected by their VCC pins to + on the breadboard, and GND to - on the breadboard. Signal pins can also be defined in config.js. The LED module is connected to PIN3, the photoresistor module is connected to A3 and the SHT31D is connected by SCL and SDA to the equivilant pins on the Arduino. The TDS meter is connected to A0 and the ph meter is connected to A1. The DS18B20 sensor requires a 4.7k ohm pull-up resistor that is connected with one end to + on the breadboard, and the other end to the same row that is connected to the signal connection from DS18B20. Another cable from this row is connected to PIN 2 on the Arduino.

Keystudio has not made any fritzing diagram available for the TDS meter, and when I asked them for one they were not compliant, so this component does not feature in the wiring diagram but is still a part of the final product.

4 Design and prototyping

Building upon the product plan there were some prerequired steps that needed to be done in order to set up and test the system. Hardware and software had to be prepared, and some parts needed prototyping to fit the various components involved.

4.1 Hardware setup

4.1.1 Arduino Uno

The Arduino needs to be loaded with a Firmata sketch to facilitate communication through Johnny Five, and the water temperature sensor spesifically requires a variant called ConfigurableFirmata. This is installed in the Arduino IDE in the Library Manager, and the sketch is loaded by navigating to File - Examples - ConfigurableFirmata - ConfigurableFirmata. This will take up 95 percent of the storage available on the Arduino, so to ensure stable operations the libraries for StepperFirmata and AccelStepperFirmata is commented out as we wont be using any stepper engines in this project. With the sketch loaded and Arduino plugged in, ConfigurableFirmata is loaded onto the microcontroller by pressing the upload button after the changes.

4.1.2 Raspberry Pi

To get the Raspberry Pi up and running it has to be loaded with a operating system in the storage system, a micro SD card. This is easily done with a USB card reader and a specially built application called Raspberry Pi Imager. In order to use InfluxDB version 2 a 64bit OS is required, and the choice falls on Ubuntu for ease of use and a solid package manager. Ubuntu server 20.04.2 LTS is selected from a sub menu for its long term support and minimalistic footprint, and is written to the micro SD card.

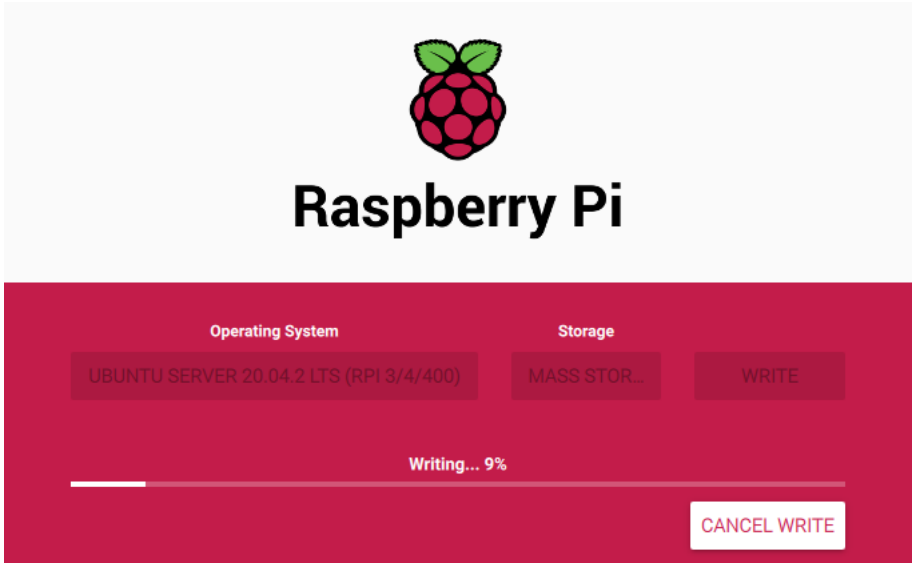

Figure 11: Raspberry Pi Imager

To set up the computer a HDMI cable, USB keyboard and Ethernet cable is connected in order to get the wifi up and enable SSH tunnel communication. First the keyboard layout is set to Norwegian, before timezone is set to Oslo and hostname altered to reflect its purpose, and a new user is added and modified to include the groups 'sudo', 'dialout' and 'tty'. This will allow the new user to run administrator commands and access the USB ports.

```
sudo dpkg-reconfigure keyboard-configuration
sudo timedatectl set-timezone Europe/Oslo
sudo hostnamectl set-hostname hydroponics
adduser marius
usermod -a marius -G sudo dialout tty
```

To set up the wireless LAN some new packages are needed and a config file has to be created. This

should bind the wlan0 interface to the wireless LAN automatically and at boot.

```
sudo apt install wireless-tools net-tools ifupdown
sudo vim /etc/network/interfaces
```

—/etc/network/interfaces

```
auto wlan0
iface wlan0 inet dhcp
wpa-ssid YOUR_SSID
wpa-psk YOUR_PASSWORD
```

With the basic setup of the system, now is a good time to add a SSH public key for authenticating the newly created user. This ensures that no passwords are sent in clear text over the network, and that no one can remotely login to the system without holding the private key. First the server keys are generated, before Putty is used to generate a new set of keys on the client computer that wants to communicate with the Raspberry Pi. Then the client key is transferred before enforcing the new SSH rules enabling public key authentication.

```
cd /etc/ssh
sudo ssh-keygen -A
mkdir ~/.ssh/
vim ~/.ssh/authorized_keys //Copy the generated key from putty into this file
vim /etc/ssh/sshd_config
```

—/etc/ssh/sshd_config

```
PubkeyAuthentication yes //Enables public key authentication
AuthorizedKeysFile .ssh/authorized_keys //Sets the location of the key file for users
PasswordAuthentication no //Disables password authentication

sudo service ssh restart
```

These steps should allow for SSH communication, and as such the HDMI cable, USB keyboard and Ethernet cable are all removed. Its time to upgrade the system for the latest bug and security fixes to the system packages, and install the software required to run the application. In case of a system restart, its also useful to enable automatic startup services for InfluxDB and Grafana.

```
sudo apt update && sudo apt upgrade -y && sudo apt autoremove -y
sudo apt intall build-essential libfreetype6 fontconfig-config libfontconfig1 //Prerequisites for

curl -fsSL https://deb.nodesource.com/setup_lts.x | sudo -E bash - //Add node.js repository
sudo apt install nodejs -y

wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.0.6-arm64.deb //InfluxDB package
sudo dpkg -i influxdb2-2.0.6-arm64.deb -y
sudo service influxdb start && sudo systemctl enable influxdb.service

wget https://dl.grafana.com/oss/release/grafana_7.5.5_arm64.deb //Grafana package
sudo dpkg -i grafana_7.5.5_arm64.deb -y
sudo service grafana start && sudo systemctl enable grafana-server.service
```

With Node.js, InfluxDB and Grafana up and running, the application for controlling the hydroponic system and do data logging is next. Its also beneficial to make a startup service for the app for the same reasons as the database and dashboard. The application also has a setup.js file for creating the necessary InfluxDB assets, such as bucket and authentication token.

```
git clone https://github.com/mariusnorheim/hydroponics-datalogger.git
cd hydroponics-datalogger
npm install
node setup.js
sudo vim /etc/systemd/system/hydroponics-datalogger
```

—/etc/systemd/system/hydroponics-datalogger

```
[Unit]
Requires=systemd-networkd.socket
After=systemd-networkd.socket

[Service]
ExecStartPre=/lib/systemd/systemd-networkd-wait-online --interface=wlan0
ExecStart=/usr/bin/node /home/marius/hydroponics-datalogger/server.js
Restart=always
StandardOutput=syslog
StandardError=syslog
SyslogIdentifier=hydroponics-datalogger
Environment=NODE_ENV=production
User=root
Group=root

[Install]
WantedBy=multi-user.target
```

```
sudo systemctl enable systemd-networkd-wait-online.service
sudo service hydroponics-datalogger start && sudo systemctl enable hydroponics-datalogger.service
```

With the microcontroller connected to the appropriate USB port, the application should be up and running, with a system service that enables it on startup.

For accessing the InfluxDB administration page the end user can visit http://ipaddress:8086 and enter the influxdb credentials listed in config.js. The Grafana UI can be reached at http://ipaddress:3000 with the default username and password combination admin / admin. Upon logging in to Grafana the user will be prompted for a new secure password. If this is ever lost it can be reset from the Raspberry Pi's console with the following command

```
sudo grafana-cli admin reset-admin-password newpassword
```

4.2 Grafana - User interface

Working from the wire frames for the UI, a Grafana dashboard has been created. The layout has been configured to fit all the realtime data in one length, and three line graphs of historical data per screen length in two separate categories. The historical data is by default set to display data from the last 6 hours, and in order to get useful and detailed information along with pleasing aesthetics the line graphs require more screen space than the real time data that are just showing a single number.

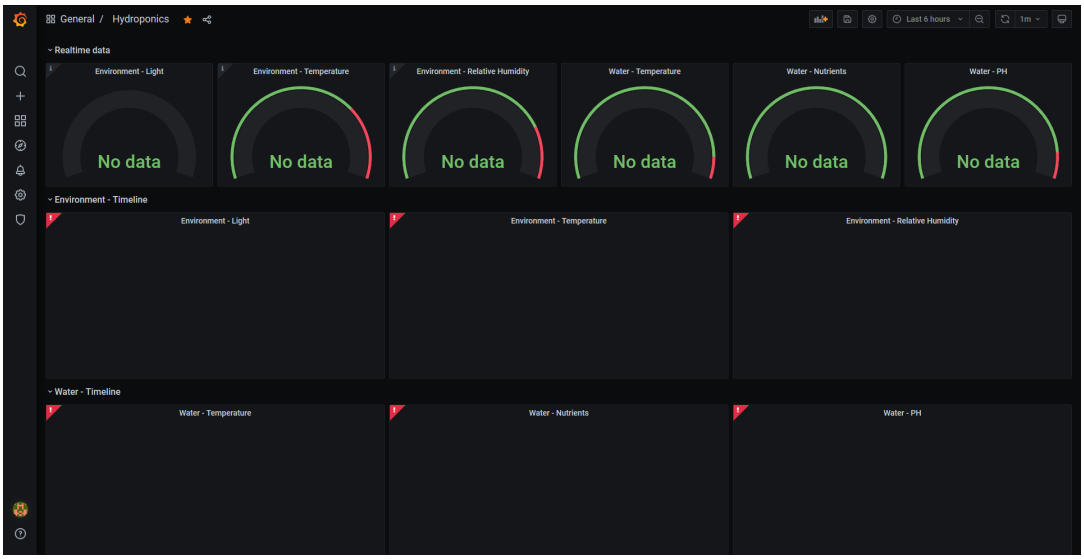

Figure 12: Grafana layout

Displaying the latest values for all sensors at the very top in a gauge style makes for a better *visualization* than just a number. It also enables a visual indicator for the maximum threshold values for each sensor that gives information to the user about how close any given measurement is to exceeding said threshold. The gauge gives a visual appearance much like a speed limiter on a car, and the user intuitively can see what the current speed is, what the the speed limit is, and how close they are to the speed limit without thinking about it or making calculations. Additionally the gauge will catch the users attention, and along with green and red color they will instantly know what the status is for each parameter.

Placed below the real time data are line graph visualizations showing a more complete historical data for each sensor. The line graph visualizations have been grouped together in two categories, one for air environment and one for water in a logical manner. Line graphs offers a logical and non-intrusive way for users to observe trends over time without cluttering the graph with too much information, like a bar chart or scatter plot would in this case.

The user can decide the time interval shown in the graphs (i.e, data from the last 6 hours) as well as how often Grafana should update to retrieve new values from the database in the upper right corner of the website. It is also possible for the user to select a time interval in the visualization for closer analysis.

The colors red and green act as natural status indicators, with red signaling danger and green that something is ready or successful. They also compliment the dark background nicely and function as a way to emphasize the primary focus of any visualization, the data. The darker background is used over its white counterpart, because it makes the visualizations stand out more and is easier on the eyes. White is very dominating and intrusive to the eyes, and there are very few colors that really stand out to catch the users attention when using a white background.

4.3 Miscellaneous parts

The peristaltic pumps need something to suspend them safely inside the box, and I found some leftover metal bits that would fit the job. I measured the distance needed to fit two pumps per bracket, and proceeded to use a step drill to make the holes for the pumps. I then used a regular drill bit to make the smaller holes for mounting the pumps and for mounting the bracket to the box. After drilling, the edges were smoothed and the holes deburred with a metal file to prevent cuts.

Figure 13: Brackets for the peristaltic pumps

I also needed something to attach the sensors for air climate to and the LED diode. Because metal is electrically conductive I decided to use plexiglass and attach the sensors with locking strips. A regular drill bit was used to make the necessary mounting holes.

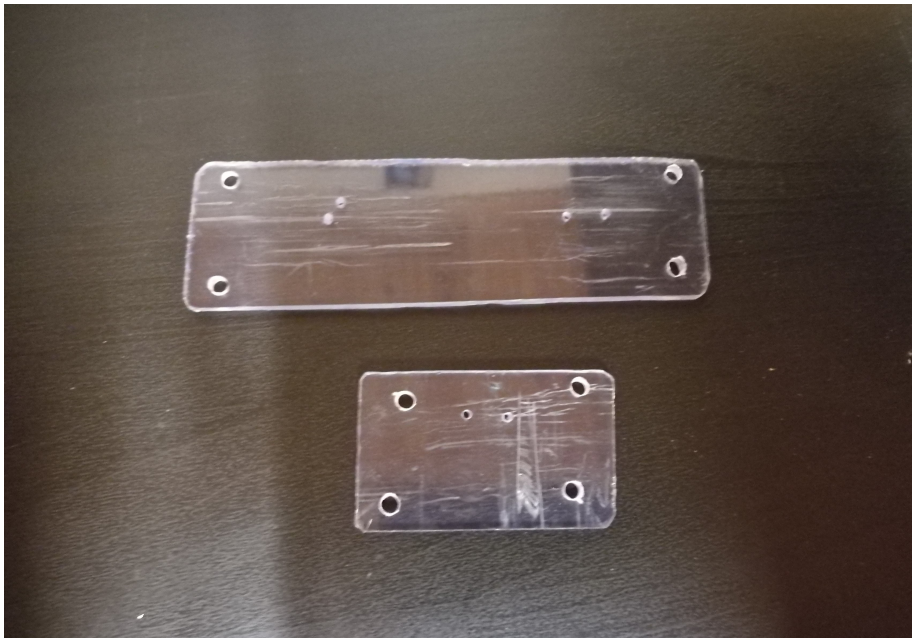

Figure 14: Brackets for sensors

4.4 Water reservoir

The water reservoir was made from a Ikea plastic container that had 9x50mm holes cut out for net pots. Holes were also drilled to fit hoses for the peristaltic pumps on one side, and for the air pump on the other side. After drilling the holes it was painted white on the side facing out as a

means to reduce algae growth and reflect light to the underside of the leafs.

Figure 15: Water reservoir

4.5 Grow box

The grow light is mounted with two included brackets at the top of the box, and the air fan was fitted by using a 120mm drill saw and mounted with screws and lug nuts. The pump bracket was made with two mounting holes and secured with screws and lug nuts as well. The sensory equipment for climate will be attached to the plexiglass bracket, which was mounted on the box with four screws and lug nuts. Plastic cable holders with adhesive tape are placed strategically in order to separate cables for 12V and 230V power sources to minimize electromagnetic interference.

Figure 16: Fitting the grow box

4.6 Electronics box

For this project I decided to use a separate waterproof electronics box as we are dealing with water, which mixes badly with electrical components in case of an accident or leak. I mounted a piece of plexiglass in the bottom for the purpose of fitting terminal blocks, the relay module and breadboards. An extension cord module with 7 plugs is used to power devices such as grow light, air pump and the Raspberry Pi.

There's two terminal blocks installed, one for 230V devices and one for 12V devices. A cord plugged into the extension module supplies power to one terminal block, and a 230/12V voltage transformer is used to supply power for the other terminal block.

Creating a plug for the 12V power source will allow for easy disconnect and to separate the electrical housing from the grow box if needed. One end is connected to the NO connector of the relays and ground on the 12V terminal block, while the other end is connected to the peristaltic pumps in the grow box.

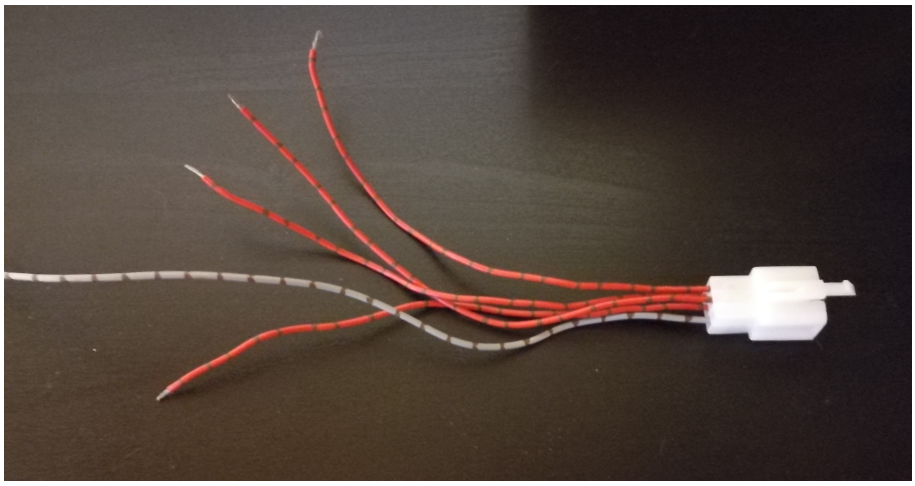

Figure 17: 12V power plug for the pumps

The electronic devices regulating the grow environment are controlled with a relay module. A relay has 3 input connections - Normally Open (NO), Common (COM) and Normally Closed (NC). The COM port is where the relays draws the power to run electronic devices. In electronic circuits, "open" and "closed" have inverse logical meaning to what one might be used to. An open circuit means that the power is not throughput and thus is turned off, and a closed circuit means that the power is throughput and thus turned on.

If we use a push button as an example, connecting a device to the normally open connector means that it should stay disconnected until the button is pushed to turn the device on, when the push button is released it goes back to disconnected. For the normally closed connector it means that it should stay connected until the button is pushed to turn the device off, and when the push button is released it goes back to powering the device.

Because we want to regulate the grow environment when the threshold values for temperature, humidity, ph or nutrients are exceeded in some capacity, it makes sense to always have the electronic devices powered off until explicitly told to turn on, so we will connect everything to the NO circuit of the relays.

We use four individual power cord extensions to power the 230V devices and cut them to expose the cables. Norwegian power cables are bundled into three separate cables, one specifically for ground, one for - and one for +.

All ground and - cables on the extension cords are connected to the 230V terminal block, and the + is connected to NO circuits on individual relays. Four cables are created from the leftover

extension cords and connected from + on the terminal block to COM on the relays, this will result in the devices connected to the extension cords only are given power when the relays are turned on.

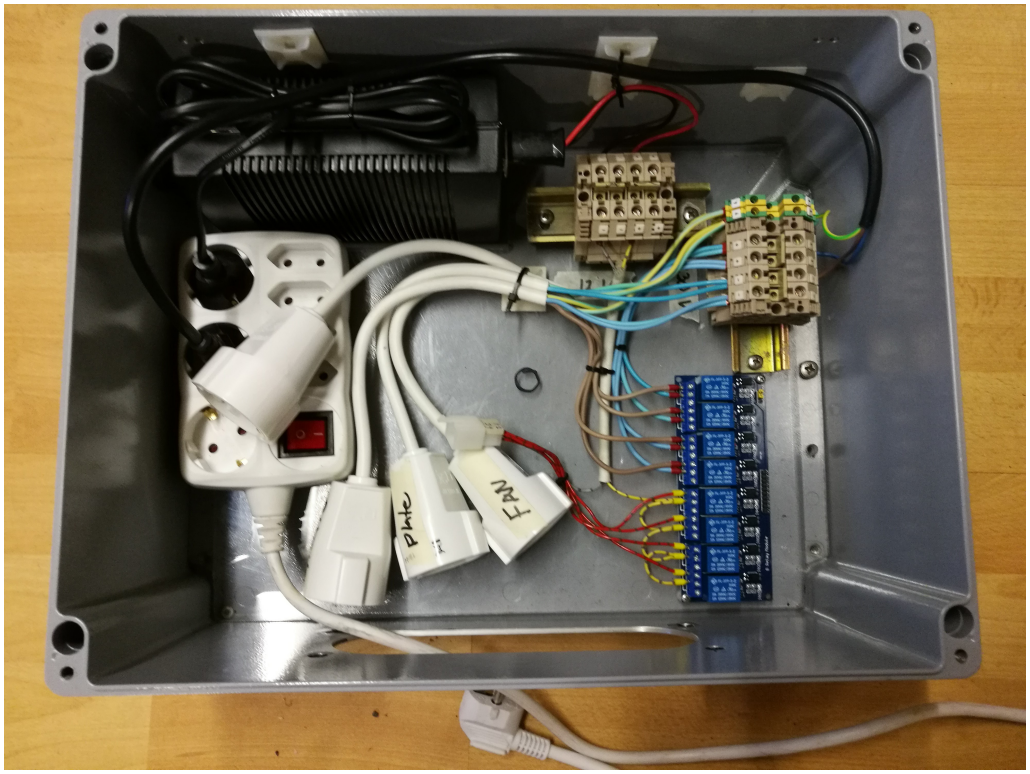

Figure 18: Fitting the electrical housing

The COM connections for the 12V pumps are connected in series due to the low voltage between the relays and then to + on the terminal block, and the ground connection on pumps is connected in series from one pump to the next, similar to how the relays were connected to the terminal block in the electrical box. All wires are fitted with an end connector to ensure no cables will inadvertently get disconnected.

5 Set up and system testing

Before testing the complete system, calibration of the PH and EC sensors had to be done, and the pumps needed to be tested with nutrients and PH +/- liquids to find a optimal pumping duration that would not adjust too much. The ph + liquid contain potassium carbonate and potassium hydroxide, while the ph - liquids contain nitric acid for plants in the growing stage. These are strong chemicals that can cause burns to human skin, so we want to dose carefully when regulating the water quality as to cause minimal stress to the plants.

5.1 Sensor calibration

Calibration on the ph sensor is done by adjusting a potentiometer screw on the ph module, and setting an ”offset” variable in the code function that calculates the ph value. The code will first calculate the voltage of the fluid it is submersed in, and then convert said voltage into ph by multiplying 3.5 * voltage + calibration offset.

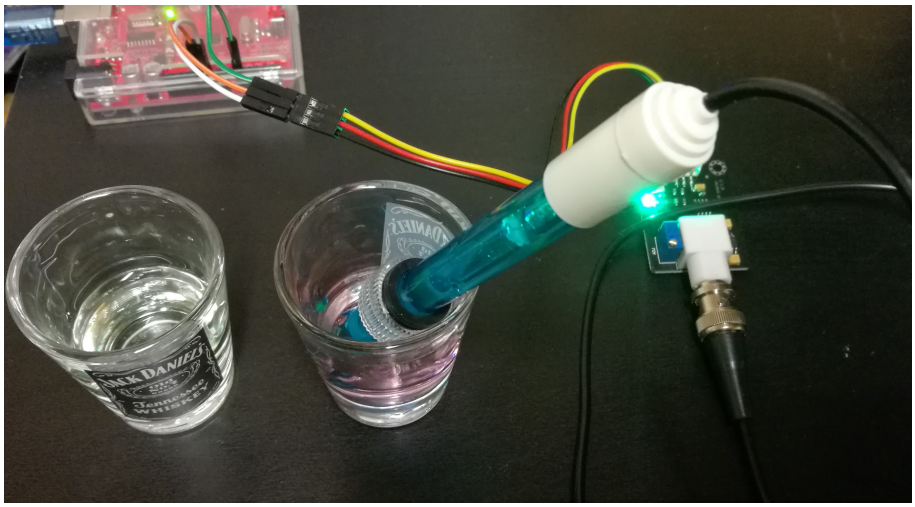

Figure 19: Calibrating the PH sensor

Two calibration fluids were used, one for ph value 7 and one for ph value 4. Without utilizing the offset value, the readings were way off the mark on the lower end when using the ph 7 calibration liquid. I started off with setting the offset value fairly high - at 0.70. I then proceeded to adjust the potentiometer until a reading of 7 was returned, before cleaning the probe in water and wiping it off before making a reading of the ph 4 calibration liquid. The readings from the ph4 liquid were a bit too high, so I adjusted the offset incrementally while using the potentiometer to finely tune the readings. With an offset of 0.75 I ended up on a correct reading for both ph liquids. This should ensure accurate readings from the probe within this two point 4-7 calibration range.

5.2 Intervals for pump dosing

Warning and word of advice - The chemicals used in the ph regulation liquids should be handled with care, and measures to protect exposed skins and eyes should be taken when handling these liquids. Handle in a well ventilated room. Always read the warning labels. If there is uncertainty about the chemical reactions at play, ask someone with experience. ***Ingesting, inhaling vapours or skin contact could lead to personal injury. Stay safe!***

As previously mentioned, the ph adjustment liquids are extremely potent and the instructions call for 5ml of ph- liquid and 14ml of ph+ liquid in a 100L water tank. Using tap water I filled an secondary water reservoir with approximately the same amount of water that would be used in production, and inserted the ph probe. My tap water was slightly alkaline with a ph value of 7.7, and a pipette was used to adjust the ph with four drops. This dramatically lowered the ph to 5.75 and confirmed my suspicions that the pumps need to be very carefully adjusted when dosing.

The threshold values for optimal ph was set to 5 as the min value and 7 as max value for the purpose of the test run. The timeout value for the pumps were set at at 0.02s for ph- and 0.04s for ph+ in the application code and the secondary water reservoir was set up with these settings to study its effects on a similar environment. This would give me fluctuating results, sometimes lowering the ph by 2.

The nitric acid solution is too concentrated for our smaller sized water reservoir and needs to be diluted. ***Following safe protocol for handling acids, we should add acid to water and not the other way around***. Adding water to acid can cause heat formation and spray acid, which would be highly unfortunate with a product that can cause burns on exposed skin. Our glass bottle has approximately 1,5dl of nitric acid which was added to another 2dl of tap water to continue testing. This step reduced the ph adjustment to a satisfying 0.5 ph value per pumping cycle.

Testing also shows that adding nutrients will lower the ph, by about 0.5 using a pipette to add 5

drops of each nutrient component.

5.3 Intervals for environment regulation

Because the air heater and mister are powerful devices that are set to do incremental gains, the interval for regulating the air environment is set at five minutes.

During the pump dosing test it became clear that the ph sensor needed a fair amount of time to adjust to the new levels of ph, and having limited circulation in form of the air pump is likely also a factor in this. When testing water regulation intervals of five minutes, the returned sensor value would not be completely accurate - causing the pumps to do more regulation when it was not needed. Increasing the timer to fifteen minutes produced more accurate readings and thus a more stable water environment.

5.4 Intervals for data points saving

In order to test the storage usage of data points, a test was run by running intervals of one minute for exactly one hour. This would allow for easy extrapolation using simple math. Before starting the test, the existing InfluxDB bucket was removed to empty test data and a new one was created. The storage space usage on the Raspberry Pi was checked with the following command

```
sudo du -h /var/lib/influxdb
```

Before initiating the test, the storage usage was at 19 MB, and after one hour with 60 data points the usage was at 26 MB for a total of 7 MB used in the duration of the test. This gives each data point a usage of 0,117 MB. If we were to save every five minutes, that would mean a daily storage use of 33,7 MB, every fifteen minutes would be 11,25 MB, and once per hour 2,8 MB. The disk space available on the Raspberry Pi is 24 GB. This means we could run the application for close to two full years before running out of storage using five minute intervals on the save points, so this factor is not deemed critical in the decision making.

Ultimately it did not seem to make a whole lot of sense to save data points at five minute intervals, even for monitoring, but more so at fifteen minutes to go along with the pump intervals. This is an acceptable rate of updates, and will ensure the system is able to run for over five years while using 4,1 GB of storage space per year. This is in part influenced by the ability to monitor real time values for all sensors through the Express web API at http://ipaddress/api/all

5.5 Relay module

The relay module will automatically turn on all the relays when given power from the Arduino as indicated by the led diodes on the module activating. This causes issues because our devices are connected in the relay connector "NO - normally open", which means that the circuit is open between the NO connector and the common connector that gives power from the terminal block - thus not forwarding power to device connected to the relay. With the relay module automatically turning on all the relays, it means that all our electrical devices are left powered on until our application is running and sending programming instructions to the relay module.

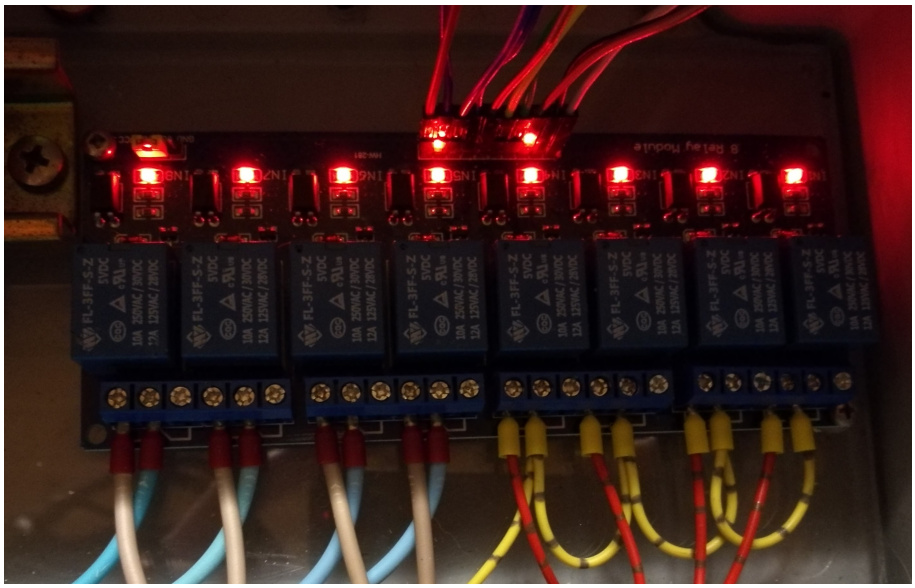

Figure 20: Powering the relay module

In an attempt to circumvent this problem, I tried changing the relay connections to electrical devices to "NC - normally closed" and use an inverse logic for handling the relays in the application. However this results in the devices being turned on for several seconds when the application initializes the microcontroller connection. This is especially problematic in regards to the pumps that will pump an excessive amount of nutrients and ph regulating fluids that will upset the chemical balance in the water reservoir should the application restart for whatever reason.

Because of the ability to disconnect the plug for the 12V pumps effortlessly, it seems like a lesser evil to have the electrical devices work in an undesirable fashion isolated to the first time the relay module is being powered on, and to make sure especially the pumps are not connected until the application is up and running. For this reasoning I decided to leave the electrical devices connected to the NO connector of relays and include a warning in the 'README' document.

Application code has been modified to power off the relays when they are initialized in the program, exit and error handlers have been added to the program code to ensure all relays are powered off if the application shuts down either through error or controlled by the user. The app service on the Raspberry Pi should ensure that it restarts if this happens.

5.6 SHT31D and DS18B20 sensor interaction

During testing on the Raspberry Pi system the application would occasionally cause errors in the runtime that it had not previously done in the development environment. The error was caused by ConfigurableFirmata timing out when trying to the read the sensor values. Another indication that something was faulty was that the SHT31D would not always respond to testing, consisting of gently breathing around the sensor to simulate high humidity and temperature and using a can of compressed air to simulate low temperatures before throwing the error, leading me to believe this sensor was the cause of the errors.

The problem was broken down into categories in order to try to identify the problem by process of elimination.

- Damaged cables
- Damaged input pin
- Damaged sensor

- Electrical interference
- Node.js environment

As the sensor had worked flawlessly during extensive testing running consistently for over a week and numerous individual days during the development process, a difference in processor architecture and Node.js environment could be a possibility.

The sensor would work when the application was restarted, hence faulty cables and input pins seemed unlikely but because it was trivial to test all the cables connecting the sensor was swapped and reconnected from SDA to A4 and SCL to A5 before restarting the application. This cause was eliminated as another error was thrown despite new cables and pins.

Another new element introduced was the Arduino and Raspberry Pi being placed in the electrical box along with other cables managing the relays. They were removed and placed further away from potential electrical interference, but this effort also proved fruitless.

When researching the error through search engines a few issues was highlighted in Johnny-Five's github repository that gave useful insights into the problem. The error was a known issue when using several DS18B20 sensors in conjunction, and in another case using DS18B20 sensor with a HTU21D sensor invoking the Multi API in Johnny-Five.

Because this project only uses one DS18B20 and there also were issues with the SHT31D sensor becoming unresponsive this indicated the issue extends to using DS18B20 sensors along with sensors invoking the Multi API, or sensors utilizing the SDA / SCL ports on the microcontroller in general. The application code was modified to separate the temperature and humidity combined sensor initialization from the Multi API to use separated variables using the Thermometer and Hygrometer APIs, and the subsequent dependencies of these variables in the code. These steps gained favorable results with the system running stable without throwing this error again until delivery of the project.

As the DS18B20 sensor and ConfigurableFirmata seems to be at the root of the problem, another solution could have been to use a supported analog temperature sensor such as LM35 or TMP36 and make a waterproof casing using a thermal conductive material to shield it from water. DS18B20 is the only sensor that requires ConfigurableFirmata, and this approach will allow for the microcontroller to use StandardFirmata. However, time constraints in the project has not allowed for this option to be further researched in depth, and is at this point only theoretical.

5.7 Grafana UI setup

A data source has to be set up for the Grafana dashboard to fetch data. A InfluxDB is selected, and the query language is selected to Flux and for HTTP url 'localhost:8086' is entered to connect to the database. In the Auth tab we enable the checkboxes for 'Basic auth' and 'Forward OAuth identity'. Basic auth details is added to the config, and is by default set to user 'admin' and password 'growbox123'. In the InfluxDB details section organization is set to 'admin', the token is also pre set in the config and is entered as 'tX6rFsAAs6zIaYKwEjv9qrXi8a-udQpmwh_5Y916DCUc5YYFoRr3-FWEcT0u8laKiO6x6J9taupV0vUPo3K1KQ==' while the default bucket is set to 'maya'.

On the left side navigation of Grafana, the included dashboard json file can be imported under 'Dashboards - Manage' and the dashboard will be available in 'Dashboards - Home' under the name 'Hydroponics'.

5.8 Requirements testing

To ensure a satisfying product is delivered the system has been tested. While some of this has been covered in the interval tests, this is a more systematic approach. At first tests were conducted in regards to all the individual requirements, followed by a test in a production environment running for two weeks without malfunction.

5.8.1 Functional requirements

1. The system must collect sensor data

This requirement is implied by all subsequent functional requirements, and further elaborated on there.

2. The system must save sensor data

By enabling InfluxDB through the configuration file, the system is able to send sensor data to the database for saving every fifteen minutes. This is confirmed through the database administration website and also in the Grafana user interface that displays line graphs with the data.

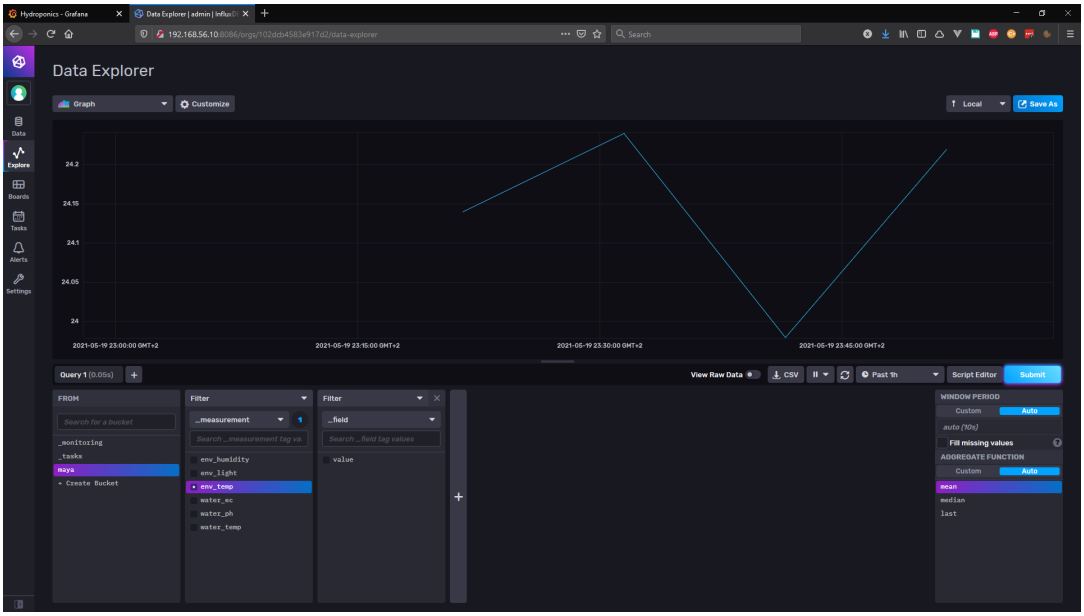

Figure 21: Testing - Save data

3. The system must display sensor data to an end user

When the application saves sensor data to the database it also displays the data captured verbosely in the console in the case that someone wants to run the application without setting up InfluxDB and utilize the data saving capabilities.

```
Air climate - Light: 96, Temp: 24.11, Humidity: 44.86
Water quality - Temp: 20.94, PPM: 592.03, PH: 6.77
Saving sensor data
```

Figure 22: Testing - Display data

This requirement is also further elaborated on in the following requirement(4).

4. The system should visualize graphs from sensor data

When the dashboard is set up and the application running, everything functions as expected and the data from InfluxDB is loaded in Grafana.

5. The system must perform air climate regulation

Air climate regulation has been tested by artificially producing conditions that exceeds the threshold values and confirm that the respective electronic device that handles the condition is started. Temperature max threshold has been tested by placing a finger over the sensor, and making sure the air fan starts. Temperature min threshold has been tested by using a can of compressed air on

the sensor, and making sure the air heater starts. Humidity max threshold has been tested by continuously breathing hot air on the sensor with my mouth, and making sure the air fan starts. Humidity min threshold has been tested by intentionally setting the threshold high at 50%, and making sure the ultrasonic mister starts.

6. The system must perform water quality regulation

Water quality regulation has been tested by using cold water below the threshold value in a separated water reservoir and making sure the heating pad starts. It has been further tested by lowering and increasing the ph exceeding the threshold values by using the ph + and - liquids and making sure the appropriate pump regulates the water back to within the threshold values. Nutrient regulation has been tested by using a water reservoir without nutrients added to make sure the pumps add an incremental amount of fertilizer to the water.

7. The system should be able to relay data queries

The API has been tested from a web browser and returns data as expected.

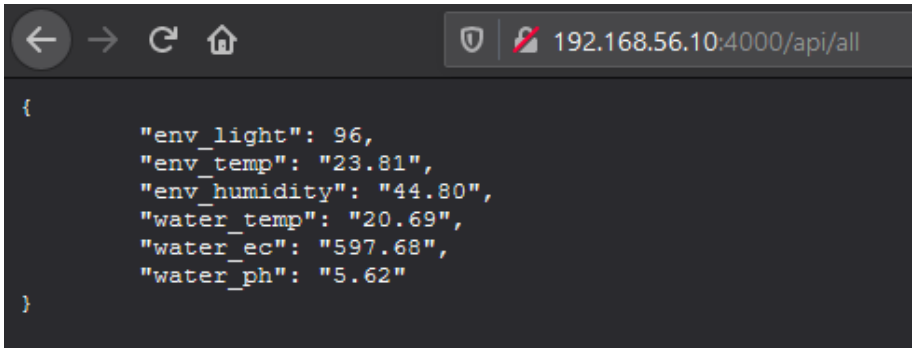

Figure 23: Testing - Data API

8. The system must send readable error messages

Readable error messages was confirmed when encountering problems with the SHT31D and DS18B20 sensor interaction, and also reconfirmed when intentionally disconnecting the Arduino from the Raspberry Pi.

5.8.2 Nonfunctional requirements

The system must be able to successfully grow plants

The system has been continuously tested by growing plants for two months prior to delivery, and thriving plants as shown in the final product showcase are a testament to its functionality.

The system must have a closed environment suited to perform regulatory actions

The enclosed grow box described in 'Design and prototyping' (p. 23) and displayed in the final product showcase functions as a satisfactory controllable environment.

The system must have configurable parameters for the desired environment

The application includes a configuration file where parameters for determining range of air temperature, relative humidity, water temperature, water ph and water ppm can be specified. Regulatory actions with electronic devices has been tested with success according to these threshold ranges. However, there is currently no automated way of adjusting water temperature and water ppm down.

```
const thresholdValues = {
  env_temp: {
    min: '20',
    max: '27',
  },
  env_humidity: {
    min: '40',
    max: '60',
  },
  water_temp: {
    min: '18',
    max: '', // No way to adjust water temperature down for now
  },
  water_ec: {
    min: '400',
    max: '', // No way to adjust EC down for now
  },
  water_ph: {
    min: '5.20',
    max: '6.80',
  },
}
```

Figure 24: Testing - Configurable environment parameters

The system must be able to run on a Raspberry Pi

The application runs successfully on a Raspberry Pi.

The system should automatically start when the Raspberry Pi is powered

The system service that was installed in 'Design and prototyping' (p.20-21) successfully starts the application when the Raspberry Pi is powered.

The system must only allow access to administrators

The Raspberry Pi has a dedicated custom user and password, and also requires a public key pair for network SSH connections as described in 'Design and prototyping' (p. 19-20). InfluxDB administration website and Grafana UI also requires a known username/password combination to access the data captured by the application.

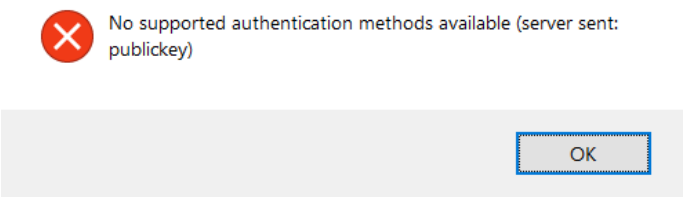

Figure 25: Testing - Administrator ssh access without the public key

The system should fit in a protected casing

A waterproof electronics casing was constructed and fitted as shown in 'Design and prototyping' (p. 24-25) and displayed in the final product showcase to make sure the components that don't need to be in the grow box are not left exposed to hazards.

5.9 User Interface testing

Spelling

The UI has been checked for spelling errors and appropriate unit displays (i.e, Celsius, percent and ppm).

Typography

The UI has been checked and user tested in regards to an easy to read font and colors that promotes readability.

Functions

The UI has been tested in regards to its functionality such as displaying data, maximum threshold value indication and that the graphs are the correct type and work as intended.

Behavior

In the upper right corner of the UI there are settings a user can change in regards to how long of a time period the line graphs will show data from, and how often the UI should refresh to show new values. The user can also highlight a time period on the graphs themselves by clicking and dragging the mouse over the desired time period before releasing the mouse button. This will zoom in on the selected interval and show more detailed information. These behaviors work as intended and are easy to use. The UI was tested with all major browsers - Firefox, Chrome and Edge without deviations to the results.

5.9.1 User testing

User feedback on the UI was collected from the same individuals that featured in the 'User interviews' (p. 7-8). Respondents were asked about typography, ease of use and behavior of the UI, and I in fact struggled to get critical responses even when pressing on what could be improved. The most negative feedback I could obtain was that one of the respondents did not like the dark theme, which is highly subjective - but also possible to change in the Preferences menu of the app. Users found the layout ordered in a logical manner, the typography easy on the eyes and the behaviors of the UI intuitive to use and understand.

6 Final product

Building upon the successful testing of all the components in the product, this is a showcase for the final assembly of the production system. The major parts that constitutes the final product are the enclosed grow box and deep water culture hydroponic system, the protected electronic box and associated hardware, the application and the user interface.

6.1 Grow box

The complete grow box consists of a LED light optimized for growth with wavelength at 660nm (red) and 450nm (blue), 4 pumps for ph and nutrient level regulation, an air fan, a coupe warmer, a heating plate and a ultrasonic mister placed in a mason jar with water. The sensory equipment for reading data parameters are a KY-018 photoresistor module for light, a SHT31D for air temperature and relative humidity, a DS18B20 waterproof temperature sensor for water temperature, a Keystudio KS0429 TDS meter for water nutrient levels and a DFRobot ph meter.

Figure 26: Final product - Grow box completed assembly with bell pepper plants

There is also a deep water culture hydroponic system to suspend the plants with an air stone connected to a air pump resting in the nutrient solution to keep it oxygenated. The grow box itself was made from an 80x60x40cm plywood box and a plexi glass sheet to enclose it, leaving a height of 10cm for ventilation.

Figure 27: Final product - Nutrient and ph regulation pumps in the grow box

The pumps and the heating mat under the water reservoir are responsible for regulating the water environment with nutrients, ph adjustment liquids and heat if necessary. The coupe warmer, air fan and ultrasonic mister are responsible for regulating the air environment for appropriate temperature and humidity.

6.2 Electronics box

The contents in the protected electronics box are electronics device outlets, a 230/12V voltage transformer, an Arduino and breadboard, a relay module, a Raspberry Pi and finally terminal blocks to connect power sources to the relays. The electronics box is made out of steel and the cable insertion hole is encircled by a piece of rubber to shield the contents from water, should a water leak occur as mentioned in our risk analysis (p. 10). It also function as a single point for cable connection to minimize risk of tripping in wires and other inadvertent damage to the electronic system.

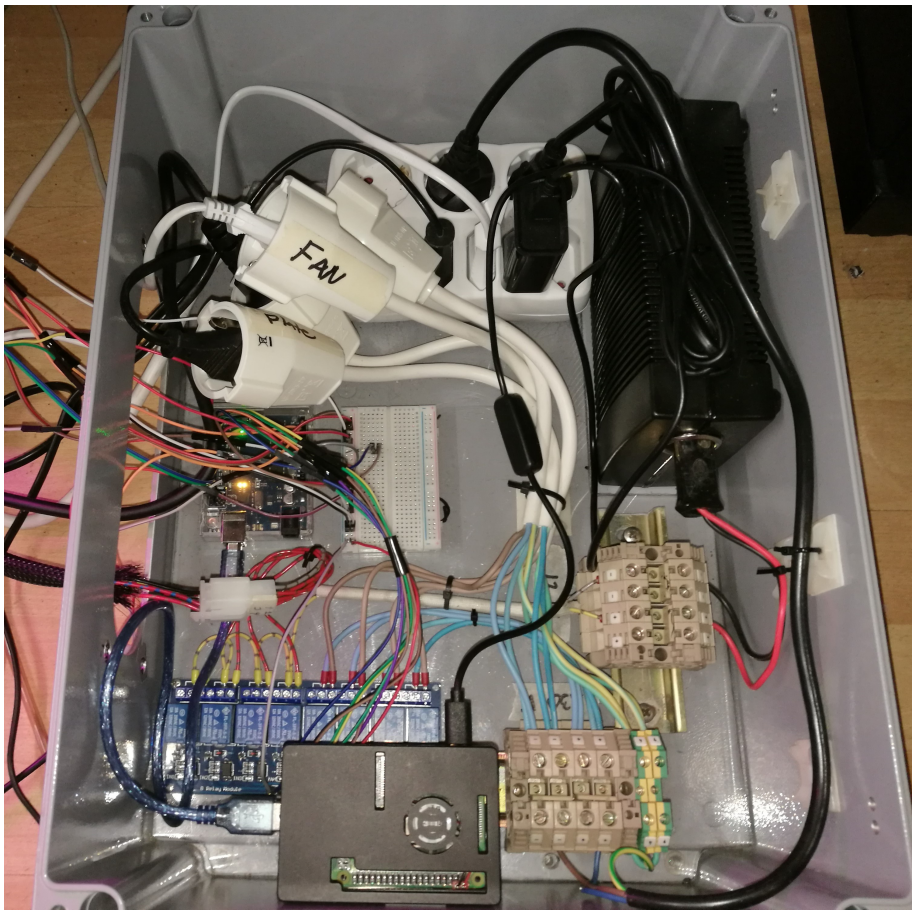

Figure 28: Final product - Contents of electronics box

The electronics box is the heart of the system, and this is where all the electronics are given power, and also where the sensory equipment in the grow box sends data to the microcontroller that in turn signals the relays to turn on the regulatory devices when an action is needed based on the aforementioned data. In addition this is where our data center, the Raspberry Pi is located and runs the application that control automation instructions, and the necessary software for saving and visualizing the data collected.

6.3 Application

The application is where the instructions of automating the grow box, saving the data and serving the web API are stored. All the sensory equipment and relays are defined in the application that in turn connect to the Arduino microcontroller and use the data from the sensors to control the relays based on threshold values in a configuration file. The application also saves the sensor data to a database if the feature is enabled in the configuration file. Lastly it serves a web API that the user can query for real time values of the sensors, and historical data from the database. When a regulatory action is performed in the grow box, the application will light up a LED diode to indicate that an action is ongoing.

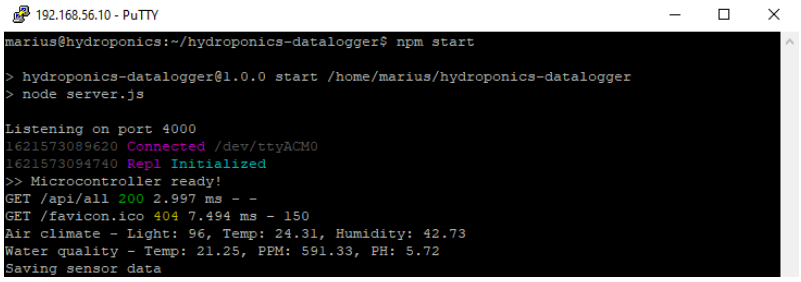

Figure 29: Final product - Application running on Raspberry Pi

The application is hosted on the Raspberry Pi along with InfluxDB which is the database software that stores the data, and Grafana which is the visualization software that acts as the interface to display said data to the user.

6.4 User interface

The user interface is where the user is presented the data saved to InfluxDB in a visually appealing format that is eye-catching and intuitive. The visualization dashboard has three sections, one at the top for the latest values captured in a gauge visualization color coded by green for a value within the defined threshold range, and red for a value outside of the defined threshold range. The other two sections are grouped into 'environment' and 'water' with line graphs that display historical data for a given time interval that is changeable by the user.

7 Process

The project benefited from following established development models and tools to keep organized and distribute the code between development and production environments.

7.1 Development model

7.1.1 KANBAN

KANBAN is a lean development method used to manage and visualize workloads within a development cycle. This fits perfect for this project, as I am a solo member executing all the work, which makes a full scale agile methodology unfit for the job, and also helps to minimize the time spent planning the project while providing a good measure for the work needed to be done. Cards that visualize workloads are effective and helps organize the work into separated categories.

7.2 Tools

7.2.1 Trello

Trello is a KANBAN style card application that is used to organize the project into different tasks across multiple categories. This has assisted in breaking down more complicated processes into individual tasks, managing the projects milestone schedule and keeping track of which tasks needs to be done as well as what has been done.

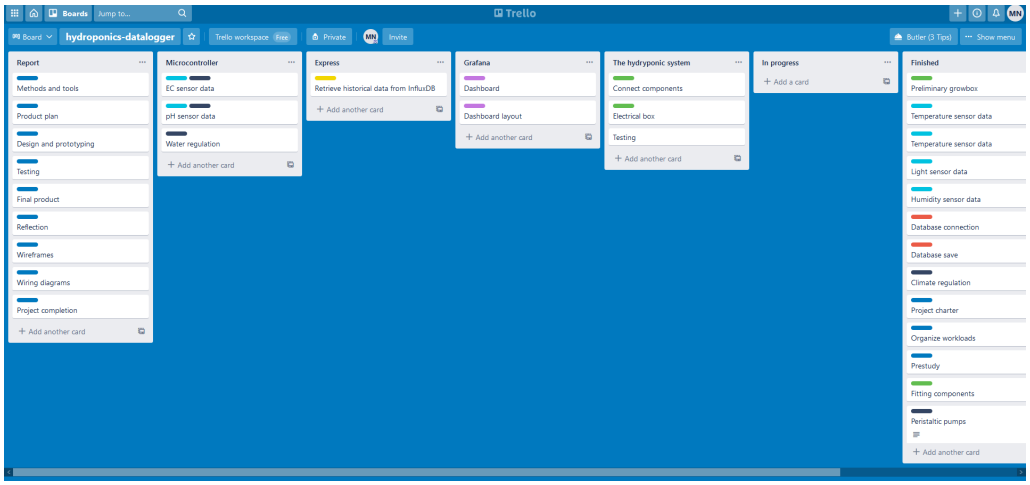

Figure 30: Trello cards

7.2.2 Github

Github is used as a central storage for code and to keep track of changes in the code. It's a tool that allows for synchronizing of code between teams and to make isolated changes in a branch before committing code to a central repository. Doing this project solo means that Github has been limited to storage and distribution for this project.

8 Reflection

During this project I have gained positive acumen into project planning, hydroponic systems, Javascript development, Internet of Things devices, electronics and system integration. It has been both demanding and satisfying with a fair amount of challenges and personal growth, especially in regards to the system integration from a Windows 10 development machine to the Raspberry Pi that caused me more than a few headaches as described in the testing section.

Developing with Node.js and a time series database was a fun experience, and has given me a greater understanding of asynchronous operations and some of its advantages. I am not completely satisfied as of yet with the scope of some aspects of the functionality and would have enjoyed to do more coding in order to achieve further advanced implementations. On the other hand, creating a working system with several components interacting successfully is a greater task than just coding. In the end the minimum viable product requirements were met and somewhat exceeded, and I will deliver this thesis knowing full well that I have done my best and worked as hard as I possibly can to achieve success.

8.1 Suggestions for further development

- Redundancy on critical components - Air pump

- Light program
- Automatic refilling of water reservoir
- Further develop handling of nutrients
- Stabilize analog sensor readings
- Manufacture a protective sensory equipment housing
- Higher accuracy for electrical conductivity sensor
- CO2 sensor for air and barometer
- Distance measuring sensor for each plant to allow for data analysis on the crops
- Further development of the data API

Redundancy on critical component should be implemented in a production environment. An air pump failure will ultimately lead to dead plants as oxygenated water is absolutely essential to hydroponic systems, and in a system such as this where the water is still, the air pump is the only generator of oxygen.

A light control program could be beneficial as not all plants require the same conditions to produce flowers as stated in the prestudy. This would also require a bigger relay module to control more electronic devices.

Automatic refilling of the water reservoir from a bigger container would have added more time for the grower to let the system run unmanaged, as manually refilling the water reservoir is currently one of the things that are absolutely essential for this system in its current state. One possible way to implement this function is to use a float sensor that acts like a switch, and either have it trigger another pump or a gravity assisted hose with a one way valve from a second water reservoir. However, I am somewhat satisfied that the automation processes can keep this system up unmanaged for more than a week without user interaction. Limitations of space has prohibited this implementation in this development iteration.

Plants require different nutrient combinations based on what stage they are in. There are two stages, a growing stage where the plant optimally has more nitrogen available in the nutrients, and a flowering stage where the plant optimally has more phosphorous available in the nutrients. This can be implemented with a button switch that rotates the nutrient program and separate configuration parameters in the application. Plants in the flowering stage also needs more nutrients as it is bigger, and spends more energy in producing fruits. Time constraints has limited this project from realizing such a implementation.

The analog water sensors (i.e, ph and electrical conductivity) are prone to minor fluctuations in the readings. One possible way to stabilize said readings could be to gather readings over time in an array and output the average of all values. This has not been given a high priority for the minimum viable product because the impact is minimal, and has been sacrificed for other functionality in this development iteration.

Sensory equipment are electrical devices, and moisture in particular can be devastating to the equipment. To compensate for the lack of a protective housing, the mister is set to only run for fifteen seconds at a time. This should make sure that not too much moisture is generated in a short amount of time and overwhelm the sensors.

The electrical conductivity sensor is limited to readings at maximum 1000 PPM. This is sufficient for most herbs and plants, but there are exceptions such as tomatoes - that require a high nutrient value for optimal growth. As mentioned in the prestudy sub-section 2.4 there are more advanced sensors available from manufacturers DFRobot and Atlas Scientific, but both were outside of the budget scope for this project and as a result the quality of this sensor has been limited. If budget had not been a factor, the instruments from Atlas Scientific seems to be the most promising with the ability to stay submerged without re-calibration for a year according to their data sheets.

The implementation of a CO2 sensor could be beneficial to commercial farmers, but cost/return research will have to be conducted over several full growth cycles for comparison. However, this is very uncommon for a recreational grower to utilize as there are usually other limiting factors in the plants growth without using expensive equipment, predominantly the amount of light available for the plants.

An IR sensor placed at a predetermined distance above the base of the plants could provide useful insights and data analysis in regards to growth cycles and yield accumulation for commercial growers, but would probably be outside the scope of what a recreational considers useful.

I regret not having more time allotted to implement extended functions, especially a more advanced nutrient delivery system and automatic water reservoir filling. Because the time constraints have been the most critical factor in this work, unfortunately it has not been possible. On the brighter side of things, I have demonstrated the functionality needed and doing further development to add this shouldn't be too technical or time consuming.

8.2 Educational gains

The educational gains from planning, prototyping and writing about this project has been substantial and pushed my learning by exploring a new coding language and utilizing the skills I have learned about project planning and development.

8.2.1 Project planning

Conducting a prestudy and writing a detailed product plan has developed my project planning skills to the next level, and has helped me realize how powerful of a tool it is to have a elaborate plan to base ones work on in a project. System requirements and user stories helped make the programming more approachable and organized, and easier to see the full scope of operations that had to be implemented. I often find it difficult to write a detailed plan without doing some experimenting to explore the potential of ideas possible, but this project has left me with a valuable lesson that a good project charter and product plan is a key factor for success in projects.

8.2.2 Hydroponic systems

A hydroponic system in itself is a great source of automation that removes mundane tasks from the growers point of view, and to be able to further automate the processes such as nutrient delivery, ph adjustment and climate control has been a true joy. From the user interviews I gained valuable insight that for the most part recreational gardeners want to automate as much as possible, without deep diving into the details of every single aspect of gardening, and that a system with as much automated regulation as possible is beneficial to them.

During the project I have cultivated bell peppers from seeds to late in the growing stage, and to see the results of my labor has been a motivating force throughout the project.

8.2.3 Serving data from an API

Making an API that fetches real time data from the sensor equipment and historical data from the database has been a new and interesting experience. The underlying idea is that the API can be used for monitoring and fill gaps in regards to the data collection that is only saved at fifteen minute intervals, and in the future work in conjunction with other visualization techniques than Grafana. It needs more work for a wholesome implementation but produces working results at a rudimentary level for now. Further enquiries into user needs for visualization and data anlysis is needed for a more extensive understanding on how to develop this.

8.2.4 Visualization dashboards

The initial idea for a visualization dashboard was to use a web client made by Vue.js and a graph library. However time constraints led to this idea being rejected during the development process, as it was too time consuming to get a working solution with adequate quality.

Grafana provided an excellent substitution, and in hindsight I can safely say it was the correct choice. Its a *much* more developed application than I could ever hope to develop in this time frame, and has great support for different databases and some great utilities such as threshold values that integrate well with this project with a final result with pleasing aesthetics.

8.2.5 Electronics box

Making a protective electronics box for the cables, relay module, microcontroller and Raspberry Pi has given me an introduction into how electronic devices are connected, and how one might power such devices in a timed fashion using relays. It has been a fun job within the project, although somewhat time consuming to set up correctly and in a way that separates different voltage powers.

It has also given a involuntary practical example of how important it is to handle potential dangerous voltage with care, as I got electrically shocked from the 230V terminal block when I forgot to disconnect the extension cord module. Don't be like me, remember to disconnect things properly, it is not a pleasant experience!

References

Fazaeli, H et al. (2012). 'Productivity and nutritive value of barley green fodder yield in hydroponic system'. In: *World Applied Sciences Journal* 16.4, pp. 531–539.

9 Attachments

9.1 Data sheets

SHT31D Datasheet - http://www.jysource.com/issi/gy-sht31-d-sht31-sensor-module.html

DS18B20 Datasheet - https://media.digikey.com/pdf/Data%20Sheets/DFRobot%20PDFs/DFR0198_Web.pdf

DFRobot PH electrode datasheet - https://image.dfrobot.com/image/data/SEN0161/PH%20composite%20electrode%20manual.pdf

Keystudio TDS meter datasheet - https://fs.keyestudio.com/KS0429

9.2 Code

The code presented in this project is developed and tested on Windows 10 with Node.js version 14.15.5, InfluxDB version 2.0.6 and Grafana version 7.5.5 before migrating to the Raspberry Pi.

Node.js version 14.15.5 - https://nodejs.org/download/release/v14.15.5/node-v14.15.5-x64.msi

Grafana version 7.5.5 - https://dl.grafana.com/oss/release/grafana-7.5.5.windows-amd64.msi

InfluxDB version 2.0.6 - https://dl.influxdata.com/influxdb/releases/influxdb2-2.0.6-windows-amd64.zip

Source code - https://github.com/mariusnorheim/hydroponics-datalogger

9.3 README

hydroponics-datalogger

Automated hydroponic system using Arduino Uno, ConfigurableFirmata, Johnny-Five, Express, InfluxDB and Grafana

Arduino components

- LED module
- Photoresistor module
- SHT31D temperature/humidity sensor
- DS18B20 water temperature sensor
- DFRobot PH sensor
- Keystudio TDS sensor
- 8 channel relay module

Electrical devices

- Coupe warmer
- Fan cooler
- Heating plate
- Mister
- 4x 5W peristaltic pumps
- 230/12 voltage transformer

TODO

I plan on adding a more advanced nutrient delivery system for different growing stages of the plants at a later point

Prerequisites

ConfigurableFirmata - https://github.com/firmata/ConfigurableFirmata

Node.js version 14.15.5 - https://nodejs.org/download/release/v14.15.5/node-v14.15.5-x64.msi

InfluxDB version 2.0.6 - https://dl.influxdata.com/influxdb/releases/influxdb2-2.0.6-windows-amd64.zip

Grafana version 7.5.5 - https://dl.grafana.com/oss/release/grafana-7.5.5.windows-amd64.msi

InfluxDB data logging and Grafana visualization can be disabled in the app's config.js file.

Setup

ConfigurableFirmata

Unzip ConfigurableFirmata to Arduino library folder, typically '/Documents/Arduino/libraries/' on Mac or Linux or '\My Documents\Arduino\libraries\' on Windows and load the Arduino IDE. Load the sketch from 'File - Examples - ConfigurableFirmata - ConfigurableFirmata' and click upload while your Arduino is plugged into your computer.

Node.js

Download the Node.js install file and follow the on-screen instructions for installing.

InfluxDB

Requires wget and PowerShell to install.

wget https://dl.influxdata.com/influxdb/releases/influxdb2-2.0.6-windows-amd64.zip -UseBasicParsing -OutFile influxdb2-2.0.6-windows-amd64.zip Expand-Archive .\influxdb2-2.0.6-windows-amd64.zip -DestinationPath 'C:\Program Files\InfluxData\influxdb2\'

Grafana

Download the Grafana install file and follow the on-screen instructions for installing. Default username/password is admin/admin. Import dashboard.json from 'Dashboards - Manage' within Grafana.

Documentation

Warning

Connecting the relay module to a power source will automatically power on all relays. Keep this in mind!

Starting the app will power the relay module for a few ms, connecting power to the peristaltic pumps should be avoided until the app is running. Unfortunately I haven't found a working solution to avoid these issues.

Install dependencies

npm install

Start app

npm start