



HYGROW AQUAPONIC SYSTEM:

CREATE A FISH CLASSIFIER



MARINE BONNAC
PABLO CRIADO ALBILLOS

BENJAMINE KPODAR

BAS HUPJÉ

MAXIME MAIROT

TOBIAS EKFORS

EPS SPRING 2022

1 Team members

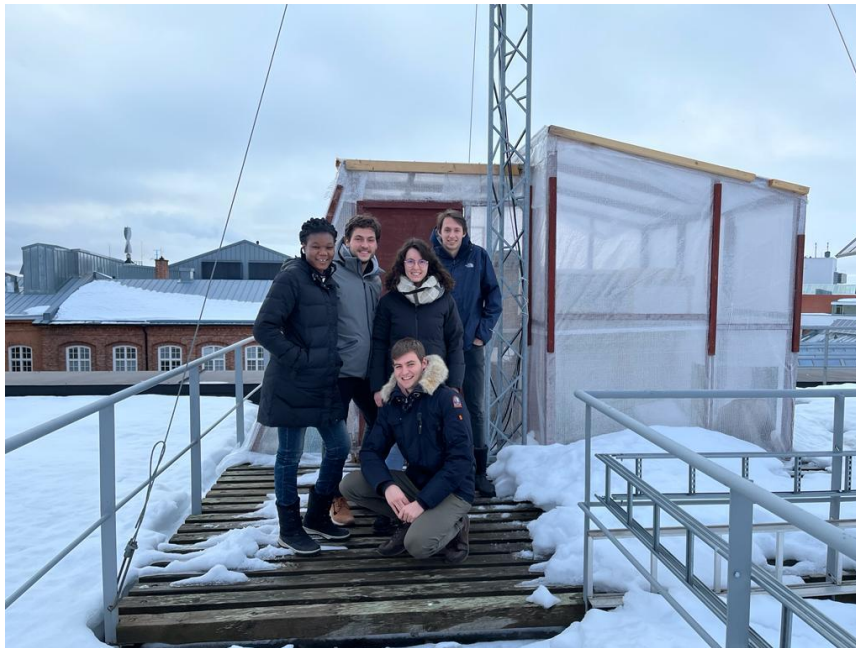


Figure 1. Hygrow Aquaponic Team - Spring 2022

Bas HUPJÉ:

I'm Bas Hupjé, and I come from The Netherlands. I am 22 years old. live in a small town near to the city Enschede. The town I live in is situated on the German border and therefore I also do understand a little German. I study Chemical engineering at Saxion in Enschede. I like the study for now because for me it's a nice combination of practical and theoretical work. I saw the European project Semester as the perfect way to get some experience in living in a foreign country.

The Aquaponics project was for me the best option because it is the most practical one. I like it when you can see the progress easily which is easy for this project. For the rest I like to be busy with sustainable technologies, which is also the case in this project.

Contact: 439064@student.saxion.nl and +31 6 185 886 84

Benjamine KPODAR:

Kokoe Benjamine kpodar is my name. I come from TOGO my country of birth. I live in the capital city Lome which is in the south part of the country. I study management at university of Lome, and I am in my 24 years. I choose EPS program because is more practical and have great concept, it is a good opportunity for me to work on team, gain international experiences and improve my language skill in English. Travel to Finland open me the door to another culture and different way of living.

I find team working very interesting, my current goal is to get maximum profit in teamwork experience. I choose aquaponic project because it is closer to the natural environment think with this project the world will be able to produce healthy and natural food for each house.

Contact: kpodarbenjamine@gmail.com and +22893208297

Marine BONNAC:

My name is Marine BONNAC. I am 21 years old, and I come from the south of France. I live in a town near Toulouse in the South-West of France. I am a student in Tarbes (also in the South of France) where I am in the National Engineering School of Tarbes (ENIT). It is a generalist school oriented in Mechanical and Industrial Engineering. Personally, I chose to go into Mechanical Engineering where I have been taking courses in this speciality for one year.

I chose this project because I particularly liked the proposed subject. Indeed, the concept proposed to create a fish classifier is interesting. I also like the idea of creating what we can call "The House of the Future" with a life cycle where the fish feed on the vegetation available in the Greenhouse. Moreover, this project calls for several skills and knowledge from different fields. I find it interesting and enriching to be able to work in a group to share our ideas and thus respond to the problem that has been proposed to us.

Contact: marine.bonnac@enit.fr and +33 7 81 90 67 90

Maxime MAIROT:

I'm Maxime MAIROT, and I come from France. I live and study in the South-West of France at ENIT (Ecole Nationale d'Ingénieurs de Tarbes). In this school, I'm studying mechanical and industrial engineering. I'm 21 years old and this European Project Semester is my first occasion to study in a foreign country. I wanted to work collaboratively and in English on an interesting topic in which I can be active. I also wanted a relatively long topic to have some reflection and hindsight on things to achieve and of course to see a significant result at the end. That's why I chose an EPS to study during in one semester. I wanted to improve my English too and discover a Nordic country, so Finland and Vaasa are the best place to be and spend good time.

To finish my presentation, I chose this project named "Hygrow Aquaponics Project" because I am really close to nature and the environment. What really interested me was the fact that this Greenhouse could perhaps be marketed later, and it would be carbon dioxide free. Moreover, as a mechanical student, I am attracted by the rethinking that leads to the construction of something, here the fish classifier.

Contact: maxime.mairot@enit.fr and +33 7 82 16 34 20

Pablo CRIADO ALBILLOS:

Hej! I'm Pablo Criado Albillos, I come from a mid-sized city in Spain called Valladolid. I study Industrial Electronics and Automation Engineering -usually just electronics for abbreviation-. I'm 21 years old and this is my fourth and last year of my study plan. I chose to do my final degree project in the form of an EPS because that's a unique opportunity to improve my skills in several fields, not just engineering but also English, cross-cultural communication and public speaking, which are often not really covered in my home university. Also, the idea of working on a bigger project, and one that actually gives the opportunity to build something really drew my attention.

Living in Finland is an open window to nature, which I love, and to a certainly different way of living. Most specifically, I wanted to experience an actually-cold winter -colder than in Valladolid for sure- and learn how society here tackles the cold, snow and changing sun times.

The Hygrow Aquaponics project is interesting in many ways. It has plenty of electronics and automation, which I love, but also biology and mechanics, which aren't that familiar to me, thus it's a nice project for which I can be useful, and from which I can learn quite a lot.

Contact: pablo.criado@edu.novia.fi

2 Preface

This final report presents the Hygrow Aquaponics project done by five international students of Novia Yrkeshögskolan in the context of an European Project Semester (EPS). This program was created mainly for students from the engineering sector, but students from business schools and management schools can also participate. It is possible to do an EPS during your school career but only at 18 different schools in Europa, and Novia University of Applied Sciences is one of them. EPS program was designed to prepare students for the economy of tomorrow and all that it entails. The projects aim to develop the multidisciplinary of each individual. At Novia, the subjects are mainly focused on energy, automation and sustainability. In each EPS project, there are between 2 and 10 students who work together every day to achieve the set objectives. The projects help to develop engineering and management skills as there are parallel courses help deepen their knowledge of project management.

Nowadays, the production of food often has a big impact on the planet: using a lot of water to produce and often a big carbon footprint to import/export the food around the world. Moreover, today the climate change is at the heart of many issues. Therefore, it's important to reduce the carbon food print. The main objective of this project called "Hygrow Aquaponic Greenhouse" is to produce plants/vegetables by minimizing this carbon footprint, if not totally eliminating these emissions.

When we arrived, the Hygrow Aquaponics project had started two years earlier. We already had something to build on. The Greenhouse was functional but then the covid came along, so it was left a bit abandoned due to various lockdowns or teleworking. Our project was to be a continuation of what had already been created and achieved. In order to be able to carry out the continuation in good conditions, we need to understand what has been done to be able to start in the right direction.

Hygrow Aquaponics team wants to thank different people and parties more or less connected to our project. First of all, we would like to thank Novia for hosting us in its premises for a semester and for giving us access to several laboratories and tools (3D printers, construction materials). Furthermore, we would like to thank our supervisor Tobias Ekfors for his help and advice during the weekly meetings. We want also to thanks the previous group who have already worked on the Greenhouse's project. To finish, we thank the EPS coordinator Roger Nylund, without whom it would have been impossible to create a strong and cohesive team. Thanks also to Hans Liden who answered our technical questions and to our customer Mikael Ehrs.

Vaasa, May 2022

Pablo CRIADO ALBILLOS

Maxime MAIROT

Bas HUPJE

Marine BONNAC

Benjamine KPODAR

3 Abstract

The main objective of this semester's "Hygrow Aquaponic" project is to create and design a fish classifier and to be able to put it into use in the existing greenhouse. In fact, the aim of this major project is to be able to feed the fish with the plants present in the Greenhouse. These fish will then be sorted by size using the fish classifier and then the larger ones can be sold to professionals or individuals to eat.

Initially, the project will be developed in a simple aquarium and then implemented in the Greenhouse if possible. It is a complex project that requires a lot of knowledge, which is why we are a group of five foreign students working on it.

To carry out this project, we first researched which fish could be used to carry out our fish classifier tests in the aquarium. This was the first part of our project. Then we made drafts to determine the final system that we will develop to obtain a functional fish classifier. To finish with this part of the project, we also put the vision machine into operation so that we can take pictures of the fish and measure them to be able to sort them.

Secondly, we are working on an already existing system as the greenhouse is present on the roof of the Technobotnia building. In this Greenhouse, we have made improvements by adding a second tank for the plants and we have also modified the website related to the Greenhouse. Indeed, we have (in parallel with the creation of the fish classifier) adjusted and modified the automation and control system of the Greenhouse. In addition to this, we also worked on the insulation of this Greenhouse and modified the entrance door.

All of this work and experimentation allowed us to implement our ideas and concepts to meet our client's requirements. During the project, we had to modify some parameters of the fish classifier to arrive at our current design. We can therefore say that the project was carried out using a step-by-step approach.

Table of contents

1	Team members	2
2	Preface	5
3	Abstract	6
4	Introduction	10
4.1	Stakeholders	10
4.1.1	Novia University of Applied Sciences.....	10
4.1.2	Mikael Ehre.....	10
4.1.3	The local restaurants.....	10
4.2	Mission, vision, and goals	11
4.2.1	Mission	11
4.2.2	Vision.....	11
4.2.3	Goals.....	11
5	Fish classifier	12
5.1	Concepts.....	12
5.1.1	Solution 1	13
5.1.2	Solution 2	14
5.1.3	Solution 3	14
5.1.4	Solution 4	14
5.2	Fish selection.....	15
5.3	Construction.....	15
5.3.1	Aquarium.....	15
5.3.2	First ideas for the design.....	17
5.3.3	Gate to separate the fishes.....	20
5.3.4	3D printed pieces for the machine vision	23
5.4	Support for the final structure	28
5.5	Machine vision	32
5.5.1	Camera lens.....	32
5.5.2	Lighting.....	33
5.5.3	Processing software	35
5.5.4	Capturing images	35
5.5.5	Processing images	39

5.6	Conclusion, advice and problems	46
6	Automation with Home Assistant	48
6.1	Editing configuration.....	49
6.2	Interface theme	50
6.3	Heating system.....	51
6.4	Lights	52
6.5	ESPHome	55
6.6	Water level sensor	56
7	Monitoring	58
7.1	Configuring Home Assistant.....	58
7.1.1	Encryption - HTTPS.....	58
7.1.2	The problem with incoming connections	59
7.1.3	Remote monitoring using IPv6.....	59
7.2	Remote server for web and proxy	61
7.2.1	Novia DNS	61
7.2.2	Encryption – HTTPS certificate.....	62
7.2.3	Nginx web server.....	62
7.3	GitHub	66
7.4	Monitor	67
7.5	Telegram bot.....	70
7.6	PM2	71
8	Greenhouse.....	72
8.1	Plans of the Greenhouse.....	72
8.2	Insulation of the Greenhouse	74
8.3	Heating experiment	75
8.4	Door construction	77
8.5	Testing the growth of plants.....	79
8.5.1	Electricity consumption	81
9	Business model	82
9.1	Theoretical research on the aquaponic environment	82
9.1.1	Fresh-water mussels	82
9.1.2	Macrobrachium rosenbergii	84

9.1.3	Freshwater shrimp	87
9.1.4	Freshwater crabs.....	88
9.1.5	Langouste.....	90
9.1.6	Larges species	91
9.2	Theoretical research on the underwater plants	92
9.2.1	Guppy grass.....	92
9.2.2	Green foxtail.....	93
9.2.3	Java Moss	93
9.2.4	Hornwort.....	94
9.2.5	Anacharis.....	94
9.2.6	Salvinia natans	95
9.2.7	Red root floater.....	95
9.2.8	Duckweed.....	96
9.2.9	Proportion between underwater plants and guppy fish	96
9.2.10	Commercial value for underwater plants	96
9.2.11	Humans benefit.....	97
9.2.12	Fish benefit.....	97
9.3	Cheapest/robust webcams for cold and damp environments	97
10	General research on aquaponics plants.....	101
11	Budget	102
12	Conclusion.....	103
12.1	The remote monitoring system	103
12.2	The fish classifier	103
12.3	The greenhouse.....	104
13	Recommendations	105
13.1	The remote controlling system	105
13.2	The fish classifier	105
13.3	The greenhouse.....	105
14	Table of figures	106
15	Bibliography	110

4 Introduction

In this part the project and the stakeholders will be introduced. The project will be executed by the team presented above together with coaching of teachers earlier mentioned from Novia. There were also other project groups working on the Aquaponics project and this report will be a follow-up from the previous ones. During the 2022 spring semester, this team is aiming to deliver a working fish classifier for the greenhouse, a remote monitoring system for the greenhouse and some adjustments to the greenhouse itself. This classifier has to make the difference between big fishes and the small fishes so that way it is easier to harvest. The monitoring system has to be improved further so everything can be put on and off remotely. Also adding new sensors and fixing the already existing sensors will be a part of this subject. At last, the adjustments to the greenhouse are about repairing things in the greenhouse and improving the isolations where possible.

4.1 Stakeholders

In this part the different stakeholders are introduced with the reason they are important for the project.

4.1.1 *Novia University of Applied Sciences*

According to the information written at novias webpage, Novia has about 4500 students and 300-man personnel and 5 campuses. (Novia, sd) Novia is also situated along the Finnish coast and operates in Vaasa, Turku, Raasepori and Pietarsaari. On the same page Novia claims to be the Swedish-speaking polytechnic in Finland. (Novia, sd) It also has a close working relationship with the business community. Further is Novia also responsive to the needs of working life and wants to lay the foundations for the sustainable development of society. (Novia, sd)

Novia is obviously important for this project since they are the providers of the EPS semester. Also, the greenhouse is situated at the roof of Novia and uses electricity of Novia. Novia also provides all the materials and the working spaces. In general, Novia has a really big part in the project.

4.1.2 *Mikael Ehls*

Mikael Ehls is the customer for this project and is this way an important stakeholder in this project. Mikael will provide the team with useful information and help the project getting to his wishes. He also knows already a lot of background information which can come in handy during the project. Keeping the customer up to date will be one of the most important things, because this way the team and Mikael know both where the project is and where it needs to go.

4.1.3 *The local restaurants*

In the future the local restaurants can be seen as stakeholder. This is because if the greenhouse is starting to produce food, they can start to buy it. For them it is important the food the greenhouse

produces is from good quality and also popular food. It can also be good marketing to tell the food is produced in local greenhouses.

4.2 Mission, vision, and goals

The goal of this paragraph is to explain the mission, vision and goals of the project. The mission and vision are both a long-term orientation while the goal is something to achieve during this project.

4.2.1 Mission

“To make a remote-controlled greenhouse which produces popular food and contributes to the sustainable society.”

This mission is not something we are finishing during this semester, but it is an achievable goal if you look at the long-term. The most important is to make sure the team delivers a good contribution to the overall project and hands it in a good way over to the next group. The concept of a greenhouse is a good one, because it is a sustainable way to produce food. This greenhouse can be used to experiment and maybe implement on larger scale in the future.

4.2.2 Vision

“Producing food with a positive contribution to climate change.”

Climate, geography, and prosperity have influence on the production of food. An Aquaponic Greenhouse takes this away and gives every household the opportunity to grow food. That is what Hygrow Aquaponic Greenhouse contributes to.

Climate change is a big issue all over the planet and therefore every step to lower the CO2 footprint is a good one. The Aquaponics concept can be seen as a good alternative for some food production. If the Aquaponics concept is used on a larger scale, it can make a big impact on climate change.

4.2.3 Goals

The goal of the project is to develop a working fish classifier and improve the isolation and monitoring system of the greenhouse.

5 Fish classifier

The main task of the project is to create a fish classifier. This is the most important expectation of the customer. For this, we have to carry out several important steps for the realization of the latter. Indeed, we have to improve the aquarium at our disposal in order to create a more rigid structure that will allow us to maintain the aquarium in case of problems (cracks, etc...). Then, we have to imagine several solutions for the development of the fish classifier. Finally, we were interested in the fish that we will be able to use during the semester in order to test and create the expected fish classifier. To carry out this task, we need relatively small fish for testing in the aquarium. If these tests are successful, we will implement the system in the Greenhouse. This will be the last step of our project. For this last step, we need to have larger fish than those used in the aquarium.

5.1 Concepts

As mentioned in the introduction and in the goals of the project, one of the main objectives of the Hygrow Aquaponics team is to build a fish classifier. The long-term goal is to implement it in the existing Greenhouse. If we have time at the end of the project, we will try to add it to the Greenhouse ourselves, but this will depend on the progress of the fish classifier construction.

- Goal of the fish classifier: The objective of this fish classifier is to separate different fish according to their size. Separating fish can be useful for feeding them. The system must be fully automatic.
- Materials available: For this fish classifier we have an aquarium at our disposal. Then we also have access to several laboratories to build what we want: wood laboratory, metallurgy laboratory. Moreover, we also have access to 3D printers, which we can use after a quick training in the field. We also have available what previous groups bought but did not use. We can use them, but we have to see how we can use them first because it was not our team that decided to buy these components. Finally, we also have a camera and Raspberry's available for size identification of fish.

One of the big problems with the project is that we don't have a defined budget, so it's difficult to project. So, we have to think about a construction with what we can find within the University of Novia and in Technobotnia.

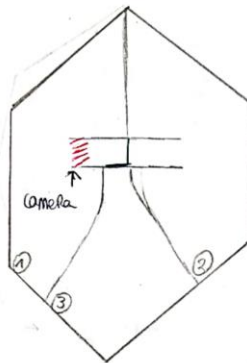
- Drafts & Concepts: Before starting to build anything, there is always a reflection phase in a project of this magnitude. We therefore held a meeting together to define several potential solutions. After comparing everyone's ideas with what is possible, we came up with 4 overall solutions which we presented to Tobias. Here are the first 4 solutions we came up with:

Fish Classifier

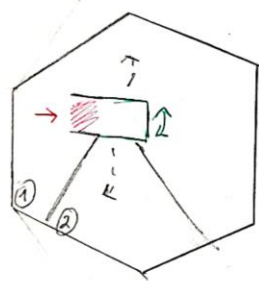
! : door

≡ : space where we take picture.

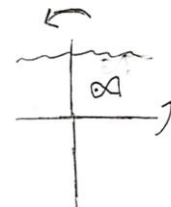
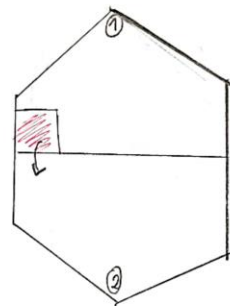
• 1:



• 2:



• 3:



• 4:

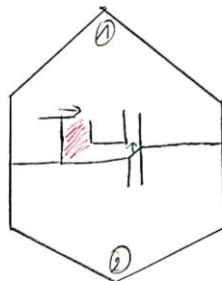


Figure 2. Fish classifier drafts

We will try to detail each of the solutions by stating the different problems related to them.

5.1.1 Solution 1

In this solution, we decided to create 3 different parts within the aquarium: in part 1, we will put all the fish in at the beginning of the study. Part 2 will be the part of the aquarium where we will collect the "big fish" that are above the threshold size. Part 3 will be the part for the smaller fish. The fish will move towards the space where the camera will be, the camera will take a picture. After studying the

picture (we don't know yet how we will treat this picture), the door in green will open either for space 2 if the fish is considered big enough, otherwise the door for space 3 will open.

One of the problems with this solution is that the fish in space 3 may have to go the opposite way and return to the original space when the door opens. We have already thought of other solutions to avoid this, which we will detail later.

5.1.2 Solution 2

This solution is close to the first one, but here we have only 2 rooms. The sorting system is the same as in the previous solution, but if the fish is too small it stays in the same space as in the previous solution.

One of the main issues is that on fish considered like a “small fish” can go as many times as it wants.

5.1.3 Solution 3

For this solution, we thought of another method to move a fish from one place to another. A sort of roulette wheel will move a fish from one part of the aquarium to another. The camera will detect a fish, take a picture and then analyse the size of the fish. If it is big enough, the wheel will start turning to transfer the fish.

The problem with this method is that the system can transfer several fish at the same time and the fish will be out of the water for a moment, so it may fall out of the aquarium.

5.1.4 Solution 4

This last solution is a bit more complex as we would need 2 actuators to implement it. The fish arrives in a small space where the camera will be placed. A door detects the arrival of the fish which will cause the door to close (in black). After analysing the size of the fish, it will either go to space 1 dedicated to large fish, or it will return to the initial space.

The problems associated with this solution are the same as for solution 2. We can also say that adding a sensor and an actuator for the door can be complex, especially for the detection of a fish.

In the meeting with Tobias, we concluded that solution number 1 is the best. We will therefore investigate this solution further, perhaps adding a door before the camera area (solution 4). We can also add that in each solution the issue is that we can never be sure that there will be only one fish at a time. We don't know the behaviour of the fish, which is also something that needs to be developed before we start building. A thorough study of the fish we can find in Vaasa is being done in parallel with this reflection.

After mentioning the tasks carried out during the first 4 weeks of the project, we will mention some of the work we are going to do in the short term so that you have a more complete picture of the project.

- Develop more the solution 1 before we start thinking about how we will build the fish classifier. We have some ideas, but we need to discuss about that and try to have just one solution before build something. Otherwise, there will be issues in a couple of weeks.
- Find a solution to add machine vision to the final solution, because without machine vision it will be impossible to build a fish classifier as the customer wants.
- Start modelling the final solution on a 3D software like CATIA V5R19.
- Prepare the water before buy fish, because the bacteria need to develop in the water before arrival of the fish.

5.2 Fish selection

For our project, we bought guppy fish. These fish were recommended to us by salesmen because it is a resistant species which is quickly able to adapt to its environment. Consequently, we took 6 fish of different sizes which was perfect for our wish because the objective of the project is to succeed in sorting the fish by size. Once the fish were purchased, we filled the aquarium and then carried out several water treatments to ensure good water quality.

Nevertheless, we had some problems with the fish. Indeed, five of them died. The explanation for this is probably the poor quality of the water with a pH not suitable for guppy fish. When the first ones died, we tried to correct this but a few days later it happened again. However, we did manage to keep one fish which was useful for the fish classifier tests and particularly useful for the operation of the vision machine.

5.3 Construction

5.3.1 Aquarium

Before we started building anything, we tested the material available in the EPS room.

Test of the material available: In parallel to the research on the fish classifier, we tested the waterproofness of the aquarium in the EPS room. At first, we were afraid that it was in bad condition but after testing and checking the walls it is in working order. However, we are thinking about a structure to put around the aquarium in case it leaks during the semester. We are going to put a drip tray underneath and create a structure to hold the walls that could break with the pressure of the water. We are now sure that this aquarium is suitable for the creation of fish classifiers, we can now concentrate on the structure that will support the aquarium, the structure that we will create for the fish classifier and the preparation of the water for the fish.



Figure 3. Aquarium

After testing the aquarium by filling it with water, we saw that it was strong enough. To strengthen it and avoid any issues, we thought of several solutions. On the one hand, we first thought of creating a wooden structure to hold the glass walls of the aquarium. This solution seemed feasible because we had found pieces of wood in Technobotnia and had access to the wood workshop. However, in discussion with Hans, we concluded that it would be too complicated and time consuming to do.

Consequently, we thought of a simpler solution: combining 3 rows of tie ribs and a retention bin for the washing machines. We will put the retention bin underneath the aquarium and then use tie ribs to hold the walls of the aquarium. We have also added a sensor to the bottom of the tank that will alert us when it detects water. This allows us to react quickly if any leaks appear during the semester.



Tie ribs



Retention bin



Figure 4. Improvement of the aquarium



Figure 5. Leaking sensor

After building the structure, we filled the aquarium as quickly as possible to accommodate the fish. We can say that this solution is one of the cheapest and easiest to realize. We bought a leaking box for 20€, tie rips for 6€ and the sensor was already at our disposal. We will detail a little more about the different expenses in this project towards the end of the report.

5.3.2 First ideas for the design

Once the work around the aquarium was done, we turned to the realization of the fish classifier. Among the sketches of solutions mentioned above, we had to choose one and develop it in order to start with a concrete basis of what we wanted to do. Finally, we decided to keep our first solution: divide the aquarium into 3 parts and connect these 3 parts with pipes. We will also add 2 gates to allow the fish to be directed into the fish part of the aquarium. We will add a door at the entrance of the first pipe. This door will be linked to the machine vision, for instance, as soon as the camera detects a fish, a command will activate the closing of the door. This allows us to be almost certain that there is one and only one fish at the vision machine. The camera will therefore process one fish at a time. The second door is used to direct the fish into either the "Large Fish" or the "Small Fish" section.

We chose this solution because we felt it answered many of our questions. We only have 2 actuators, so we had to think accordingly. Then, there is a permanent link between the part where the fishes are initially and the "Small fishes" part. The fish can go in the opposite direction and be analysed again by the vision machine.

To help understand our system, we drew these ideas on a 3D software called CATIA. This helps to quickly understand the ideas presented. It also allowed us to see what materials we needed. It raised a number of issues that we hadn't thought of. Sometimes an idea is clear in your head but it is very hard or impossible to reproduce it in real life. This 3D model helped us a lot to answer these problems.

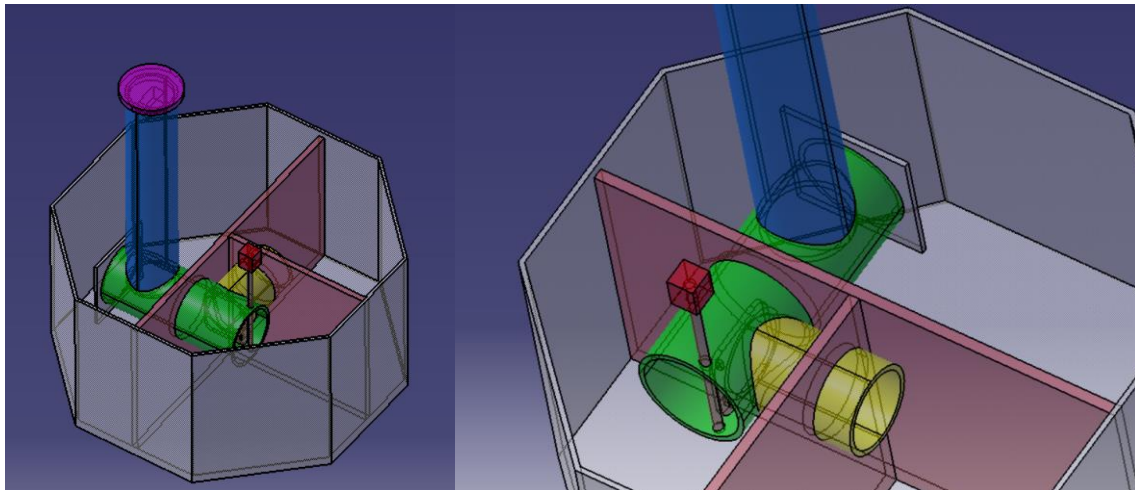


Figure 6. First 3D model

Once the model was approved by the whole team, we started by listing the materials needed to build the fish classifier. We had to consider the budget available, the material already presents in the EPS room and the fact that we can use the 3D printers as we wish.

During the construction of the fish classifier, we thought in parallel about printing some parts in 3D. This allows us to draw the shapes we want and thus have parts that are almost free. Having followed a 3-hour training course on 3D printers, the university let us use the 3D printers. To save time and money, we decided to print some parts, for example the parts for the 2 gates and the part that will support the camera of the vision machine and the Raspberry Pi that goes with it. The 3D printers can help us a lot but we had some issues due to them. Sometimes, some parts do not print correctly, so the design must be redesigned before they are printed again. It can be a big problem on parts that require printing times of around 12 hours.

We were also confronted with many problems. Firstly, we rushed a bit too much in buying fishes. We bought them before we even started building the fish classifier. This caused us to rush through this part of the project as well. Secondly, once the fish classifier was built, we faced other issues. Some of the 3D printed parts didn't work as we wanted them to. We had to constantly rethink the design of the fish classifier.

The original solution was therefore no longer valid. It was also necessary to find a balance between the mechanical part of the fish classifier itself and the more electronic part of the camera vision. This construction led us to carry out many tests. Day after day, we tested different solutions. We were well helped by the availability of 3D printers. We first thought of adding 2 doors, one at the entrance and one in the middle of the fish classifier (distinguishing between the part for big fish and the part for small fish). The one at the entrance will be used to be almost certain that there will be only one fish in front of the camera. So, we designed it on 3D software and printed it.

Once the doors were printed, we decided to get more materials to build with. With the budget available, we bought pipes for the main structure of the fish classifier. We then added 3D parts that we had designed and printed. It would have been impossible to build something without 3D printers because finding parts that are exactly the right size, that fit perfectly into the pipes. Also, the space available in the aquarium is not that big. So, we had to be inventive and creative. Two members of the team went to the market to buy the pipes they wanted. This was a conscious decision; we wanted a pipe that was wide enough but not too wide.

Then, construction could begin. It should be noted that we had decided to stick perfectly to the solution drawn on CATIA. The pipes were cut by us, and the assembly could start. For instance, we have linked the two pipes by making a hole in one of the pipes. The objective will also be to add the door to direct the fish into this space.

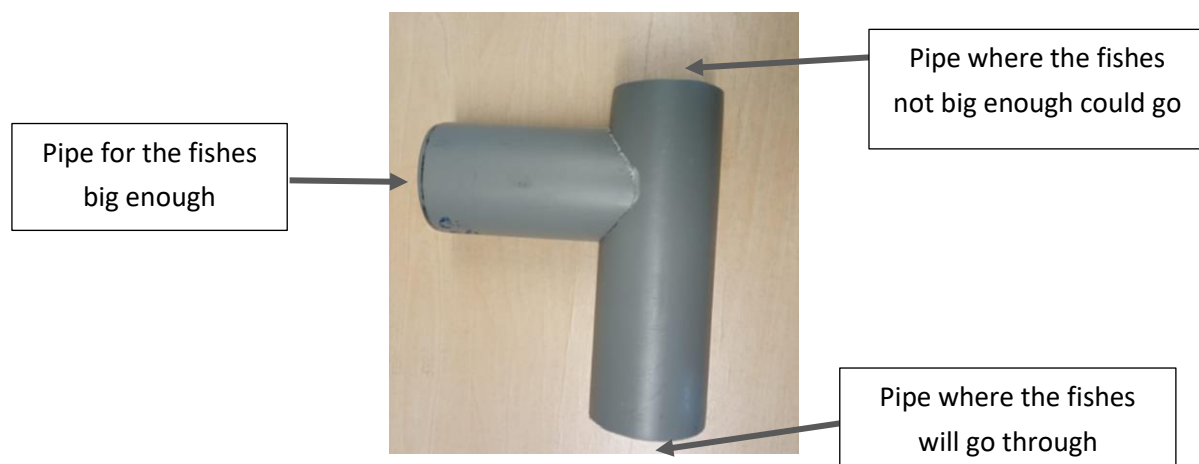


Figure 7. Junction between 2 pipes

As shown above, this pipe assembly connects two pipes. We did the same for the pipe for the camera vision and the pipe where the fishes can go through.

When it came to testing, we realised that there were several problems. It's when you test that you realise things that you hadn't thought of before. So, being confronted with issues that were not foreseeable allows us to bounce back and move forward. During the testing of these parts, we observed that these assemblies were not strong enough. We had to design a part to make it stronger. Another problem we saw was that it would be complicated to make the door that would direct the fish into the pipe. It is too narrow and tricky to make. The main reason for these problems is that we only bought one size of pipe. As the diameters are identical, to join 2 pipes the hole will go to half of the drilled pipe which will weaken it strongly.

So, we printed this 3D part to improve our solution about the junction between the machine vision's pipe and the other one:

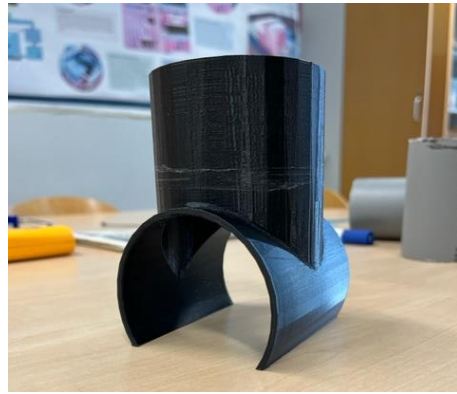
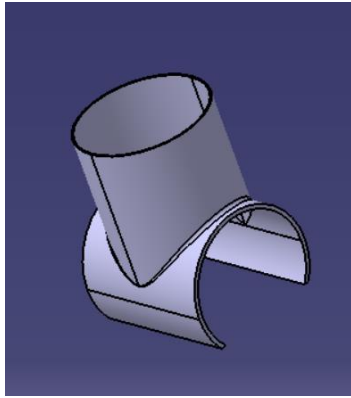


Figure 8. 3D printed part to create the junction

And then we have this result:



Figure 9. Improvement of the junction with the 3D printed part

5.3.3 Gate to separate the fishes

We started building the fish classifier, we had to create something to clearly separate fishes.

As mentioned earlier, the other problem was the narrowness of the pipe to build a door in. Consequently, we rethought the basic solution. We created a kind of cubic elbow with a gate directly integrated. This allowed us to get around a number of design and long-term life concerns. On this elbow, there are 3 inlets/outlets: one to accommodate the hose related to the vision machine, one to direct the larger fish into a part of the aquarium and the last one for the smaller fish.

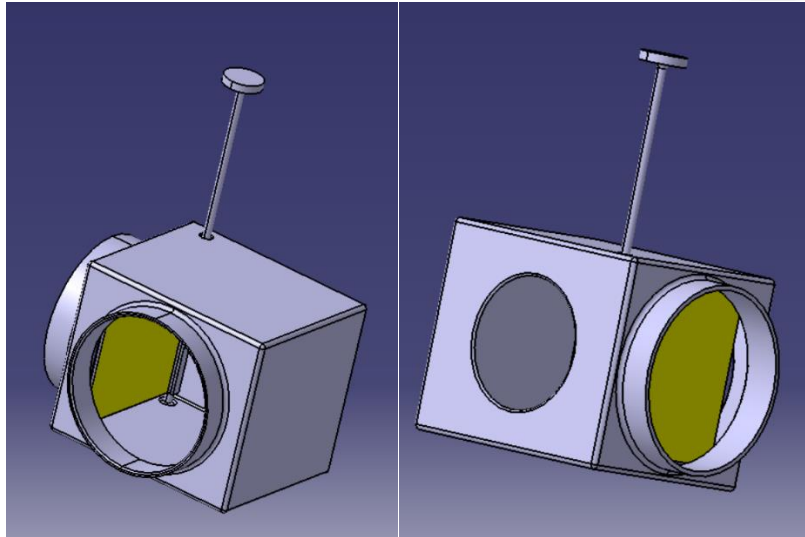


Figure 10. 3D model of the cubic elbow included the gate

The gate piece consists of a long stick with a circular shape printed on it. This shape will allow the servomotor to be linked to the part and thus allow the gate to rotate. We then connected the servomotor to obtain the following result:

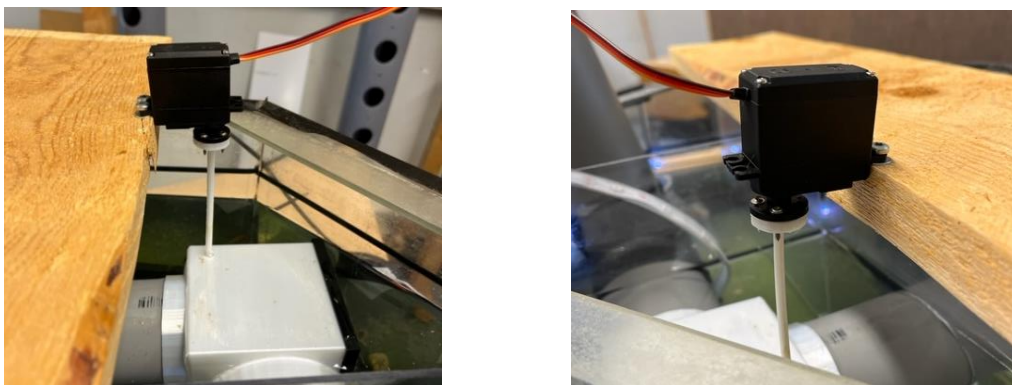


Figure 11. Servomotor to rotate the gate

The servomotor is attached to a piece of wood to hold it in place. This also allows the motor to work properly, it has a place to rest on. The motor is also connected to the Raspberry PI (RPI) which controls the opening and the closing of the door inside the cubic elbow. The part about the RPI will be develop in another part a bit later.

Now a quick example of the servo motor in operation:



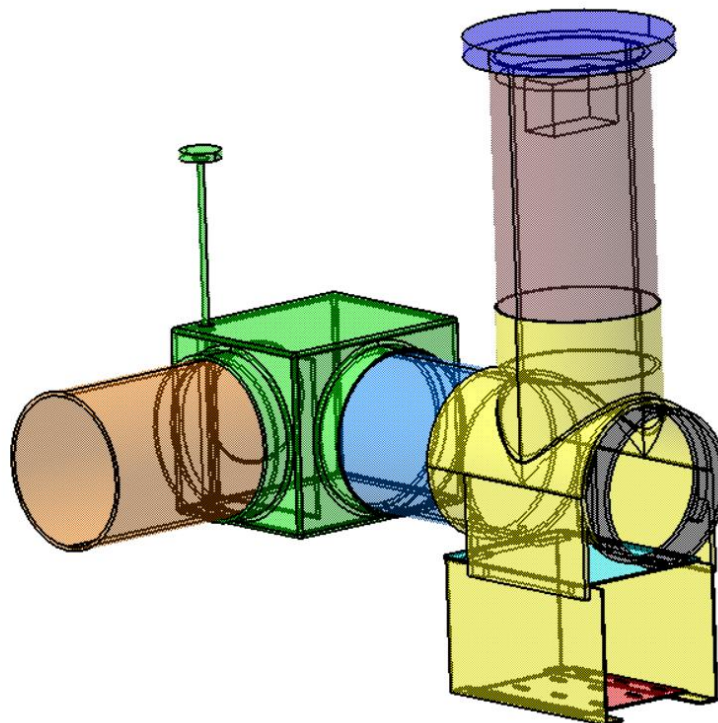
Closed door



Open door

Figure 12. Two different positions for the gate

Finally, after multiple changes to the original solution and several tests, we arrived at the following result:



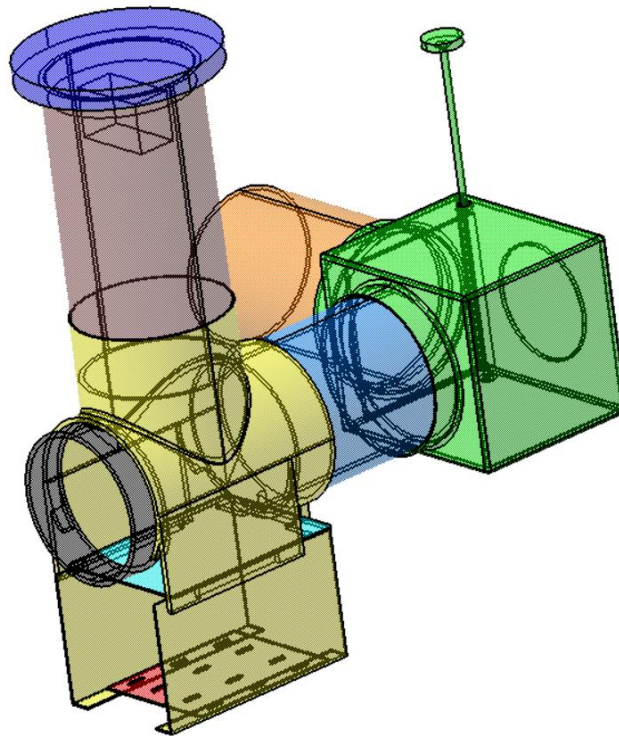


Figure 13. 3D model of the final solution

We created this structure and implemented it in the aquarium. By the way, every colour represents a different piece, and the three blue parts, the red one and the green one are 3D printed parts.

5.3.4 3D printed pieces for the machine vision

To finish with the explanation of the fish classifier, we have to talk about one of the main parts of this construction: the centrepiece linking the machine vision and the rest of the system.

Once most of the fish classifier skeleton was built, we looked at how we were going to graft the vision machine onto the rest. A brainstorming session began, and several successive tests were carried out. We tried to keep the initial solution with the 3D printed junction. The complicated part of the vision machine is to get a clear picture of the fish even though everything is completely submerged. We tried to capture images without adding artificial light, the result was far from what we expected.

Thus, we had to find a way to add artificial lights. Looking around the EPS room, we found some LED's surrounded by plastic. It was therefore possible to put them in the water. Going back to the design, we first had the idea to 3D print another part to house the LEDs. This cylindrical part fits into the existing pipe.

Then, we tested it with the piece below:

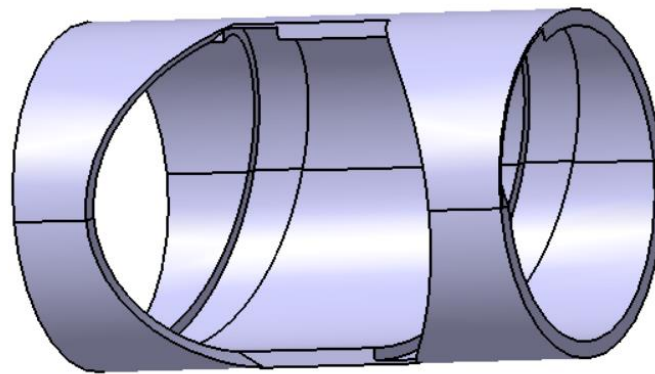


Figure 14. part to hold the LEDs for the machine vision

Three major problems appeared. First, the LED's chain did not fit properly in the groove because the chain is flexible but not infinitely so. Secondly, when capturing images, we could see parts of the LED's which was not ideal, especially for the camera focus. Lately, the brightness with the LED's positioned in a circle did not really help to get sharp pictures. The image was either too bright or not bright enough.

Following this failure, a new technical solution was needed. In discussions within our group and in a meeting with Tobias, we first thought about the result we wanted to have for the photos. Then we thought about the design around this.

The result we wanted for the pictures was to have a white background so that we could distinguish the fish. When a fish passes in front of the lights, it will be easier to detect its shadow and use this image for further processing. To get a white background but not too bright, we tested and saw that the LEDs had to be placed at 6 cm from a white plate. Finally, we had to design a room that took this distance into account. To design the room, we used the existing room connecting the two pipes. We took the 3D model and added a dedicated part for the LEDs. A support for the LEDs and a thin white plate for the background will also be printed. In the end, we end up with the following solution:

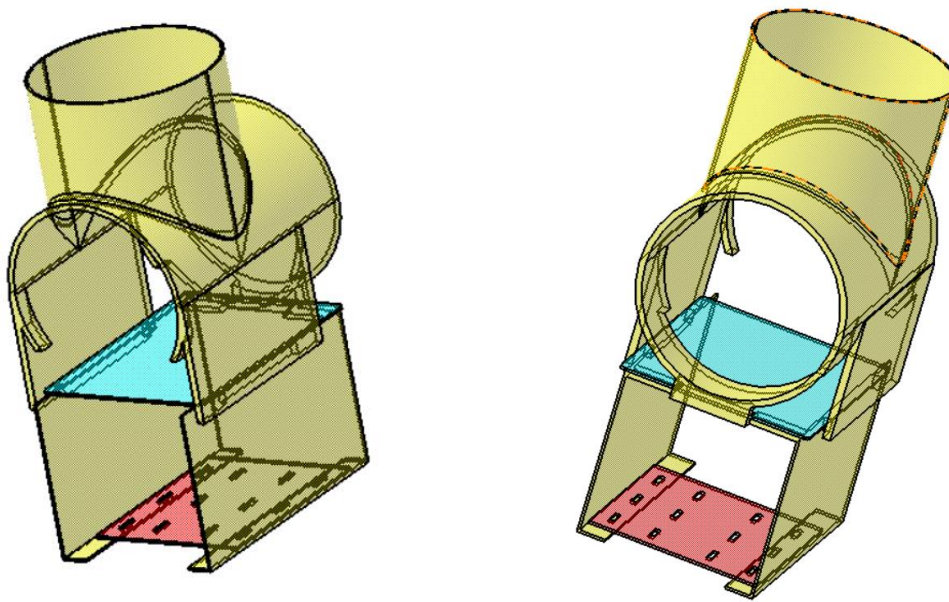


Figure 15. Assembly of 3D parts for the machine vision

As mentioned earlier, to arrive at this final structure, we linked 3 parts together, all 3D printed. Here is the list and the details:

- A thin white piece for the background (blue in the illustration above):

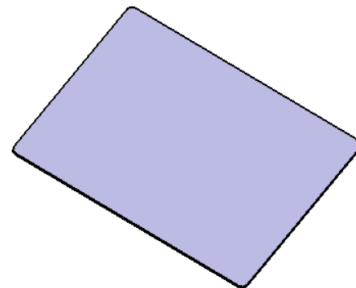


Figure 16. White piece for the background

- A support plate to hang the LEDs, situated 6 cm below the white piece (red in the illustration above):

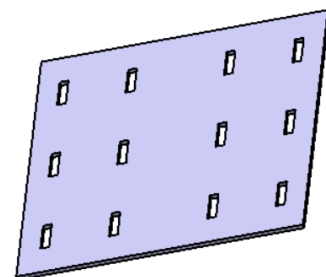


Figure 17. Support for the LEDs

- The main part on which the 2 other rooms will be added (yellow in the illustration above):

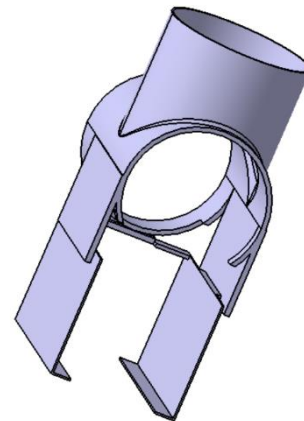


Figure 18. Main part of the machine vision assembly

At the end, we gather all these elements to form what we will call the final version of our fish classifier. Moreover, we had a last 3D part in the top of the camera vision pipe. This 3D printed part holds the camera in place and we did a space to implement a Raspberry PI. The camera is directly linked to the Raspberry PI, so we had to find a spot beside. Therefore, we printed a part that has the 2 functions described. The piece has different shapes: on one side one to fit into the pipe and stay in place and on the other side 4 holes to screw in the Raspberry PI. The Raspberry PI has a key role for the fish classifier, because all of the information, mainly from the camera vision, goes through the RPI. We will more develop his role in the following parts.

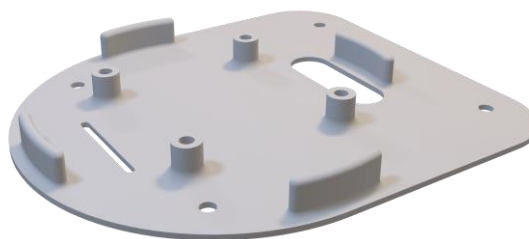


Figure 19: 3D part linking Raspberry and the pipe

Here is the final version of the fish classifier, with first the 3D model made on CATIA V5 software and then some pictures of what we built in reality.

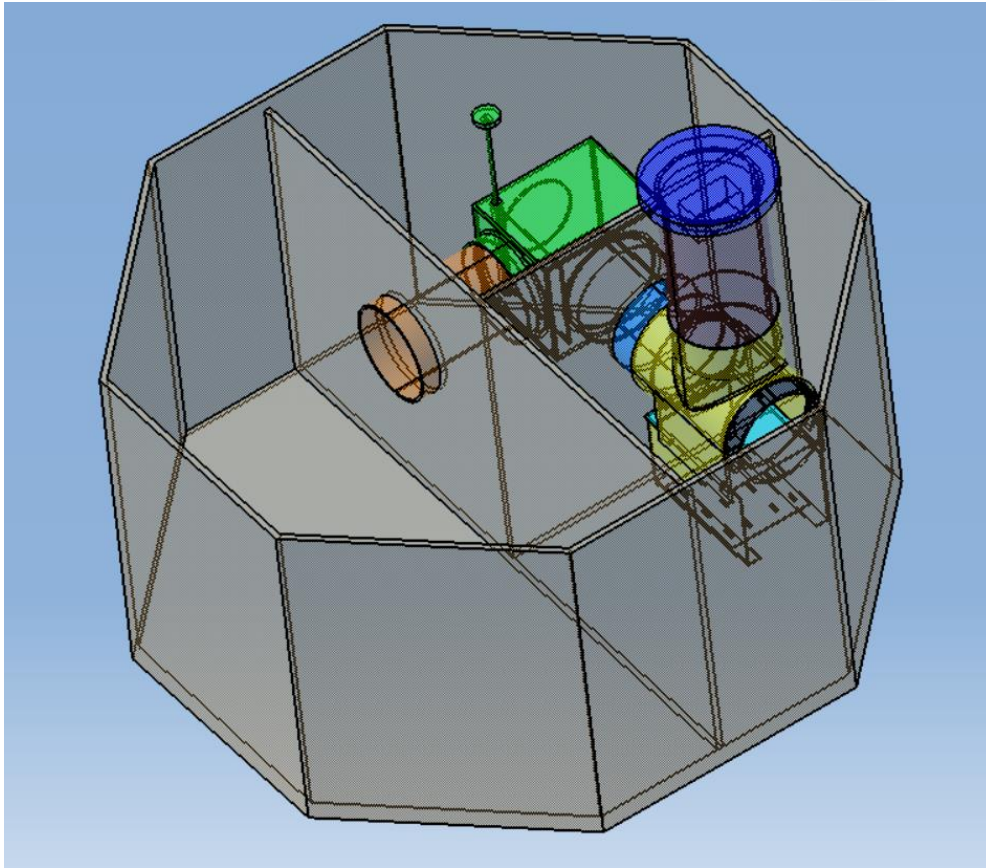


Figure 20. Final 3D model of the fish classifier

To create these parts, we represented them in 3D modelling software, in our case CATIA V5R19. Once the parts were created in CATIA with the desired shapes, we converted them into ".stl" format and then printed them in 3D in the room made available at the university.

For our 3D parts, the material of the parts does not matter to us. Indeed, we do not have any color criteria for the parts created. The only requirement is that the chosen material does not react to water. We used PLA or PETG. These materials are water resistant and do not deteriorate over time in an aquatic environment.

We created two types of supports:

One support that could go under the 3D part allowing the connection of pipes (flat surface at the end of the support) and another support intended to go at the end of the pipe (surface that follows the shape of the pipe at the end of the support).

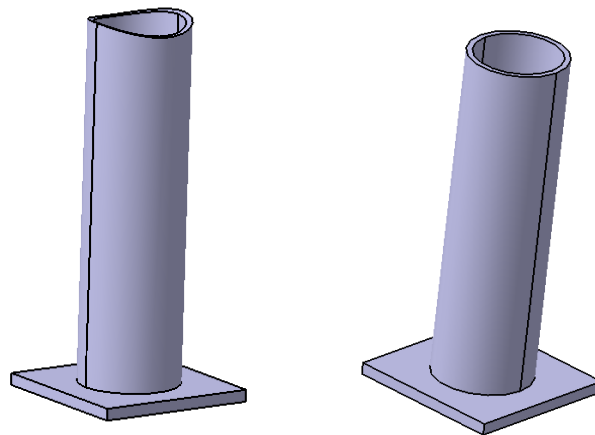


Figure 22. Foot to maintain the structure (pipes and 3D link)

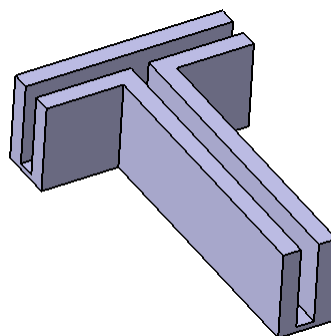


Figure 23. Plexiglass holder

Then, we also created another part in 3D printing to make the junction of the Plexiglas and thus avoid having a gap where the fish could have passed in having been sorted out with the fish classifier. Indeed, with the influence that water can have on a light material like Plexiglas, we had to create a 3D part to ensure the stability and the junction of our parts.

These are relatively simple parts, but they ensure that our system in the aquarium remains in place.

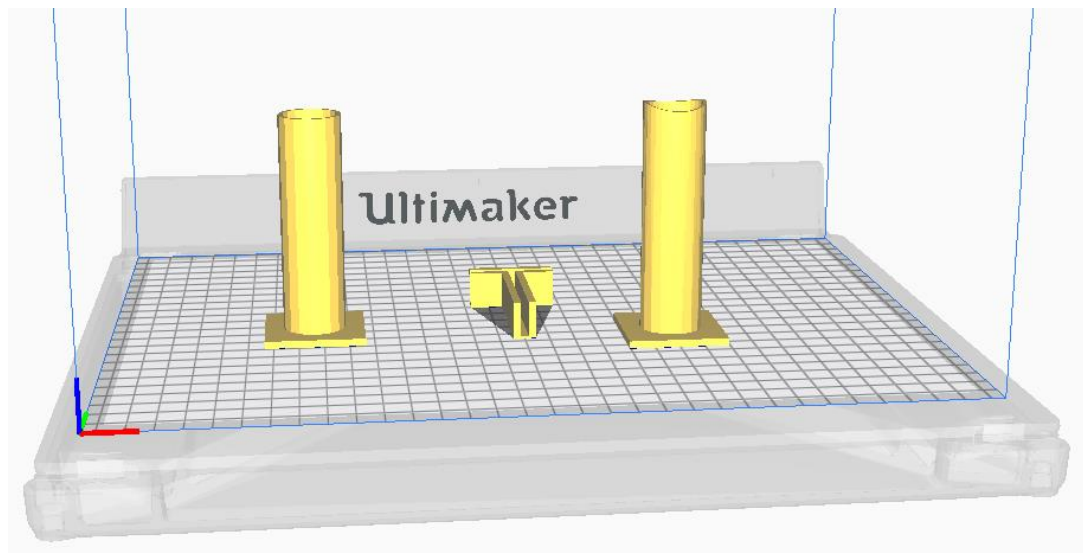


Figure 24. Parts in the 3D printer

Settings for 3D printing of parts:

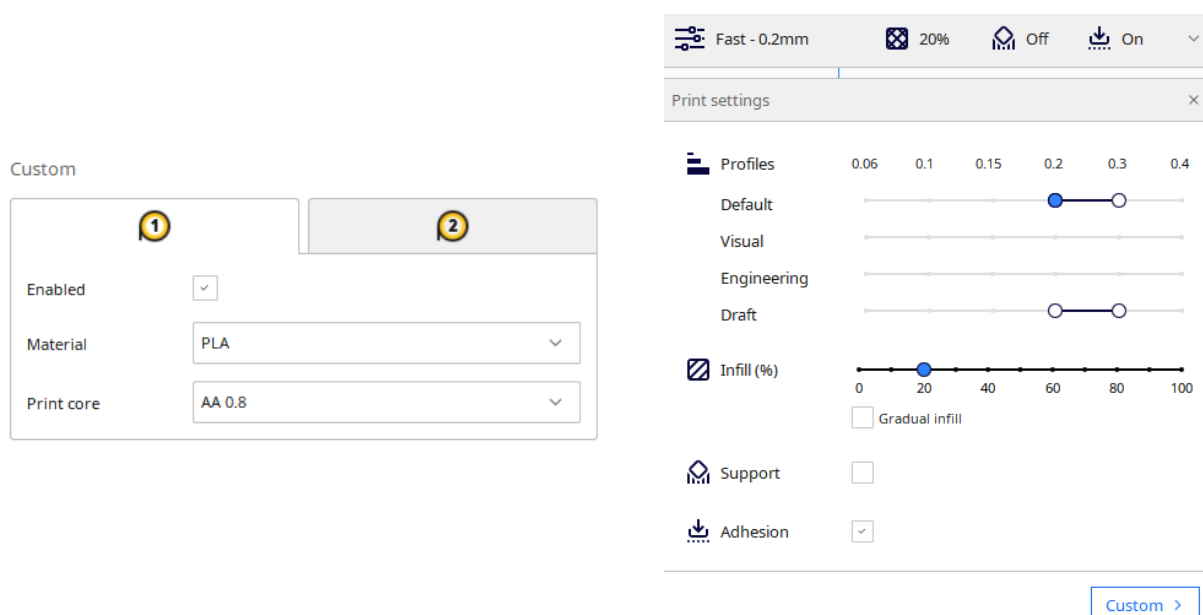


Figure 25. Settings of the 3D printer

With these different settings, we were able to manage the choice of material according to what we have available on the machines. Then we also made different settings so that the 3D printing would not last too long. These settings include the choice of having a part with a 20% thickness fill, but also allowing surface defects with a 0.2 characteristic. We accept that the surfaces are not perfectly smooth because these support parts do not have a primary functionality that requires a perfect surface finish.

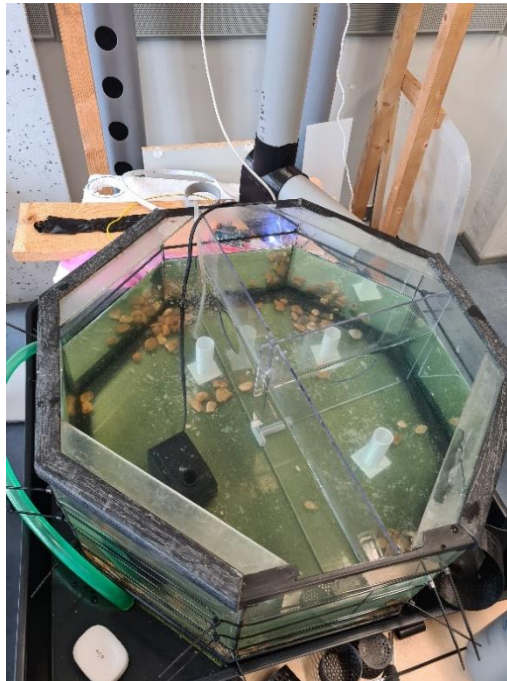


Figure 26. The 4 supports in the aquarium



Figure 27. Vision of the support under the pipe and the holding of the plexiglass

Plate for fixing the servo motor and protecting the electronic components:

We have also added a plate to the top of the aquarium. This plate has, in fact, three major roles. Indeed, it must allow to fix the servomotor but also to maintain the pipe where the camera is located in the good position and finally, the plate also plays the role of protection for all the electronic components (Raspberry PI, etc....) which are present above the pipe at the level of the camera.

The presence of the servo motor is essential as it will allow the door to be rotated, or not, after the image processing. Indeed, the door driven by the servomotor will allow the fish measured and analyzed by the camera system to go into the compartment of the aquarium adapted to its size. This is why the servomotor must be next to the rod connected to the door.

5.5 Machine vision

The machine vision part of the fish classifier is responsible for the recognition of the fish sizes, and potentially other attributes, such as the presence or not of one or more fishes, location of them, etc. This will be done by processing images in real time, there are many alternatives for this. We have been given a Raspberry Pi High Quality Camera, which includes a Sony sensor and interface cable, and a telephoto lens for it.

The Raspberry Pi has processing power enough for real-time image processing, which is highly desirable if fish need to be tracked accurately. Its power is however too limited to use advanced technologies such as neural networks; thus, a more traditional approach must be taken.

5.5.1 Camera lens

Size estimations from images with depth usually requires telecentric lenses, since they provide perspective-free images, which avoids errors caused by perspective with close objects. These lenses are unfortunately very expensive to acquire, mainly because they are only targeted to the industrial market.

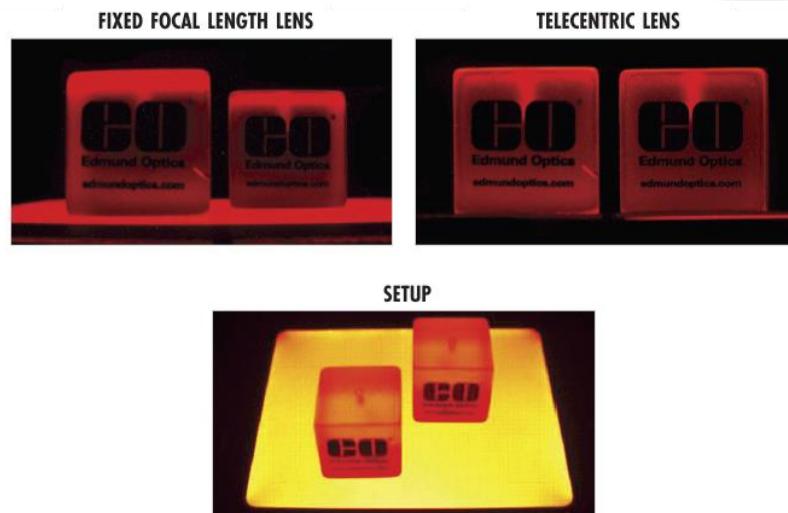


Figure 28. Size distortion in traditional and telecentric lenses (Gregory Hollows, s.f.)

Another solution to this problem involves using a telephoto lens located far enough from the object to measure, which reduces perspective effects. This is the solution that we will go for. Note that this only reduces the effect but doesn't completely remove it; to reduce this even further, we need to make sure that objects to be measured (fish in this case) are always at the same distance from the camera. The downside to this is that it requires creating a structure to hold the camera still quite far from the aquarium itself. This is necessary to focus the fish and to keep them inside the field of view.



Figure 29. Raspberry Pi HQ Camera with telephoto lens mounted on it

5.5.2 Lighting

Lighting is probably the most important factor to account for when creating a Machine Vision system. When trying to accurately recognize the shape of objects, backlight can be used, which produces a high-contrast image with background white and objects close to black. This, however, needs for

background light to be available, which was complicated in our scenario, since LEDs needed to be placed either inside the water (just behind the surface over which fishes to be recognized are), or below the aquarium (which would then require higher power LEDs and may disturb other fish). Also, backlit illumination requires completely opaque objects, while fishes can be lightly translucent.

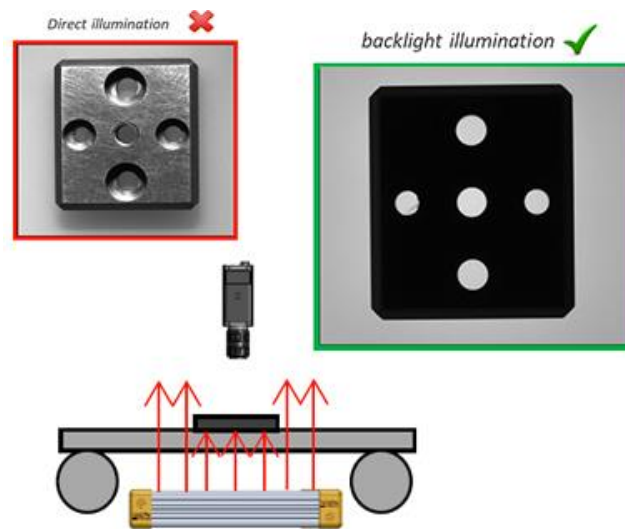


Figure 30. Representation of backlit machine vision (Effilux, s.f.)

Due to the increased complexity of a backlit setup, our first attempts were done using direct lighting, which produced clear enough pictures to be recognizable by human but were too hard to process because of the reduced contrast and presence of other items. Here is an example of an image captured this way:



Figure 31. Fish imaged captured using direct lighting

See how most of the fish is darker than the background, while parts of the tail are brighter because of reflections. This makes it really hard to get the fish contour using traditional image processing methods. Also, the LED strip is partially on the way, which was easy to remove in the processing as long as it stayed still all the time, but the reality was it could move around pretty easily.

For all these reasons, the team decided to move to a backlit system. The physical setup of it, for which a 3D printed part was used, has already been described. A thin plastic sheet is used to diffuse the light from the LEDs, which are now located below the fish and far enough to create a homogeneous background. This results in much more contrasted images:



Figure 32. Fish image captured using background light

5.5.3 Processing software

Python is the language chosen for the processing of the images. Its higher level enables a more time-efficient engineering labor when compared to others such as C, while libraries such as NumPy do the hard processing and memory intensive tasks efficiently. Python also provides great tools to handle saving images and showing them to the user, plus has great support by the Raspberry community with lots of code examples and forums. Python can also be used for the I/O with other components such as the LEDs for lighting or Servo motors for the fish gates, making interoperability between the vision and mechanical parts easy.

We have used OpenCV as well, which is an open-source library with Python support for high-performance machine vision. It is under active development (latest version to date was released in December 2021) and was originally developed by Intel. It is released under the BSC license, one of the main reasons why it has become one of the most widespread tools in the machine vision sector (OpenCV, s.f.).

5.5.4 Capturing images

The first step of the processing is obtaining the images. First of all, we are using a library called PiCamera that handles the interface with the camera hardware in the Raspberry Pi. The first part of

the script imports all the necessary libraries, after which the camera is initialized. The camera configuration is set manually to values that have proven to work well, instead of leaving it in auto mode which is its default behavior. This ensures that captures are consistent, all with the same, exposure, ISO, white balance and others (focus and aperture are controlled manually in the lens and so are consistent by design). Here is the first part of the script:

```
from time import sleep
from picamerax import PiCamera
from picamerax.array import PiRGBArray
import datetime
import cv2
import numpy as np

FPS = 10
SAVE_RATE = FPS
DIFF_LOCAL_THRESHOLD = 0.1
DIFF_THRESHOLD = 0.0004

camera = PiCamera()
camera.resolution = (2400, 1808)
camera.framerate = 4/1
sleep(2)

camera.exposure_mode = 'off'
camera.awb_mode = 'off'

camera.shutter_speed = int(1e6/100)
camera.iso = 200
camera.digital_gain = 1
camera.analog_gain = 1
camera.awb_gains = (5, 1)
camera.exposure_compensation = 0
```

Figure 33. Camera movement detector code, part 1

After the initialization of the camera, a loop runs constantly, in which images are captured at full resolution into a NumPy array (a three-dimensional array with a shape of width, height and three-color channels, red, green and blue).

The script then detects if any movement have occurred from one frame to another, presumably detecting whether a fish is present or not. To speed-up the processing, the image is resized to 30% of its original size (only for the processing, once saved they are kept as captured). The image is then filtered to reduce noise, and the difference between the previous capture and the new one is calculated in absolute value. This will return an image which is completely black by default, with clearer pixels where movement has been detected.

After the difference is calculated, all pixels below a certain threshold are set to 0, this is to remove noise that is inevitably present in the image and always changes a bit from one capture to the next. All pixels are then averaged, if this value is above a minimum threshold, it is considered that there has been a movement, and the script saves the image to a *.jpg* file with the current time and frame count in its name. Optionally, all frames can be saved to a temporary location under */dev/shm/* to preview them in pseudo-real time. This is the code responsible for the processing and saving:

```
prev_output = None

while(True):
    with PiRGBArray(camera) as output:
        camera.capture(output, 'rgb')
        output = cv2.cvtColor(output.array, cv2.COLOR_RGB2BGR)
        print('Captured %dx%d image' % (
            output.shape[1], output.shape[0]))

        frame_count += 1

        kernel = np.ones((3, 3), np.float32)/25
        processed = cv2.resize(output, (0, 0), fx=0.3, fy=0.3)
        processed = cv2.filter2D(processed, -1, kernel)/255
        if(prev_output is None):
            prev_output = processed

        diff = abs(processed-prev_output)
        diff[diff < DIFF_LOCAL_THRESHOLD] = 0
        average_diff = np.mean(diff)
        prev_output = processed

        print(average_diff)
        # print(diff[0, 0])

        if average_diff > DIFF_THRESHOLD:
            cv2.imwrite(f"out/movement_{datetime.datetime.now()}_{frame_c
ount}.jpg".replace(
                ':', '_').replace('-', '_'), output)
            # print("Saved frame due to movement detected")
            print('\033[;32mMovement detected\033[0m')
            pass

        # cv2.imwrite(f"out/frame{start_date}_{frame_count}.jpg", frame)
        # cv2.imwrite("out/frame.e.jpg" ,jpg" , frame)
        # cv2.imwrite("/dev/shm/frame.jpg", output)
        # cv2.imwrite("/dev/shm/frameR.jpg", output[:, :, 2])
        # cv2.imwrite("/dev/shm/frameG.jpg", output[:, :, 1])
        # cv2.imwrite("/dev/shm/frameB.jpg", output[:, :, 0])
        # cv2.imwrite("/dev/shm/diff.jpg", diff*255)
        print("Frame %d" % frame_count)
        print('')
```

Figure 34. Camera movement detector code, part 2

The result of the execution of this script is the following:



Figure 35. Images captured by the movement detector

5.5.5 Processing images

The processing script for the images has been written in an interactive Jupyter Python Notebook, which is really convenient when developing to test different parts of the code selectively. Even though the development has been done locally on the computer, it is completely portable to the Raspberry Pi, and thus could easily run on it in the future. Interacting with servos and other peripherals of the RPi should be seamless as well for future development.

During the future sections of the document the writer will go through all the stages of the processing.

5.5.5.1 Loading the images

Since the development is not done with real-time images (that would require to have a fish constantly going in front of the camera), it is necessary to load the images that were saved by the previous script. It is also a great practice to crop them to remove the sides of the image which are unnecessary and resize them so that the resolution is consistent among different captures. This is done in the following way:

```
# Load test image
fish = cv2.cvtColor(cv2.resize(cv2.imread(
    f'{FOLDER}{FILE_NAME}'), (960, 720)), cv2.COLOR_BGR2RGB)
fish = fish[:,(960-TARGET_SIZE)//2:(960+TARGET_SIZE)//2,]
```

Figure 36. Code for loading images in python

This loads the image as a Numpy array, the cropping is done so that the middle part of the image is kept, and three channels are being used (red, green and blue). Processing speed may be improved in the future by loading the images in grayscale, and potentially lowering the resolution, which could enable real-time processing locally in the RPi.

Since the image contains parts that need to be removed, such as the pipe border, or effects caused by the water surface, a background images needs to be provided as well, which is loaded in the same way. This image shall be captured using the exact same setup but with no contents in it, so that the script can then distinguish what is the actual object. This background image is named *background.jpg* and is loaded and preprocessed in the same way. Here are the sample and background images after rescaling and cropping:

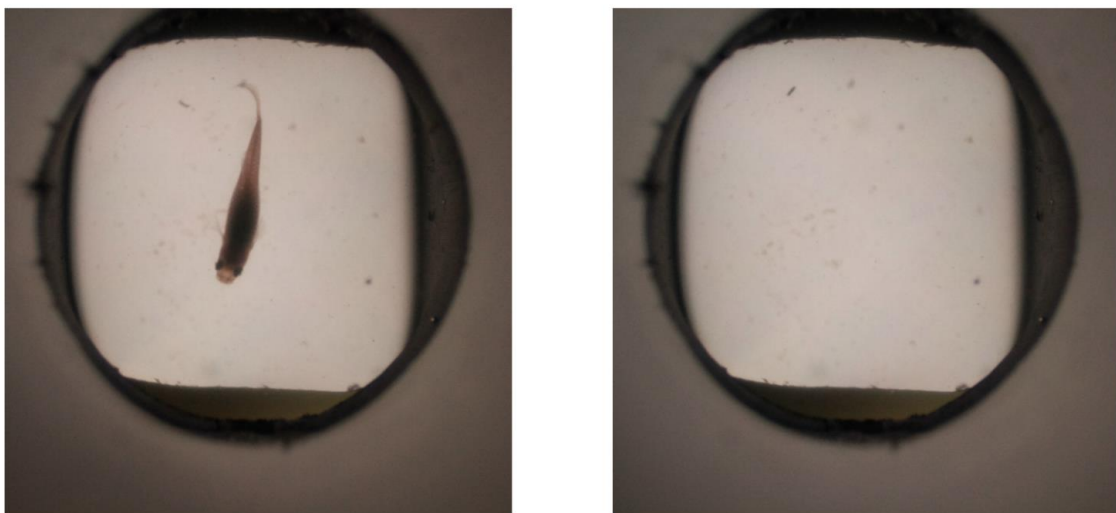


Figure 37. Sample test and background images

This background image has been handpicked for the testing. In the future, the RPi would automatically obtain updated images of the background by, for instance, taking a new one occasionally when no fish is detected. This would ensure that the processing works properly even under slight changes in the background, like the camera moving or deposits forming on the diffuser.

5.5.5.2 Calculating difference image

In order to get the background removed, both the fish and background images are converted to grayscale (these are displayed with a heatmap representation to improve clarity) and stored as signed 16-bit integers. This is done to avoid overflow when subtracting one image to another (the default unsigned 8-bit type cannot store negative numbers). The absolute value of the difference is then calculated and converted back to the default image format.

```
# Convert images to grayscale
# Type is extended to signed integer 16-bit to support negative values
when subtracting
grayscale = cv2.cvtColor(image, cv2.COLOR_RGB2GRAY).astype(np.int16)
background_gray = cv2.cvtColor(
    background, cv2.COLOR_RGB2GRAY).astype(np.int16)
# Calculate absolute difference between images
diff = abs(grayscale - background_gray).astype(np.uint8)
```

Figure 38. Code for calculating image difference

The previous code produces the following result, in which only the fish, and some unavoidable noise is present:

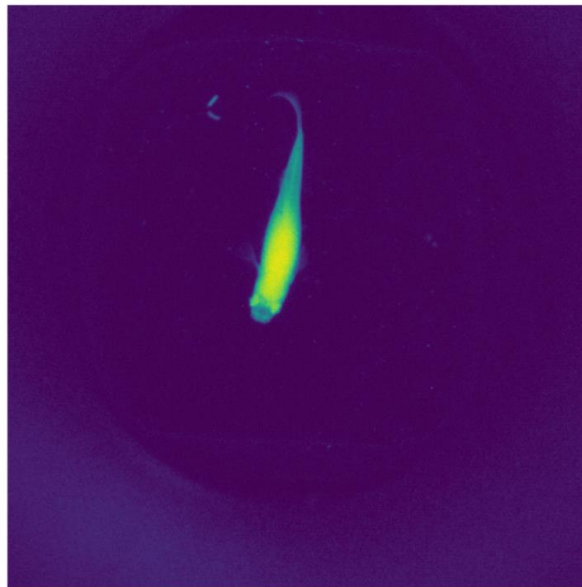


Figure 39. Fish image difference heatmap

5.5.5.3 Binarization and size filtration

Even though the previous image shows the fish pretty obviously for a human, it is not yet completely understandable for a computer. To help with this, the image is binarized, which means that all pixels are converted to either black or white (no grayscale). The simplest algorithm converts all pixels under a certain threshold to black, and the rest to white. To get the optimal threshold Otsu was used, after which it was determined that a value of ~55 produces pretty nice results. This threshold is then hardcoded, because otherwise Otsu would return a nonsense threshold when no fish is in the image, thus producing terrible results.

After the binarization, a small dilate is executed, which helps remove any accidental gaps that may be present due to noise and produces more rounded contours.

Finally, connected components are found, which means that all white areas of the images are classified as a single object. OpenCV also returns the area (in pixels) of each object, which allows us to remove any small objects (bits of food have proven to be always in the way for instance).

```
# Otsu was used to determine optimal threshold value, which turned out to
be around 55
# otsu_threshold, binary = cv2.threshold(
#     grayscale, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU,)
# print(otsu_threshold)
_, binary = cv2.threshold(diff, 55, 255, cv2.THRESH_BINARY)

# Dilate to avoid small holes
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (3, 3))
dilated = cv2.morphologyEx(binary, cv2.MORPH_DILATE, kernel)

# Get connected components with statistics
num_labels, img_labels, stats, cg = cv2.connectedComponentsWithStats(
    dilated)
# Remove any components that are too small
for (i, stat) in enumerate(stats):
    if stat[4] < TARGET_SIZE*TARGET_SIZE//200:
        dilated[img_labels == i] = 0
# Optionally erode again to recover original size
# eroded = cv2.morphologyEx(dilated, cv2.MORPH_ERODE, kernel)
```

Figure 40. Code for binarizing and filtering by size



Figure 41. Binarized (left) and filtered by size (right) fish images

In our sample image there are two objects visible after binarization, only one of which is the fish. The other one was properly removed by the size filter.

5.5.5.4 Size estimation

From the last image of the processing, it is now possible to obtain the area of the fish in pixels. First of all, connected components are calculated again (with everything but the fish already removed from the image). If everything went properly, only 1 component should be found (background) if there is no fish in the image, or 2 components (background + fish) if there is one fish.

In the case of only one fish, we want to know its size. There are a few options on what to do to get this size. First off, we obtain the contour of the fish, from which we can fit an ellipse, which is useful to know its length and width. Even though we do it for demonstration purposes, it turned out not to be really useful since fish are not always straight, thus cannot always be properly fitted into an ellipse.

The second option, and the one we went for, is to obtain the surface of the object in pixels, which can then be converted to a physical equivalent (ignoring any slight deviations due to perspective, changing distance to the camera, etc.). From the test images it was determined that the round area in which the fish can be (the end of the pipe) takes ~41% of the image surface, which we can calculate (in pixels) because we know the dimensions of the square image. The physical area of the pipe section is also known since it is 7cm in diameter. With all this information it is finally possible to obtain the size of the fish in cm².

To generate the output image, the number of fish detected and its size (in the case of exactly one) is written into the original image, as well as the fitted ellipse in green and the fish area in blue.


```
# Get again connected components with statistics, ideally only background and
fish are present now
num_labels, img_labels, stats, cg = cv2.connectedComponentsWithStats(
    dilated)

# Copy the input image to draw on it
output = np.copy(fish)
# Print zones detected as fish on the output image
output[dilated == 255] = (0, 0, 255)

output_text = ''
if len(stats) == 1:
    output_text = f'No fish detected!!'
elif len(stats) == 2:
    # Get contours of the fish
    contours, hierarchy = cv2.findContours(
        dilated, cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_NONE)

    # Assert only one contour was found
    assert len(contours) == 1, 'More than one contour found!'
    contour = contours[0]

    # Fit fish to ellipse
    ellipse = cv2.fitEllipse(contour)
    # Draw ellipse on output image
    cv2.ellipse(output, ellipse, (0, 255, 0), 2)

    # Calculate size of the fish
    # Complete area of the pipe is that of a Ø7cm circle
    pipe_area = np.pi * 7 * 7/4
    # The pipe takes around 41% of the image area based on our
calculations
    pipe_pixel_area = (TARGET_SIZE * TARGET_SIZE)*0.41
    area = cv2.contourArea(contour)/pipe_pixel_area*pipe_area

    output_text = f'One fish detected!! Area: {area:.2f}cm^2'
else:
    output_text = f'{len(stats)-1} fish detected!!'

cv2.putText(output, output_text, (TARGET_SIZE //
    4, int(TARGET_SIZE*0.94)),
cv2.FONT_HERSHEY_PLAIN, 1.5, (255, 0, 255))
```

Figure 42. Code for estimating fish size

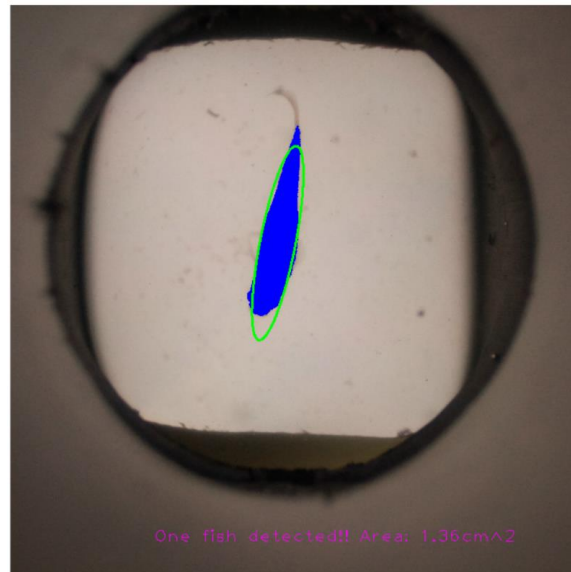


Figure 43. Output of the fish size estimation

This code produces the output just seen above, but the fish size could also be used in the future to decide whether the classifier mechanisms need to be operated or not.

The size estimator was validated by running the same code with other objects of known size. It has proven to give pretty reliable estimations, even though the values can vary slightly depending on the position of the object. For instance, placing them close to the borders of the pipe produces a distortion because of the water surface tension which could affect the results. Another problem could be objects which are only partially into the visible area. This should not be a problem though, because in both cases the returned size is always less than the actual one, thus the classifier would not fire. In this scenario, false-negatives are acceptable, false-positives are not. Here is an example of the validation test that was done:

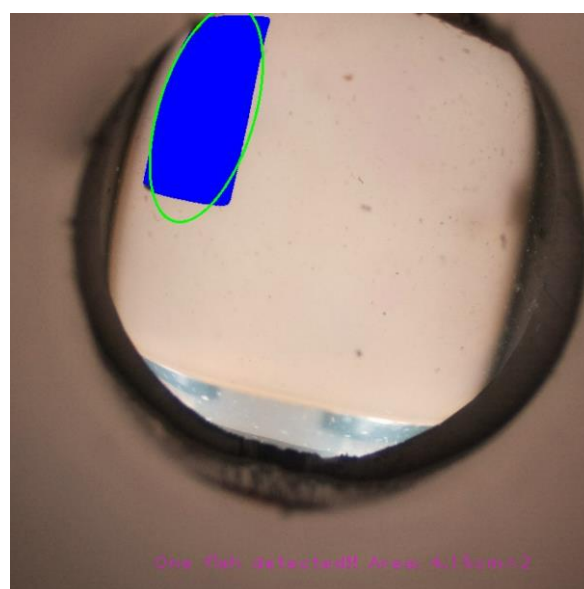


Figure 44. Size estimator validation with object of known size.

Measured size of the object was 5.25 cm², the one estimated by the classifier was 4.15cm². This difference, as stated before, is mainly because of the object placement close to the pipe border, which causes distortion (look at the top-left corner of the object). As observed, measurements are more reliable when placed in the middle of the viewport and tend to be lower than reality when placed close to the borders.

5.6 Conclusion, advice and problems

In this section we will discuss about the problems we had, some advice for future groups working on the project and a general conclusion on one of the main parts of the project. First of all, we have been confronted with different issues, but it is an integral part of EPS projects, especially those involving manual work. We tried and tested a lot before finding the good solution or a solution answering our questions.

- Problems encountered:

We had some problems with the 3D printers themselves but also with some designs. Sometimes, 3D printers didn't print properly and other times we didn't design well the pieces we wanted. Another issue was the thinking about the conception of the fish classifier, because we had thought of a solution without worrying about whether or not it would be easy to achieve. So, we changed our mind during the project, and this was possible thanks to the presence of 3D printers because we can create parts endlessly and in a short time. By the way, we changed the design of the gate to separate the fish and we abandoned the idea of putting a door before the camera to lock the fish in the fish classifier. To finish about our problems, we can talk about the behaviour of the fishes.

- Tips:

After spending one semester on the Hygrow Aquaponics project, the team wants to give some tips for the following team which will work on this project. For instance, if you have to build a fish classifier another time, make sure it is fully constructed before you buy the fish. This will allow you to get the most out of it and make the necessary changes. Secondly, try to use 3D printers as much as possible. They are easy to use, accessible and can generate almost any shape in a short time. Finally, to improve the fish classifier, try to influence the behaviour of the fish. For example, once the photo has been taken and analysed, why not use a flash to make the fish run in the direction or activate a pump to generate a current that will drive them towards the exit.

To conclude this part about the fish classifier, we can say we had several problems that we more or less solved. We can't say that we have a fully functional fish classifier, but there is a solid base to build on. The fish classifier can be useful for later use in the greenhouse to feed fish according to their size for example. Nevertheless, the machine vision that we will now detail is one of the strong points of this fish classifier, so it can be reused. In the case of our project, it can be interesting to see the growth of fish in the aquarium for example.

6 Automation with Home Assistant

One of the tasks of the team was to improve the greenhouse and prepare it for Finnish winter. Previous designs of the water tank have shown weaknesses resisting freezing temperatures, to the point of pipes breaking and water spilling. For this reason, there's currently a single tank in use, with no external parts containing water. Heating is done from the inside, with two aquarium heaters making a total of ~270W, which is also a change from the previous external heater.

The automation is based on Home Assistant, an open-source platform for the Internet of Things. It is capable of handling a wide range of devices thanks to its modular design. Many manufacturers offer compatibility with the platform, plus custom devices can also be integrated, which is supported by the vast community behind HA (Home Assistant). The platform is hosted in a Raspberry Pi, located within the Greenhouse; a 4G router creates a Wi-Fi access point to which the RPi and other devices connect. Commercial products work together with other custom-made sensors:

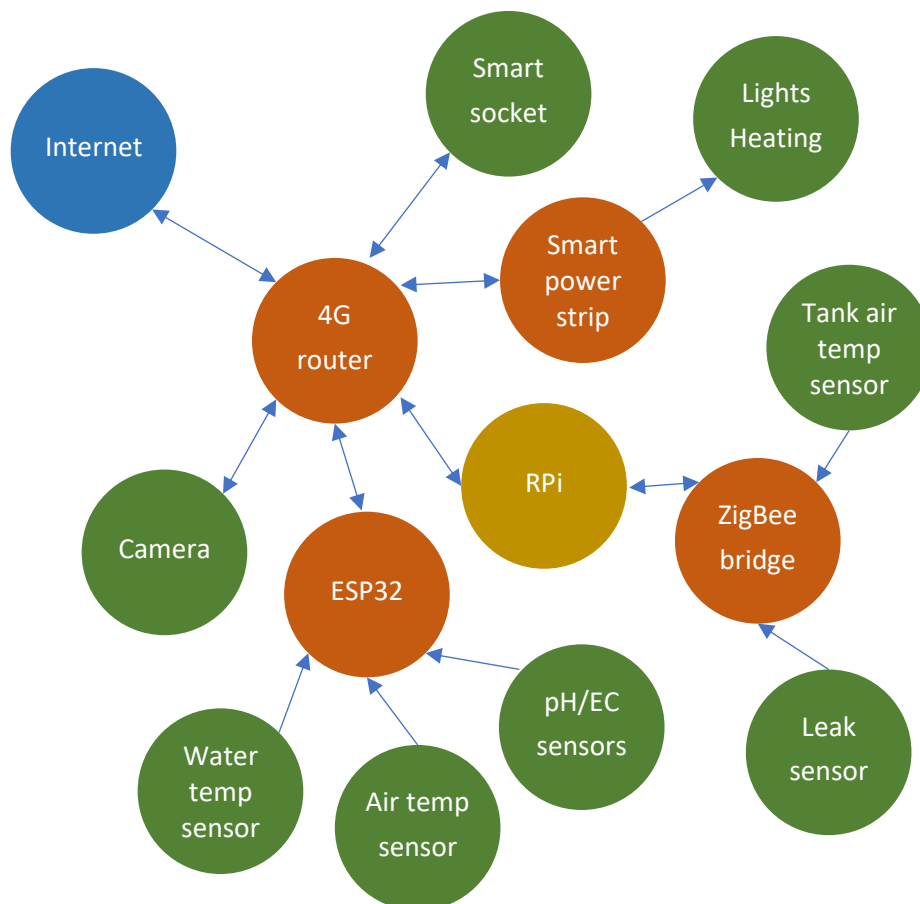


Figure 45. Automation diagram

The Raspberry Pi accesses all sensors and actuators either directly (Zigbee bridge for instance), locally over Wi-Fi (camera, ESP32 and smart socket), or through the manufacturer servers remotely (smart power strip). The latter is undesirable, since this makes the Internet connectivity a possible failure

point, which could cause a loss of control for certain device. Other than that, the Internet is not really necessary, except for remote monitoring which will be covered later in this document.

Home Assistant provides a user interface to configure integration with all devices, configure automations and view a dashboard with historical data. This was already existing at the beginning of the semester, but some changes have been made to improve usability. This is the current look of the dashboard:

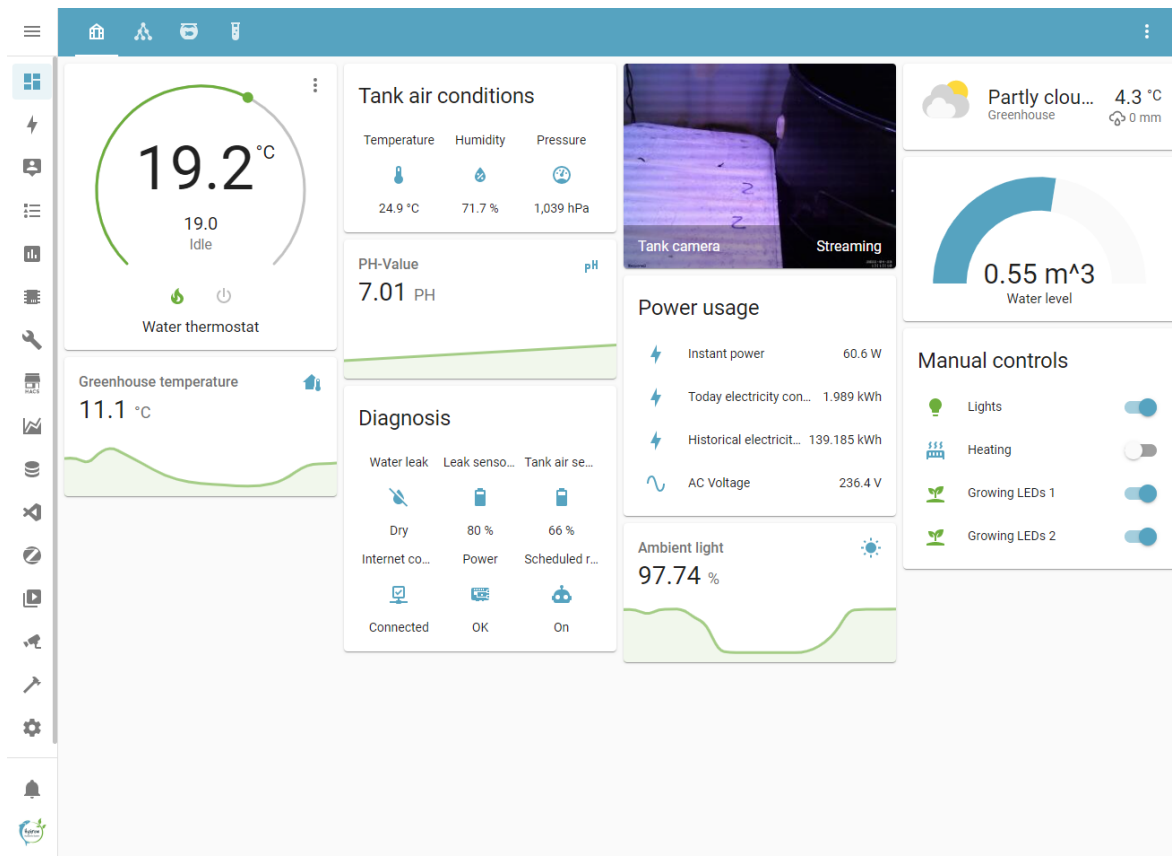


Figure 46. Home Assistant dashboards. EC and pH sensor missing

HA enables a vast number of customizations. The dashboard itself has been fully reorganized, removing those items that were not useful at all, and adding some others. This can easily be done through the main interface, and is well explained within the HA official documentation, so it needs no further explanation here.

6.1 Editing configuration

HA provides many of its configurations over the user interface, which means it is pretty intuitive to do and modify. This includes changing the layout of the dashboard, adding and or removing devices from different manufacturers, and configuring some basic automation rules.

Other sets of configurations are not available from the interface, mainly due to the platform being under active development, and need to be changed directly using the configuration files. These files use the *yaml* format, and can be edited using the file editor (wrench icon) inside the web interface:

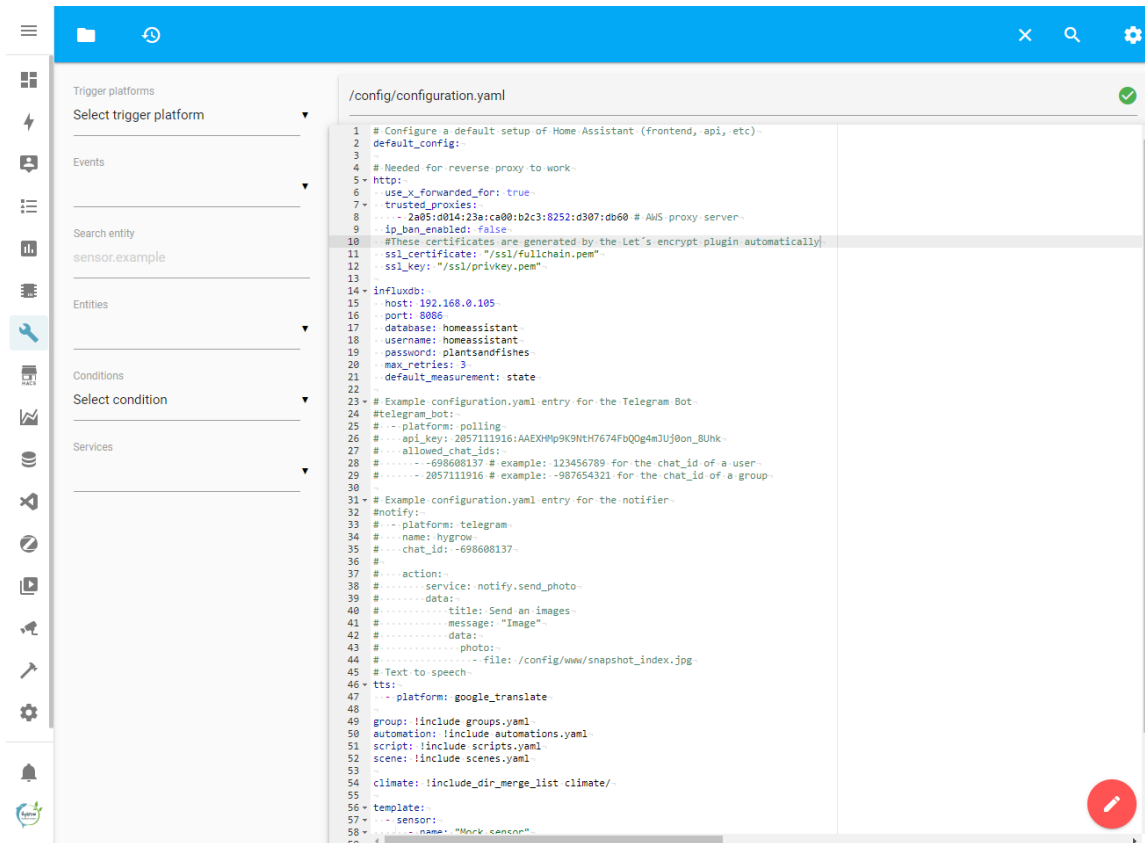


Figure 47. HA File editor, *configuration.yaml*

The entry point for all the configurations is the *configuration.yaml* file. Some sections refer to other configuration files using the *include* directive, which is used when more parameters are needed to keep everything clear. The explanation for each section will be detailed when needed in the following sections.

6.2 Interface theme

The looks of the HA interface have been updated with a custom theme, so that it matches the Hygrow Aquaponics branding. Dark and light themes are also supported, depending on the local device specific configuration. The theme is configured in the *frontend* section of the main configuration file, which refers to *frontend.yaml*. Here, two main colors are defined, primary and accent (blue and green taken from the logo). All other entries refer to certain colors of the interface, to which either of the two main colors are assigned:

```
/config/frontend.yaml
1 themes:
2   hygrow_theme:
3     primary-color: '#56A3C0'
4     state-icon-color: 'var(--primary-color)'
5     state-icon-active-color: 'var(--accent-color)'
6     switch-checked-color: 'var(--primary-color)'
7     state-climate-heat-color: 'var(--accent-color)'
8     info-color: 'var(--primary-color)'
9   modes:
10    light:
11      accent-color: '#6DAE3C'
12    dark:
13      accent-color: '#6DAE3C'
```

Figure 48. Frontend theme configuration

For this theme to be applied to anyone logging in, it needs to be applied when booting HA, for which an automation has been created. This is configured under *Configuration -> Automations & Scenes -> Automation*. Then entry is called *Set Home Assistant theme at startup*, it is set to run on start of Home Assistant, and it calls a service to set the *hygrow_theme* to all devices.

6.3 Heating system

The automation behind the heating system has been removed from Node-RED, which was the platform used before for all the automations, in favor of the native thermostat element for HA, which provides a nice-looking interface allowing to dynamically set temperature. The custom element uses the water temperature entity to measure, then turns on the heaters when temperature drops 0.15°C below the set point and turns it back off when it goes 0.15°C above the set point.

```
- platform: generic_thermostat
  name: Water thermostat
  heater: switch.mains_power_sockets_socket_3
  target_sensor: sensor.water_temperature
  min_temp: 10
  max_temp: 25
  ac_mode: false
  cold_tolerance: 0.15
  hot_tolerance: 0.15
  precision: 0.1
```

Figure 49. HA configuration for thermostat element

The following figure shows the heating system working, including set temperature, actual temperature, and heaters on/off:

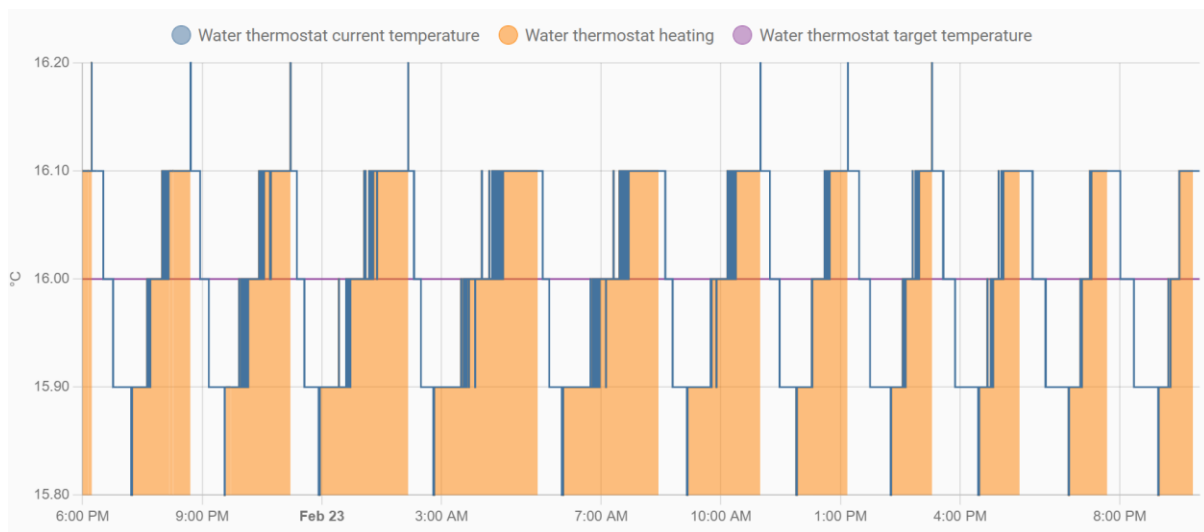


Figure 50. Automation heating

The hysteresis in the temperature can be noticed in the figure. Note that the heating periods tend to be longer the lower the temperature is outside, coldest time in the night shown was -14°C inside the greenhouse. Periods off show that the system could stand even lower temperatures with the heaters currently in use, this could be further improved with better greenhouse and tank isolations or higher power resistors.

Also note that the heaters being used have their own temperature control mechanism. They are designed to be continuously powered, and automatically turn off when the set temperature is reached. This behavior can be overwritten by configuring them to a temperature much higher than the ones that are going to be set inside HA, so that they always heat when powered.

6.4 Lights

Another aspect that is controlled by Home Assistant in the Raspberry Pi are the lights. The fish tank is equipped with whole-spectrum lights, specifically meant to replace sunlight in growing environments:



Figure 51. Water tank with growing LEDs. Taken from final report Spring 2021

These lights need to emulate sunlight, and thus turn on/off at specific times (plants may stand continuous sunlight, but fish need day/night period). This automation was previously achieved using Node-RED as well, it is now ported to HA. Two entries are needed for this, one is responsible for turning the lights on in the morning, while the other turns them off in the night:





<input checked="" type="checkbox"/> Tank light OFF	April 28, 2022, 10:00 PM	RUN ACTIONS	  
<input checked="" type="checkbox"/> Tank light ON	April 29, 2022, 8:00 AM	RUN ACTIONS	  

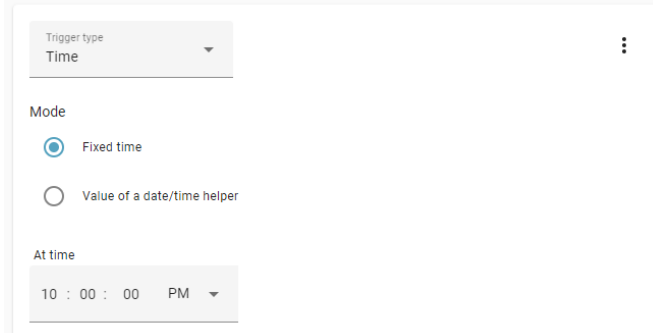
Figure 52. HA automation for the tank lights

When to fire the automation is configured in the triggers section of the automation. HA allows for complex automations, such as turning on/off some appliance based on the detection of motion or configuring the temperature depending on the presence of people in a room. In our case, both entries are time-based, they fire once every day at a given time:

Triggers

Triggers are what starts the processing of an automation rule. It is possible to specify multiple triggers for the same rule. Once a trigger starts, Home Assistant will validate the conditions, if any, and call the action.

[Learn more about triggers](#)



Trigger type
Time

Mode

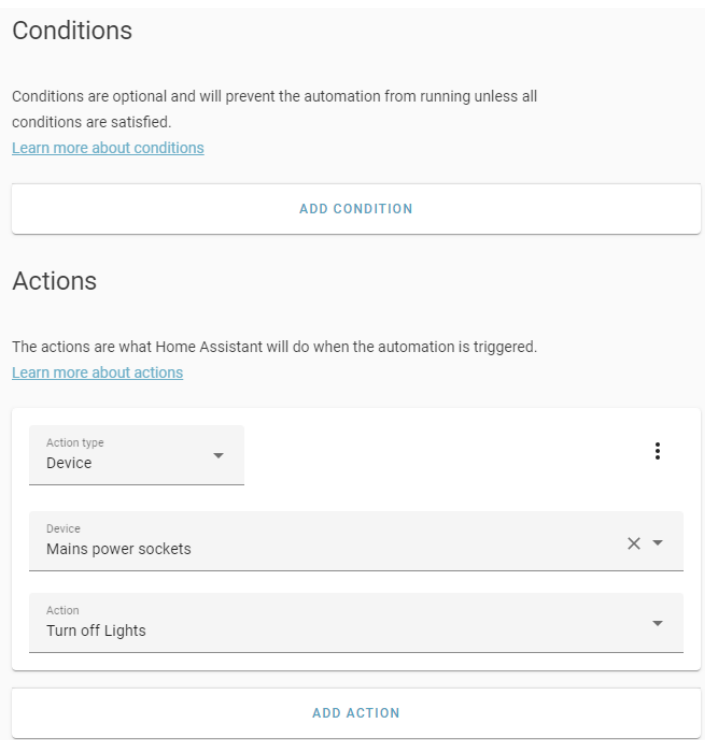
☒ Fixed time

☐ Value of a date/time helper

At time
10 : 00 : 00 PM

Figure 53. Triggers configuration for the light automation

Conditions specify additional requirements that need to be met in order to fire the automation. In our case it will fire every day at the given time, so no more conditions are needed. Finally, actions specify what to do when the automation is fired:



Conditions

Conditions are optional and will prevent the automation from running unless all conditions are satisfied.

[Learn more about conditions](#)

ADD CONDITION

Actions

The actions are what Home Assistant will do when the automation is triggered.

[Learn more about actions](#)

Action type
Device

Device
Mains power sockets

Action
Turn off Lights

ADD ACTION

Figure 54. Conditions and actions for the light automation

6.5 ESPHome

“ESPHome is a system to control your ESP8266/ESP32 by simple yet powerful configuration files and control them remotely through Home Automation systems.” (ESPHome, 2022). This platform allows users to create custom-made sensors, based on either ESP8266 or ESP32 microcontrollers, all of which provide plentiful of I/O interface, as well as Wi-Fi connectivity. The main benefit of ESPHome is that it requires no programming to get sensors working, a single configuration file is enough, and it supports most DIY sensor available in the market.

The ESPHome plugin handles all the integration with Home Assistant, which makes it incredibly seamless. It also provides a web interface to edit configuration files. Programming of the devices can be made either using Serial connectivity over USB, or wirelessly (this requires the device to be already running ESPHome, which means it is only suitable for updates).

A single device called “example” is created, its name is rather inappropriate, but has been left unchanged not to break data history inside HA and other configurations. This is the current configuration of the device, in YAML format:



```
1 # esphome:
2 name: example
3 platform: ESP32
4 board: esp32dev
5
6 # Enable logging
7 logger:
8
9 # Enable Home Assistant API
10 api:
11
12
13 ota:
14   password: "d42e1ee4bf10532473a5834a80bf113a"
15
16
17 wifi:
18   ssid: "AGH"
19   password: "plantsandfishes"
20
21 # Enable fallback hotspot (captive portal) in case wifi connection fails
22 ap:
23   ssid: "HygrowAquaponicsESP"
24   password: "plantsandfishes"
25
26 captive_portal:
27
28
29 dallas:
30   - pin: 5
31     update_interval: 1s
32
33
34
35 sensor:
36   - platform: dallas
37     address: "0xcD9516711F49FF28"
38     accuracy_decimals: 1
39     filters:
40       - sliding_window_moving_average:
41         window_size: 100
42         send_every: 15
43     name: "Water Temperature"
44
45   - platform: dallas
46     address: "0xc508080842804E28"
47     accuracy_decimals: 1
48     name: "Water Temperature"
```

SAVE INSTALL CLOSE

Figure 55. ESPHome device configuration

All basic configurations for ESPHome are written first, including board name, version of the hardware that is being used, Wi-Fi SSID and password and others. Then the sensors section defines all input devices connected to the ESP32 board, the integration automatically creates a HA entity to match the sensor in the board.

The Dallas sensors are a family of temperature sensors that use a 1-Wire bus to connect to the master using a single connection, all of them share the same bus and have a unique address to identify themselves.

Analog sensors are defined as “adc”, which stands for analog-to-digital-converter. Most of the sensors under use include a filter, which can be either median, or sliding window average, to reduce noise effects.

6.6 Water level sensor

A water level sensor allows to monitor the amount of water that is inside the tank, which ultimately serves maintainers to know when a refill is necessary. It could also be used in the future (when more tanks are added) within the automation, to control the pumps based on the level of each tank.

Multiple solutions are available to do this. The simplest ones being digital on/off sensors, which are used when the only information needed is whether the level is higher or lower than a certain value. One option to implement this was to place two sensors in the minimum and maximum working levels respectively, which would allow to fire error events whenever the level was not within the specified range. This would be really limited though, since modifying the range would need to physically move the sensors, and the information that can be obtained from them is really limited.

For these reasons, an analog sensor is preferred, there are multiple commercial solutions for this, some of which measure pressure differential and calculate depth from the hydrostatic pressure. Industrial solutions tend to be quite expensive because they provide a level of reliability that is not needed for this project. We decided to go on making our own sensor, using the materials that are already available, which consist of an ultrasonic distance sensor, a pipe, polystyrene and a custom 3D printed part.



Figure 56. Ultrasonic distance sensor

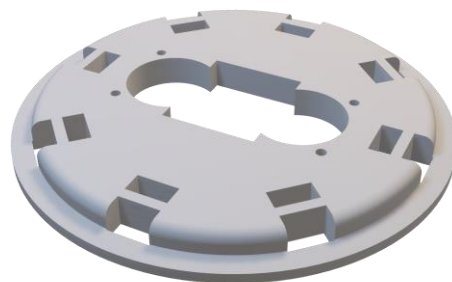


Figure 57. Pipe cap to hold ultrasonic sensor in place

The way this sensor works is, the pipe is vertically placed in the water, inside of which is the polystyrene part floating (this is optional, since the water surface reflects sound on its own, but it was added to make it more reliable). The ultrasonic emitter and receiver are placed on top of the pipe, held in place by the 3D printed part. The sensor emits an ultrasonic soundwave, which is reflected on the polystyrene and detected then by the sensor. The time it takes the sound to go and come is

proportional to the distance of the polystyrene, which is higher when the water level is lower. Calibrating the output of the sensor, it is possible to get a pretty precise measurement of the water level.

The ultrasonic is controlled by the ESP32, which runs ESPHome. Its configuration file has been edited to include this sensor, trigger and echo pins are specified, as well as the median filter and other configurations. The lambda filter converts the output from the sensor (distance to the polystyrene) to the water volume, which involves the three dimensions of the tank (1x1x1m) and a small correction factor due to the placement of the sensor, its height and the one of the polystyrene layers.

```
- platform: ultrasonic
  trigger_pin: 17
  echo_pin: 21
  name: "Water level"
  update_interval: 1s
  accuracy_decimals: 2
  filters:
    - lambda: return (1 - x + 0.03) * 1 * 1;
    - median:
        window_size: 20
        send_every: 10
  unit_of_measurement: "m^3"
```

Figure 58. ESPHome configuration for water level sensor

The final setup of the sensor is:



Figure 59. Water level sensor mounted

7 Monitoring

As a centralized data point, Home Assistant can also be used to monitor the state of all the systems. Indeed, some of the sensors that are currently included are just for monitoring purposes since they are not used in automation (the greenhouse temperature sensor for instance). See Figure for a screenshot of how the main dashboard looks like.

At the beginning of the project, the dashboard was only accessible from the same network as all the automation equipment, which limited the usefulness of the monitoring. In order to make it accessible from a remote location and because of the complexity of dealing with a 4G connection, extra steps needed to be taken.

7.1 Configuring Home Assistant

In the following section, all the configuration needed to make Home Assistant remotely accessible will be exposed.

7.1.1 Encryption - HTTPS

First of all, HTTPS was enabled for the HA server, which is an encrypted version of HTTP, the protocol used to serve webpages worldwide. The secure version requires some extra configuration, but is currently used by most webpages, and is crucial when sending login credentials and sensitive information over the Internet.

An encryption certificate is obtained for free by *Let's encrypt*, it certifies that our domain name is actually owned by us, and so it is directly tied to our DNS, `hygrowteamintern.duckdns.org`. This certificate needs periodical renewal since its validity is only 3 months. This could be done by the DuckDNS plugin, but it has been disabled for reasons that will be later discussed. Instead, the Let's Encrypt plugin for Home Assistant is being used, it has been installed and marked to start on boot, every time it runs it checks whether renewal is needed and gets it if necessary. The plugin configuration is as follows:

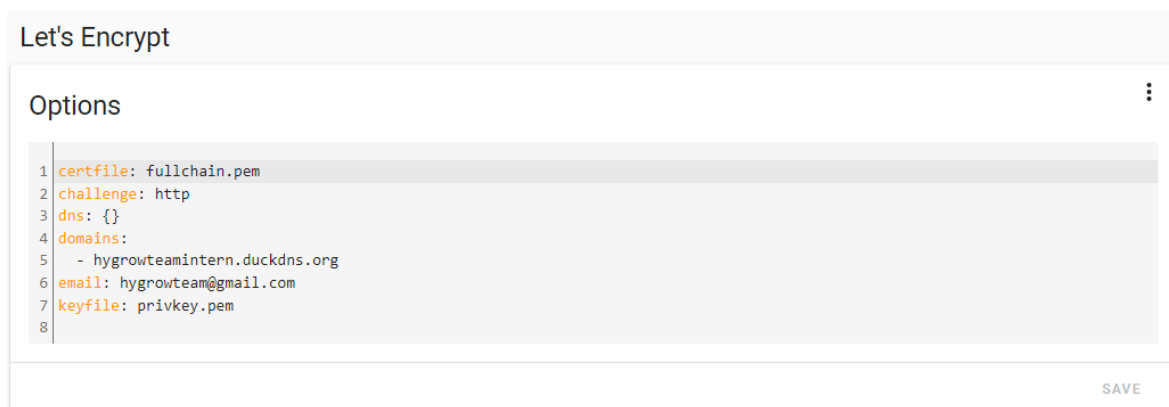


Figure 60. Let's encrypt plugin configuration

7.1.2 The problem with incoming connections

After HTTPS has been enabled, the server is ready to securely accept incoming connections from remote locations, but we first needed to expose the RPi to the internet, which came with a few problems.

The IPv4 protocol is supported by all the devices currently connected to the internet, which uses a limited number of IP addresses to identify devices online. Since these IPs are so scarce, usually many devices share the same address under a NAT, which handles all the addresses conversions. This saves lots of IP addresses, but also limits functionality, since devices under NAT can no longer be accessed from the Internet. This is usually not a problem, because most connections are outbound, meaning that the user starts the connection to a remote server, not the other way. In the case inbound connections are needed, their shared IP is used to address all devices and the NAT is configured to forward the incoming connections to only one of the devices (commonly known as opening ports).

Usually, Internet Service Providers assign a single address per contract, so that users can configure the NAT in their home routers. This is changing lately with ISPs running out of addresses, many have started using what is called carrier-grade NAT, which means that there are multiple clients sharing the same address, and 2 (or more) NATs are needed. In this case, users can no longer accept incoming connections because the carrier NATs are not configurable by them, which is the case with the 4G connection of the greenhouse. Sometimes, ISPs offer getting out of CG-NAT for a certain price, which tends to be an expensive solution.

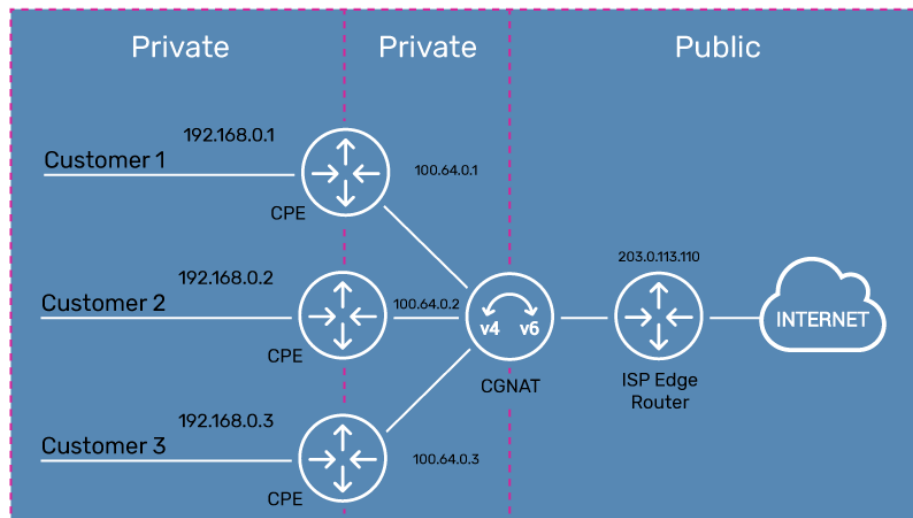


Figure 61. Diagram showing CG-NAT customers sharing public IP address (Huawei, s.f.)

7.1.3 Remote monitoring using IPv6

IPv6 is a newer version of the IP protocol which accepts many more different addresses, making them widely available, and NATs not necessary. Luckily enough, the 4G connection on the greenhouse supports IPv6, thus the RPi can be directly addressed from the Internet using its unique IPv6 address, but not its IPv4 address which is shared with other devices and customers. The problem with IPv6 is

that its adoption is rather limited, and many connections don't currently support it now. For this reason, using IPv6 will only work when trying to access the dashboard from a limited number of networks. For instance, Finland uses IPv6 in 40% of the connections, but Spain does it only in 3% of them (Google, s.f.).

The IPv6 address of the RPi may be changed periodically by the ISP, to avoid having to keep track of it manually, DuckDNS service is used. They provide us with a free domain name, which is much easier to remember, always pointing to the right IP. The Raspberry Pi needs to send DuckDNS its own IP for the domain to work. The DuckDNS plugin cannot be used for this because it does not support IPv6. Not only this, but we also need to make sure that the DNS does not contain the IPv4 address, otherwise devices may try to use that one instead, which will not work for the given reasons. The following script retrieves the IP using the *ipify* service, and sends it to the *duckDNS* API to get it updated:

```
domain=hygrowteamintern.duckdns.org

token=token removed from the report for security reasons

ipv6addr=$(curl -s https://api6.ipify.org)

curl -s "https://www.duckdns.org/update?domains=$domain&token=$token&ipv6=$ipv6addr"
```

Figure 62. Script to update DuckDNS record

This script has been configured inside *configuration.yaml* using the *shell_command* directive. Then, this script can be used in automation rules, making it run every 5 minutes:

Triggers

Triggers are what starts the processing of an automation rule. It is possible to specify multiple triggers for the same rule. Once a trigger starts, Home Assistant will validate the conditions, if any, and call the action.

[Learn more about triggers](#)

Trigger type

Time Pattern

Hours

Minutes

/5

Seconds

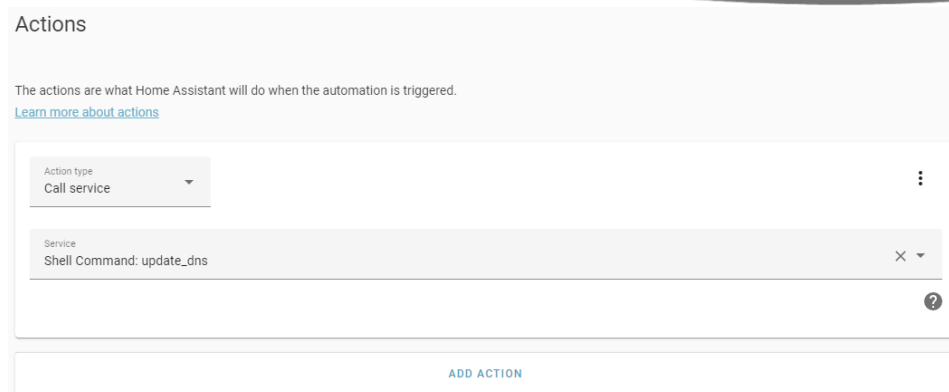


Figure 63. DuckDNS automation configuration

This ensures that, when the IP is reassigned (this usually occurs when rebooting the router but may also happen from time to time when the 4G connection is lost and recovered, for instance), the maximum time with the greenhouse unreachable will be 5 minutes.

7.2 Remote server for web and proxy

In order to host our webpage, which servers as an entry point to direct users to the monitor, the dashboard, the Telegram bot endpoint and other services in the future, we have configured a server running in *Amazon Web Services*.

The server is running 24/7, it can be accessed using SSH in ubuntu@3.72.218.189 (any SSH client can be used, using *Putty* if running Windows is an option) authentication is done using a client-side key. This key will not be included in this document due to security reasons, to get access to the server request access to the AWS manager, which is *Hans Lindén* at the time of writing this.

7.2.1 Novia DNS

To make it easier to access our website, we requested Novia for a subdomain under the root one. We were given hygrow.novia.fi, whose A record will point to our AWS server, this means that any client trying to connect to our domain, will get back the IP, so that the website or necessary content can be retrieved. This is easy to check by doing a ping:

```
H:\>ping hygrow.novia.fi
Pinging hygrow.novia.fi [3.72.218.189] with 32 bytes of data:
Reply from 3.72.218.189: bytes=32 time=32ms TTL=44
```

Figure 64. Ping to our domain name

This means that the DNS is properly configured, the computer got the IP, tried to connect to it, and got a response from the server in 32ms.

7.2.2 Encryption – HTTPS certificate

The same way as connections to the Home Assistant need to be encrypted, the AWS server needs a certificate to protect connections to it from external attacks. The same *Let's encrypt* free certificate will be used, they provide a client called *Certbot* which can get this certificate automatically, and renew it before it expires, for which it interacts with the chosen HTTP server *Nginx* (further about this in the next section). This document will not include further details into the configuration of *Certbot*, since [their webpage](#) provides a pretty detailed guide.

7.2.3 Nginx web server

Nginx is one of the most used web server software, serving over 20% of the websites on the Internet (Wikipedia, 2022). It is mainly used because of its lightness, its flexibility working as a reverse proxy, load balancer and other, and it being open source. Nginx was installed in the server using Ubuntu's integrated package manager (APT), which also keeps it updated.

Nginx will load any configuration files located under `/etc/nginx/sites-enabled/`, it is usually a good practice, and the way it is currently set up, to place them under `/etc/nginx/sites-available/` and then create a symbolic link to reference them from the previous folder. A single configuration file is being used, called *HApprox.conf*, which contains the following:

```
map $http_upgrade $connection_upgrade {
    default upgrade;
    ''      close;
}

server {

    server_name hygrow.novia.fi;

    location = / {
        proxy_buffering off;
        if ($is_args = "") {
            return 307 /web/;
        }

        resolver 1.1.1.1;
        proxy_pass https://hygrow.novia.fi/lovelace/$is_args$args;
    }

    location /web/ {
        alias /home/ubuntu/hygrow/hygrow-webpage/;
    }

    location /public_api/monitor/ {
        proxy_pass http://localhost:3000/;
        proxy_buffering off;
    }

    location /public_api/telegram/ {
        proxy_pass http://localhost:4000/;
        proxy_buffering off;
    }
}
```

```
location / {

    # To make dynamic DNS update
    resolver 1.1.1.1 8.8.8.8;
    set $backend "https://hygrowteamintern.duckdns.org:8123";
    proxy_pass $backend;

    proxy_buffering off;

    proxy_set_header Host $host;
    proxy_http_version 1.1;
    proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection $connection_upgrade;

}

listen 443 ssl; # managed by Certbot
ssl_certificate /etc/letsencrypt/live/hygrow.novia.fi/fullchain.pem; #
managed by Certbot
ssl_certificate_key /etc/letsencrypt/live/hygrow.novia.fi/privkey.pem; #
managed by Certbot
include /etc/letsencrypt/options-ssl-nginx.conf; # managed by Certbot
ssl_dhparam /etc/letsencrypt/ssl-dhparams.pem; # managed by Certbot
}

server {
    if ($host = hygrow.novia.fi) {
        return 301 https://$host$request_uri;
    } # managed by Certbot

    server_name hygrow.novia.fi;
    listen 80;
    return 404; # managed by Certbot
}
```

Figure 65. Nginx web server configuration

The following parts of the document will explain in more detail the different sections of this configuration file. One can modify this file using *nano*, a command-line file editor, or any other tool of its wish, and apply changes by restarting the nginx service by doing *sudo service nginx restart*. Keep in mind that any malformed configuration file will prevent the server from running, leaving the webpage unavailable.

7.2.3.1 HTTP redirect

The configuration defines two servers running inside of nginx, the last one, which is used by certbot when obtaining or renewing certificates, listens on port 80 (default non-secure HTTP). It is configured to redirect requests (code 301) to the equivalent version of the webpage but using HTTPS (many browsers will attempt connecting to the insecure version of the webpage first for compatibility reasons, but it is not acceptable to send any information over that connection).

If a client connects to the server using a different host than *hygrow.novia.fi*, such as when connecting using the IP address directly, the server will respond a 404 – not found error.

7.2.3.2 HTTPS server

A second server is defined, this one listens on port 443 (default HTTPS port). It is configured to use SSL with the certificate, key, and some other configuration provided by the certbot client. Actual requests will be handled by this server, all of them using encryption. Since there are many different services provided by this server, we use different *location* directives to specify each of them.

7.2.3.3 Location = /

When a client first attempts to access our website, it will do so with no specified path. In this case, we redirect the user to */web/*, which is where the main welcoming webpage is located. This is done this way not to interfere with the Home Assistant web structure. This location only matches request with an exact path of */*, not if the path contains something else afterwards, in which case it is handled by other location directives, this is configured using the *=* symbol.

Sometimes HA may use the root */location* as well, and it expects this to be redirected to the HA server, not our webpage. For this reason, if any arguments are passed with the request, we proxy the request to */lovelace/*, which is the entry point for the HA interface.

Note that HA expects to be the only service accessible at a certain domain, which is the reason why certain workarounds like this need to be made. Another more elegant option would be to use different subdomains for each, such as *hygrow.novia.fi* for the main website, and *homeassistant.hygrow.novia.fi* for the HA interface. This would require additional domain names to be provided by the university IT department, and so it was discarded. Another option is to place HA under a different path, like *hygrow.novia.fi/homeassistant/*, but this is not supported by HA, and [never will](#) according to its developers.

7.2.3.4 Location /web/

This location serves all the static content of the main public webpage. Webpage files, which are HTML for structure, CSS for style, JS for scripts and images are located inside of */home/ubuntu/hygrow/hygrow-webpage/*. Nginx is told to look for files to satisfy requests in that folder, if the file does not exist it will throw a 404 error, if no file is specified (just accessing */web/*) it will default to *index.html*.

7.2.3.5 Location `/public_api/monitor/`

This location proxies the requests coming from the monitor frontend (the webpage that shows all the information regarding to the current state of all the tests) to the backend (the actual script that runs the tests). This script runs on the same server, it will be detailed in a future section, and only listens to requests coming from the same machine. This way, it is not necessary to setup HTTPS for every service that is running, instead, nginx takes all the requests and proxies them over localhost, which is inherently secure. This also allows us to use the same external port (443) for every service.

7.2.3.6 Location `/public_api/telegram/`

Configured in a similar fashion as the monitor proxy, this handles request coming from the Telegram servers (such as when a user sends a message to the bot). The telegram bot endpoint is also written in a script that will be covered in a coming section.

This mechanism, called *webhook*, in which the Telegram servers send a request to our server when needed, replaces the previous one, *polling*, in which it is our server which checks periodically for changes. Even though it is slightly more complex to configure, it is the recommended way to go by Telegram, since it is faster (messages are received instantly), produces less junk traffic (no need to constantly check for changes, which would need to be done at least once per second) and is more stable.

7.2.3.7 Location `/`

This is the last location directive; it will handle all other requests that are not done by any of the previous sections. Nginx will proxy all requests to the Home Assistant instance in the greenhouse. To do so, the `proxy_pass` directive tells the server to send requests to <https://hygrowteamintern.duckdns.org:8123>, note that, since the AWS instance has IPv6 connectivity, it has no problem connecting directly to the greenhouse, clients could also use the direct connection, but they need to ensure they have compatibility first. This is a functional diagram of the current setup, and how it is able to fix the potential problems that IPv4-only clients could face:

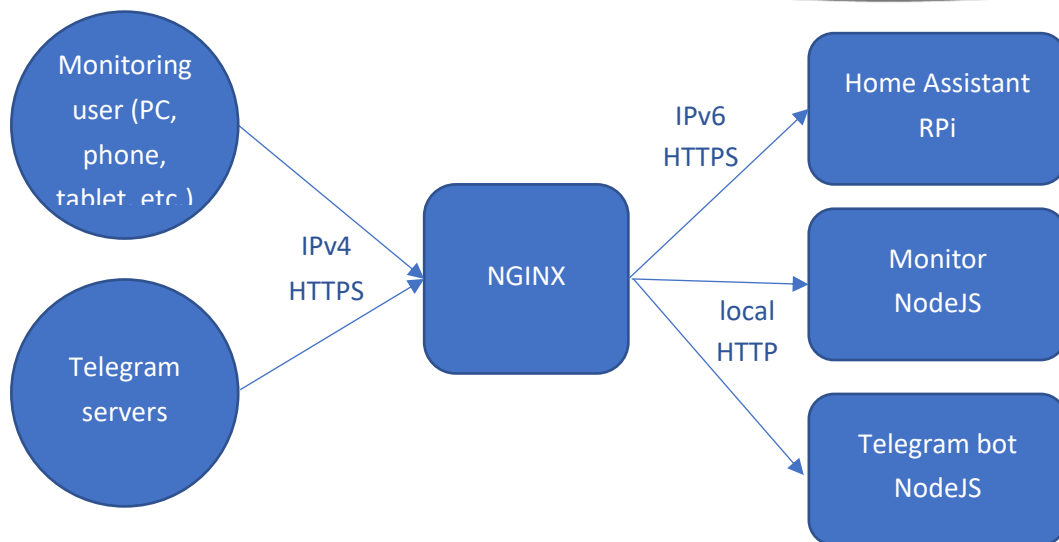


Figure 66. Proxy server connection diagram

The necessity of this proxy may disappear in the future once IPv6 compatibility is included in most connections, but even then, using the proxy will still work and provides a centralized access point to all the Hygrow information. A couple of the downsides of this approach is the increased complexity of the system and the latency that it adds to every request.

The `proxy_pass` is using the `$backend` variable, instead of defining directly the address. This is a workaround to make the DDNS work properly. If done otherwise, nginx would obtain the IP address associated to our DuckDNS domain when starting and would cache it for its whole execution. This means that, if the IP was to change for whatever reason (which is safe to assume will happen at least once a week), nginx would attempt to use the old one until restarted, which would leave the dashboard inaccessible. By using this workaround, and defining the `resolver` directive, which tells the server what DNS servers to fetch the IP from (we are using CloudFlare and Google servers here), nginx will retrieve the IP for every connection, so that it can get the updated IP immediately.

The rest of the directives are necessary for Home Assistant to work with a reverse proxy (it needs to understand that the proxy server is not the actual client, but rather an intermediary. For HA to accept connections from a reverse proxy, it also needs to be configured in such way. This is done in the root `configuration.yaml` by enabling the HTTP header `x_forwarded_for` and whitelisting the AWS IPv6 address:

```

http:~
  - use_x_forwarded_for: true~
  - trusted_proxies:~
    - - 2a05:d014:23a:ca00:b2c3:8252:d307:db60 # AWS proxy server~

```

Figure 67. HA HTTP proxy configuration.

7.3 GitHub

For the following sections, code was created, which has been uploaded to GitHub, a platform for software version control, issue tracking, automated testing, and many others. This ensures that the

code is always available in the future even if the AWS instance, which is where it is running, is removed or lost. An account was created using our Gmail profile, credentials are not included here for security reasons.

Only two private repositories are being used for now, but future teams are encouraged to upload any other new suitable content to the platform. Version control allows users to see a history of changes, which especially useful when dealing with software but also handy for other cases. Here is a look of the profile:

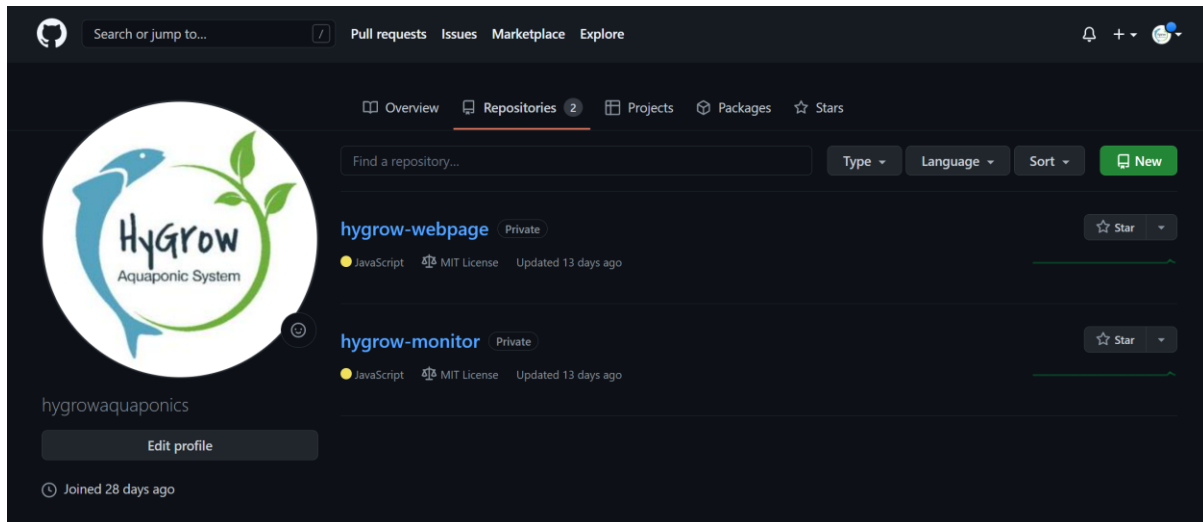


Figure 68. Hygrow GitHub profile

7.4 Monitor

The monitor is a piece of software that was created to replace the previously existing Telegram bot notification system. Even though it pretty much served its purpose, it had many flaws in it. To start with, its implementation using Node-RED left very little space for customization. Chat ids (used by Telegram to refer to a certain chat) were hard coded, which is a really bad practice and made it impossible to talk to the bot from any other chat (this is also the reason why the bot was not working in the beginning of the semester, Telegram had changed the id for the group that was being used). Also, if the automation stopped working for whatever reason (loss of internet connectivity, electricity shortage, misconfiguration of the RPi or any other) the users were not being informed about it.

In the new setup, the monitor is running externally, in the AWS instance right now, and remotely checks that measurements are within reasonable values. It has been written in JavaScript, which is really suitable for networking since it follows an asynchronous request-callback scheme. The code runs on NodeJS, an open-source JavaScript runtime designed to run backend servers.

The values are gotten from the Home Assistant instance using its official REST API, no additional configuration was needed here other than specifying the internet address of it. A node library called *homeassistant* makes the interface even easier ([see the library in NPM](#)).

Tests all inherit from a base class, which provides a plain interface, so that each of them can be called, returning either pass and a small explanation if the test was successful, or failed and the reasons for it as a string if it was not. Three types of tests were created, though it is really easy to create new types if needed:

- Connectivity test: Attempts to create a TCP connection to the given address and port combination, it passes if connection can be established. It is used to check whether the greenhouse is online.
- SSL expiry test: It makes an HTTPS request to the given URL, it fails if the certificate is not valid. This could occur for different reasons, like if using an invalid authority (really unlikely to occur unless *Let's encrypt* gets banned) or the certificate being expired or close to expiration. *Certbot* is configured to renew certificates everywhere they are used, so no manual maintenance should be needed, but this adds a layer of protection since it will detect if the renewal did not work before making it inaccessible. Currently, two instances of this test are created, one for the certificate being used in the frontend, and another one for Home Assistant in the RPi.
- Home Assistant entity test: It gets the state of a given entity, and checks whether its values are available and within expected. It also checks last time the value changed, so that it is possible to detect sensors being offline (most values will change once in a while even if they are not supposed to because of noise). Everything is configurable and optional for this test, so that one entity may have a minimum value but no maximum, it could have both or even a fixed value (like being dry for a leakage sensor). The test returns a summary of its value and last date of change if successful, or the reason why it failed if not. Note that this test will also fail if HA is not reachable.

All these tests are programmed in a file called *monitorTests.js*. They are instantiated in another file called *testList.js*, which is the file that should be modified if any change is wanted, like adding/removing a sensor or changing its acceptable values. This is the current definition of the tests:

```

1 // Hygrow Aquaponics
2
3 // Test list
4 // EPS Project, NOVIA UAS, Vaasa, Finland, Spring 2022
5 // Author: Pablo Criado Albillos
6
7 const HomeAssistant = require('homeassistant');
8 const config = require('./config').config.monitor;
9 const tests = require('./monitorTests');
10 const HAEntityTest = tests.HAEntityTest;
11 const SSLEntityTest = tests.SSLEntityTest;
12 const ConnectivityTest = tests.ConnectivityTest;
13
14 const hass = new HomeAssistant({
15   host: config.HAHost,
16   port: config.HAPort,
17   token: config.HAToken,
18   ignoreCert: false
19 });
20
21 exports.tests = [
22   new ConnectivityTest('Greenhouse connectivity', { url: 'tcp://hygrowteamintern.duckdns.org:8123' }),
23
24   new SSLEntityTest('hygrow.novia.fi SSL certificate', { url: 'https://hygrow.novia.fi', min_duration: 10 }),
25   new SSLEntityTest('hygrowteamintern.duckdns.org SSL certificate', { url: 'https://hygrowteamintern.duckdns.org:8123', min_duration: 10 }),
26
27   new HAEntityTest('Greenhouse temperature', { type: 'sensor', entity: 'outside_temperature', unit: '°C', hass: hass, max_age: 60 * 60 * 12, max_value: 50, min_value: -30 }),
28   new HAEntityTest('Water pH', { type: 'sensor', entity: 'ph_value', unit: 'pH', hass: hass, max_age: 60 * 60 * 12, max_value: 7.5, min_value: 6.5 }),
29   new HAEntityTest('Water level', { type: 'sensor', entity: 'water_level', unit: 'm³', hass: hass, max_age: 60 * 60 * 18, max_value: 0.9, min_value: 0.4 }),
30   new HAEntityTest('Tank leakage', { type: 'binary_sensor', entity: 'lumi_lumi_sensor_wleak_aql_b819f106_ias_zone', hass: hass, required_value: 'off' }),
31   new HAEntityTest('Tank leakage sensor battery', { type: 'sensor', entity: 'lumi_lumi_sensor_wleak_aql_b819f106_power', unit: '%', hass: hass, min_value: 15 }),
32   new HAEntityTest('Water tank air temperature', { type: 'sensor', entity: 'lumi_lumi_weather_73930807_temperature', unit: '°C', hass: hass, max_age: 60 * 60 * 12, max_value: 30, min_value: -20 }),
33   new HAEntityTest('Water tank air sensor battery', { type: 'sensor', entity: 'lumi_lumi_weather_73930807_power', unit: '%', hass: hass, min_value: 15 }),
34   new HAEntityTest('Aquarium leakage', { type: 'binary_sensor', entity: 'aquarium_leak_sensor_ias_zone', hass: hass, required_value: 'off' }),
35   new HAEntityTest('Aquarium leakage sensor battery', { type: 'sensor', entity: 'aquarium_leak_sensor_power', unit: '%', hass: hass, min_value: 15 }),
36   // new HAEntityTest('Mock test sensor', { type: 'sensor', entity: 'mock_sensor', hass: hass, max_age: undefined, max_value: 30, min_value: -20, unit: '°C' }),
37 ];

```

Figure 69. Monitor tests definition

The monitor runs all tests periodically and exposes the results of them publicly through a REST API, this API is only reachable locally, but is exposed to the internet as discussed in 7.2.3.5. This API is used by the monitor frontend, which is accessible from the main webpages and shows the current state of all tests:

Hygrow Aquaponics

Test	Result	Description
Greenhouse connectivity	PASS	Reachable
hygrow.novia.fi SSL certificate	PASS	Valid, expires in 57 days
hygrowteamintern.duckdns.org SSL certificate	PASS	Valid, expires in 60 days
Greenhouse temperature	PASS	12.9°C at 11/05/2022, 11:39:50
Water pH	FAIL	Measurement (6.26pH) is less than 6.5pH, system or sensor may be malfunctioning
Water level	PASS	0.47m³ at 11/05/2022, 11:39:32
Tank leakage	PASS	off at 11/05/2022, 09:49:44
Tank leakage sensor battery	PASS	77% at 11/05/2022, 11:15:07
Water tank air temperature	PASS	16.8°C at 11/05/2022, 11:26:20
Water tank air sensor battery	PASS	62% at 11/05/2022, 09:49:44
Aquarium leakage	PASS	off at 11/05/2022, 09:49:44
Aquarium leakage sensor battery	PASS	94% at 11/05/2022, 09:49:44
Monitor	PASS	Running

Tests last run 49 second(s) ago. Dates shown in Helsinki time.

You can subscribe to our Telegram bot to get notifications: @hygrow_aquaponics_bot

Back

Figure 70. Monitor frontend

The API is also used by the Telegram backend, which will be discussed in the next section, to send notifications to Telegram clients or summaries when asked for a summary.

7.5 Telegram bot

The telegram bot backend is also written in JavaScript and runs on NodeJS as well. It receives commands from the Telegram servers whenever a message is sent to it, this was explained before in section 7.2.3.6, as now a webhook is being used instead of polling. An HTTP server is configured on port 4000, the Nginx server will handle encrypted requests coming from the internet and forward them to this script. Whenever a request is received, it is passed to a library being used ([see Node.js Telegram Bot API in NPM](#)) which decodes it, and calls any of our callback functions. The library also handles sending back messages.

The bot will only reply to certain messages, and it will do so to anyone that asks it, so that now it is possible to have private conversations with it, or add it to a group as well. The possible commands are the following:

- */start*: This command is used by Telegram when talking to a bot for the first time, which makes it introduce itself and explain its functionality.
- */report*: This message is kept for backwards compatibility, it was used before to fetch the status of the automation system. Since the dashboard is now accessible from the internet, and it provides a better interface than sending all the information through messages, the user is reminded about this, and a link to the website is sent.
- */tests*: It sends back the status of all the tests, which the script has cached before. A summary is sent first, with the date of the last execution and the number of tests passed/failed, followed by an explanation for each of them.
- */addtopush*: This adds the current chat to the list of push chats, meaning that notifications when a test fails will be sent to it. The script saves the chat id of everyone that have asked for it (which could be persons but also groups or channels) in a file so that changes are preserved when restarting the server.
- */removefrompush*: This removes the current chat from the list of push chats, and saves changes to disk as well.

To keep track of the state of all tests, the bot will periodically fetch the status of all of them using the monitor API. If any change is detected, such as a new test failing, a notification is sent to all the chats in the push list. Reminders are also sent periodically if tests keep failing, or when one passes again after being failing for a while. This behavior can be configured in a file called *config.js*, this is the current one (tokens remove for security reasons):

```

1  exports.config = {
2    telegram: {
3      botToken: '*****',
4      webUrl: 'hygrow.novia.fi',
5      dashboardUrl: 'hygrow.novia.fi/lovelace',
6      monitorUrl: 'http://localhost:3000/test_results',
7      webHookUrl: 'https://hygrow.novia.fi/public_api/telegram/',
8      reminderPeriod: 1000 * 60 * 60 * 4, // 4 hours
9      monitorPollPeriod: 1000 * 10, // 10 seconds
10     warningThreshold: 1000 * 60 * 5, // 5 minutes
11   },
12   monitor: {
13     testPeriod: 1000 * 60, // 1 minute
14     HAHost: 'https://hygrow.novia.fi',
15     HAPort: 443,
16     HAToken: '*****',
17   }
18 }

```

Figure 71. Telegram bot backend configuration file

7.6 PM2

To run both the monitor and telegram bot backend, pm2 is used. It is a production process manager for NodeJS, which automatically starts on boot, and runs any script that it is configured to in the background. It has also tools to view the current state of processes, to do this type *pm2 status*, which will return a list of all running scripts:

```
ubuntu@ip-172-26-11-69:~$ pm2 status
```

id	name	namespace	version	mode	pid	uptime	□	status	cpu	mem	user	watching
0	bot	default	1.0.0	fork	33299	13D	2	online	0%	33.5mb	ubuntu	disabled
1	monitor	default	1.0.0	fork	33311	13D	20	online	0%	55.8mb	ubuntu	disabled


```
Module
```

id	module	version	pid	status	□	cpu	mem	user
2	pm2-logrotate	2.7.0	112362	online	0	0%	32.8mb	ubuntu

Figure 72. PM2 running monitor and Telegram bot on AWS

The *logrotate* module is used to keeps logs clean so that the server does not run out of disk space. Other useful commands can be *pm2 start "script_name"* or *pm2 stop "script_name"*.

8 Greenhouse

8.1 Plans of the Greenhouse

Another of our client's wishes was to redesign the Greenhouse in order to free up space in the interior and to insert a new tank for the plants. To do this, we drew up plans of the Greenhouse in order to get an idea of the future layout we would give to the Greenhouse.

To do this, we have taken note of all the elements present in the Greenhouse (fish tank, rack for plants, etc.). In addition to these elements, we removed the parts that were taking up space. We dismantled them and stored them in a place at the university.

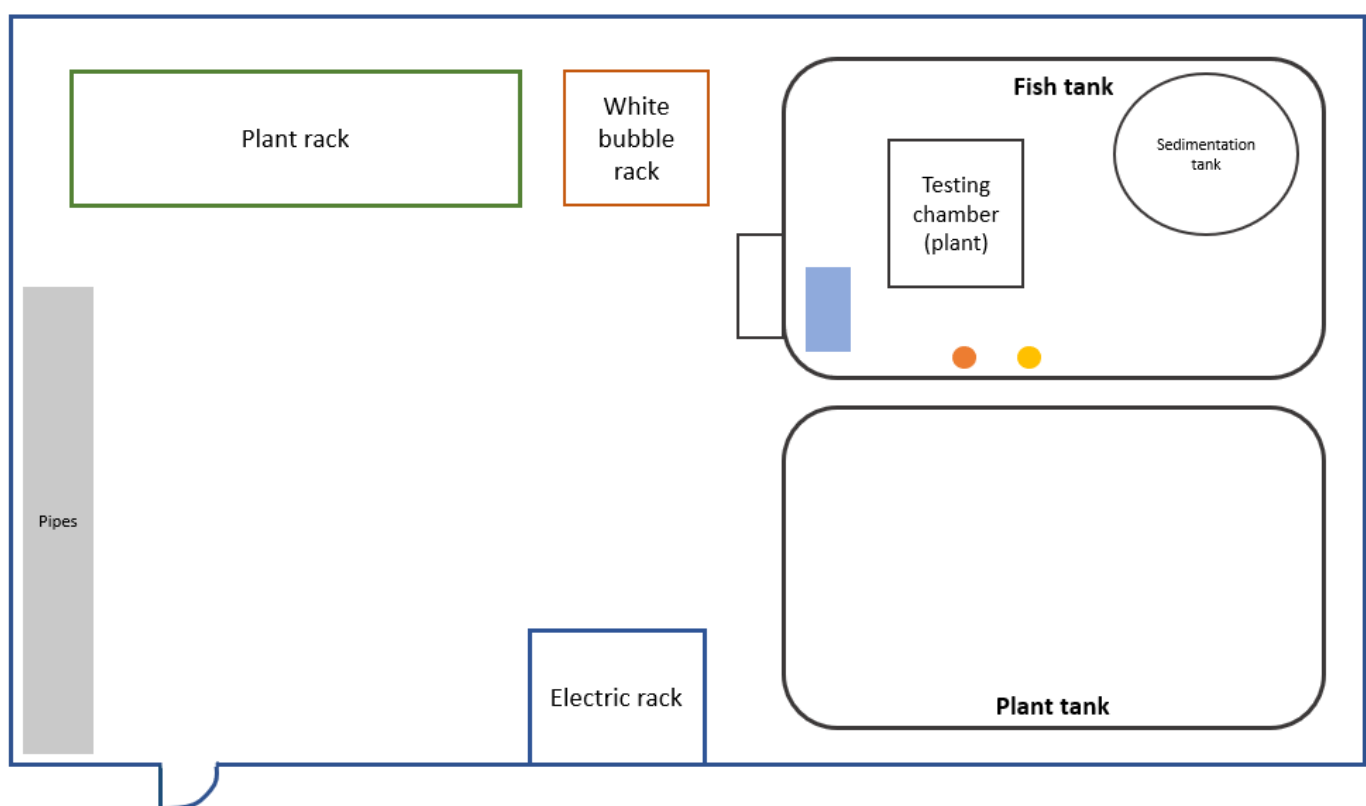


Figure 73. The previous layout of the Greenhouse

Thus, by freeing up space inside the Greenhouse, we were able to fit it out as planned. However, we did not fix the tank with the white balls on the plant tank as we had specified on the plan.

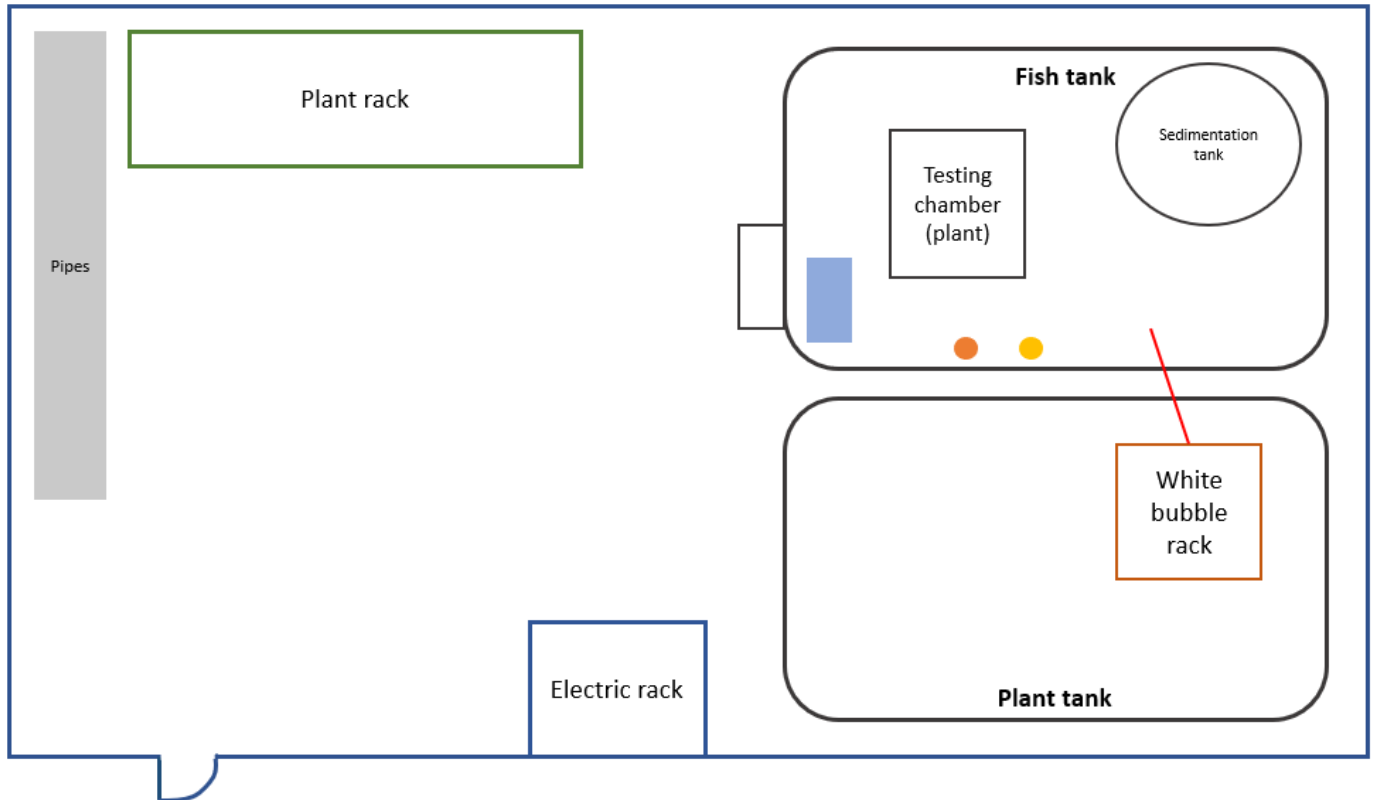


Figure 74. The future layout of the Greenhouse

To have a better view of how the tank with the white bubbles will be arranged, we made a front view of the plan.

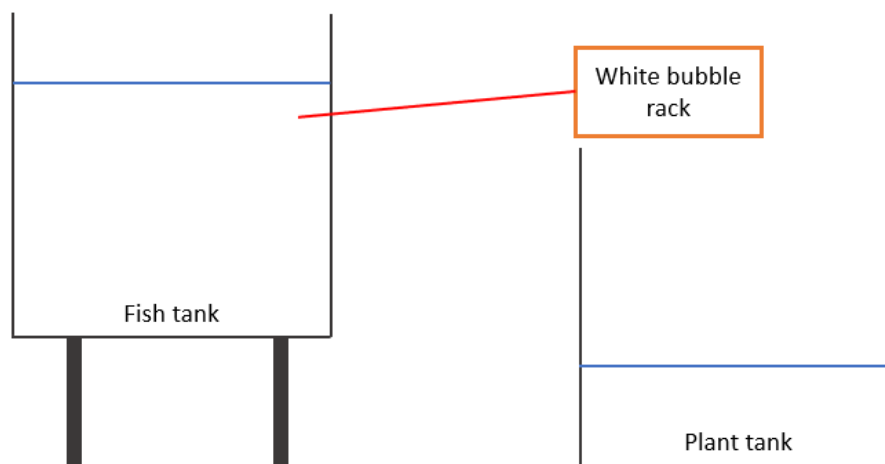


Figure 75. Vision in front of the tanks

In general, we find on these diagrams the following elements:

- - Air pump
- - Temperature sensor
- - Water level sensor
- - White bubble tank: at the top of the plant tank with a pipe to link the fish tank and the plant tank
 - At the bottom of the plant tank, we can find a pump to move the water.
 - In each tank, we can find a water level sensor.

8.2 Insulation of the Greenhouse

The next step when working on the greenhouse was adjusting further isolation on the inside. This was done to make sure the greenhouse is as airtight as possible. To isolate the insight there were blue plastic sheets that could be used. These sheets were really big and hard to handle at some times. This made the isolation harder. Also, there was chosen to not isolate the roof from the insight, because the roof was already isolated with double layered plastic plates. This was probably not the weakest point. There was also a gap into the outer isolation on the left side of the greenhouse. This gap was made because the wind is really heavy on that sight. This way the wind could break the plastic. This obviously needed to be repaired, because otherwise the isolation wouldn't work.

Another problem faced during isolating of the inside was that the fish tank was fixed to the ground. This way it was impossible to isolate the wall behind the tank with 1 person. When there was 1 extra person available the isolating with the sheets was completed.



Figure 76. Example of the blue sheets on the inside for insulation.



Figure 77. Example of the reinforcements to the inside isolation.

But this was not everything for the isolation on the inside. To make sure the isolation would stay in place, some planks were used to reinforce the sheets. This way the wind couldn't break the sheets that easily. The nails on their own were not really strong, especially because the blue isolation sheets rip easily.

8.3 Heating experiment

After the isolation was completed, a heating experiment was started. This experiment was used to see how good the greenhouse was isolated and which temperature there could be held inside. This is important because if the greenhouse is good isolated, it can run as well in the winter. This is especially tricky in the Finnish winter, since the greenhouse needs to be really good insulated.

The experiment was done with a heater which could work on different power settings, it was set to 2kW, but the observed power was slightly less. It is worth noting that the high power of the heater, combined with the high temperatures and direct sunlight made the power cord overheat, triggering its thermal protection. For this reason, it was necessary to unwind it from the base, which improves its thermal dissipation and allowed us to run the experiment properly.

Important for this experiment was that the data should be recorded while the sun is set. This way the sun wouldn't heat the greenhouse and influence the experiment. Also, it is important that the temperature inside and outside can be measured precisely. A Zigbee temperature sensor was added in the outside to achieve this, we first run a trial time with both sensors inside to verify that the measurements were consistent, which turned out to be the case.

The first attempt of the experiment didn't end up well. The data retrieved was not reliable because the temperature inside the greenhouse was not uniform. This was fixed by adding a fan which forces an air flow inside of the greenhouse, however, this could also increase the heat losses since it makes more air go out of the greenhouse and increases conductivity with the walls of it. This solution fixed the problem with the temperature differences inside, and proper data could be recorded. Here are the results of the experiment:

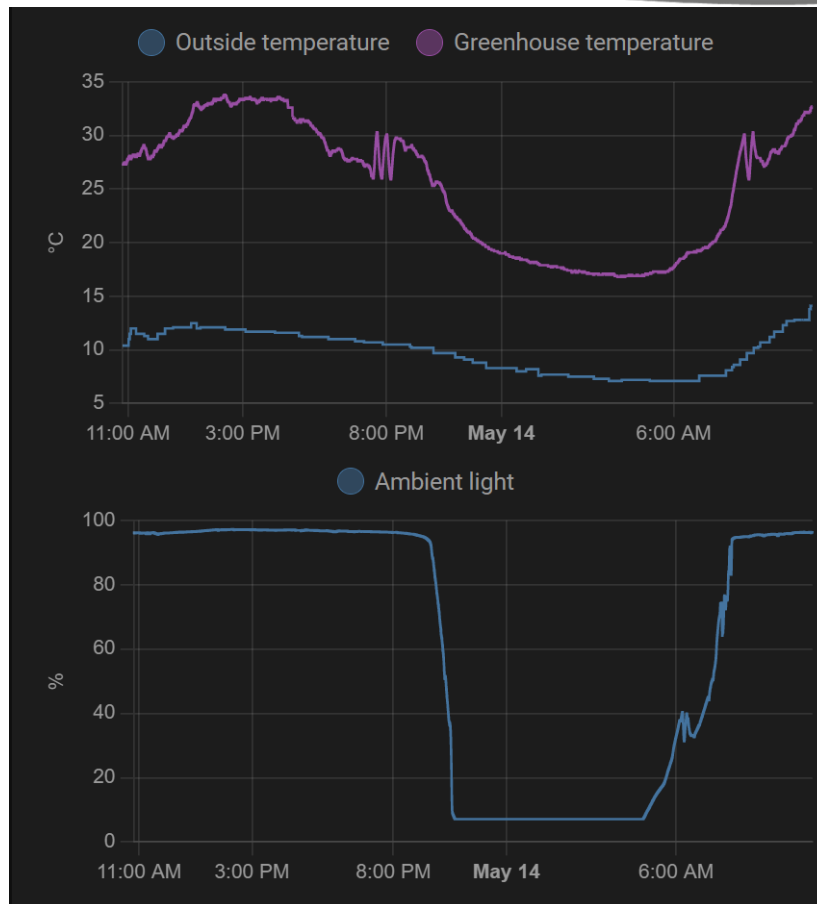


Figure 78. Heating experiment temperature chart.

As stated before, the data is only relevant when no ambient light. The greenhouse showed an average temperature differential of $\sim 10^{\circ}\text{C}$ throughout the night. We also need to analyze power usage:

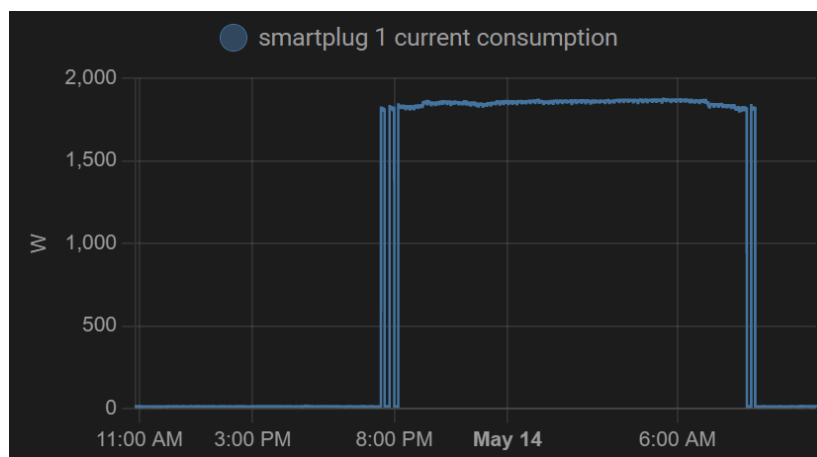


Figure 79. Heating experiment power usage.

The power consumed by the greenhouse during that time was approximately 1850W, which includes the power of the heater, as well as the consumption of the rest of the appliances, which also

contribute to the heating of the air inside. We may conclude that the thermal loss of the greenhouse with its current insulation is **185W/°C**.

To illustrate this, a typical winter night in Vaasa may be around -10°C average, if it was required to keep some comfortable 15°C inside, 4625W would be needed. This consumption could last for around 10h every night, making a consumption of 46kWh per night, and electricity in Finland was paid at an average of 0.1767 €/kWh in 2021 (Statista, 2021). This would produce a cost of 8€ every night, or 244€ monthly, which is drastically unacceptable, not to mention environmental impact of this huge consumption. This all assumes that sun is enough to keep the greenhouse warm during the day, which is not the case either, especially during the winter, making it even less feasible.

If future teams need to keep the greenhouse heated, they are advised to implement better insulation, removing all air gaps, and maybe using opaque polystyrene. Even if that means losing sun heating, it is not too relevant during the winter.

8.4 Door construction

After the reinforcing the inside isolation was done. The next problem was that the door wasn't closing well. Because of the air gap between the door and the frame, the greenhouse wasn't airtight. There were two options the project group came up with. Option one was refitting the door and hoping that it would stay in place. The second option was constructing a new lighter door, so the door wouldn't move that much in the frame. The team chose to do the second option, constructing a new door. The new door needed to be lighter than the previous one, so it would not change place that much. Therefore, there was chosen for a not completely solid wooden door, but only a wooden frame. In between there is a layer of poly styrene and to cover the isolating sheets were used.

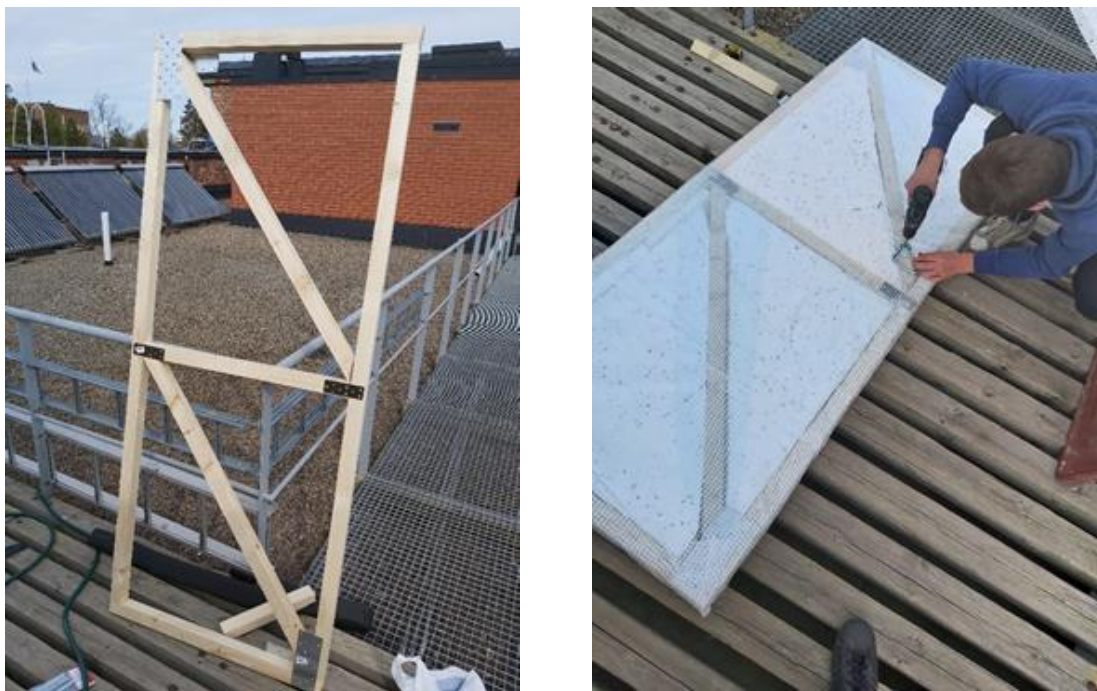


Figure 80. Construction of the door. Frame on the left and almost end product on the right.

Also, the locking system was improved. At forehand there was a constructed wooden part which held the door closed. This system was not optimal and therefore the idea to buy a new handle with closing lock was adjusted. This way the door had a proper closing system.



Figure 81. Comparison of the old (left) and new (right) locking system.

Further there were some other things that were improved on the door. The first thing is that there is now a handle on the inside. Closing the door from the inside was really hard at first because there was no handle on the inside. Reusing the handle from the old door was the easiest and cheapest option for this.



Figure 82. End result of the door.

8.5 Testing the growth of plants

In addition to the fish classifier, we also conducted tests with plants. The objective of these tests was to visualize the growth of plants with a part under LEDs and another part without artificial light. Thus, we want to observe the differences in growth, speed of growth but also to see if we can save energy. Indeed, if our tests show us that the plants grow without major problems and without a contribution of artificial light, we will be able to place the selected plants in the tank envisaged for this purpose without light.

In our trial, we have two distinct sides that can be seen by the numbers 1 and 2. Side number 1 corresponds to the side that had no light above the cultivated plants. Side number 2 was in contact with light.

In our test, we find four different plants associated with the colors/marks of the markers made on the support.

- Green → Strawberry
- Orange → Salad bowl
- Blue → Matador
- Black → Lollo rosa

To compare whether the use of light has a strong influence on plant growth, we took pictures at different time intervals to visualize their evolution. The observation is divided into two parts.

First, we notice that the lettuce plants that were not illuminated are bushier than the others. Nevertheless, we also notice that the lettuces that have been illuminated have a color that is similar to the colors of the lettuces that we can eat and that we find in the shops. In addition to that, we also notice that the raspberries had a better growth in the light.



Figure 83. Evolution of our tests with plant growth

We notice that the raspberries are not growing fast. These last ones, which attracted the attention of the customer to make them grow in the aquaponics system, showed us that they are not adapted to be in an aquaponics system. Indeed, we can see on the pictures that their growth is long and that they have difficulties to germinate.



Figure 84. Focus on raspberries growth

Following these different observations, we can conclude that raspberries are not adapted to the aquaponics system. Indeed, we need plants which have a fast growth to feed the fish. Raspberries do not allow us to reach this objective. Thus, we are obliged to remove them from our system because they do not meet the requirements, we have for life in an aquaponics system.

In general, we can also ask ourselves which plants are most suitable and used in aquaponics. To answer this question, we continued our research and found information related to the project.

The most suitable plants for aquaponics are leafy vegetables (lettuce, spinach, chard, leek, watercress...) and herbs (basil, chives, parsley, mint, coriander...). Then you can start with tomatoes, cucumbers, squash, aubergines...

8.5.1 Electricity consumption

Each experiment was run using a full spectrum 5W LED strip. Our measurements corroborate that, indeed, the consumption of them was slightly under that number, even considering the inefficiencies of the power adapters. They were running 8h and 16h respectively, and electricity in Finland was paid at an average of 0.1767 €/kWh in 2021 (Statista, 2021). This allows us to do some basic calculations:

	Power (W)	Daily running time (h)	Monthly consumption (kWh)	Estimated monthly cost (€)
Side 1	5	8	1.2	0.21
Side 2	5	16	2.4	0.42

Table 1. Lighting cost estimations.

These cost estimates clearly show that using artificial lighting to grow plants is feasible, even if more light may be required when plants are bigger, the cost of it is way lower than that of the heating.

9 Business model

9.1 Theoretical research on the aquaponic environment

Our project also includes a theoretical part aiming at answering the questions of our customer. Indeed, the latter wishes to have information on the types of fish and molluscs that can be raised in an aquaponic environment. To answer this, we have done some research. We also made conclusions on the contribution that certain species could have in an aquaponic environment.

In addition to our research, we also conducted experiments in the Greenhouse with different types of plants. Indeed, we also have to take into consideration the plants present in the Greenhouse that will be present in the new tank that we have inserted in the Greenhouse. Now there are two tanks; one for fish and the other will be mainly used for growing plants in the aquaponics system.

9.1.1 Fresh-water mussels



Figure 85. Fresh-water mussels

The freshwater mussel is a mollusc whose diameter can reach up to 25 cm. Its presence indicates the good health of the aquatic environment in question. Indeed, this mussel does not support pollution. It also tends to have a slight preference for acidic waters although its shell is calcareous.

Another positive point is that it is a mussel edible by humans. Indeed, the objective of the project is to be able to realize an autonomous Greenhouse. Indeed, the Greenhouse must be able to provide plants which will make it possible to nourish the various species of fish present inside the tank, and finally, these fish, will be sold then consumed by the Man. Nevertheless, it is not recommended to eat this mussel because it has no taste for humans, and we must not forget that it plays an important role in the filtration of the water of the aquaponic system.

Nutritional needs:

It feeds on micro-organisms such as plankton and bacteria. In simple terms, it is a self-contained filter that feeds on anything in the water that is very small. They feed on algae.

Temperature:

It prefers cold water, at around 13°C but the water can reach temperatures ranging from 10°C to 25°C.

pH: between 6 and 8

Oxygen and CO₂ production:

When the freshwater mussel breathes, it exchanges carbon dioxide for oxygen. This mussel thus makes it possible to oxygenate the water. Indeed, when the water is not oxygenated or little, the mussel opens its shell naturally to increase the quantity of water which it can treat.

Life cycle:

The freshwater pearl mussel is one of the longest living species in northern rivers, where it can live for 200 years or even longer. Its life cycle depends on the two fish species, salmon and trout. Its reproduction takes place in summer from June to August. The eggs laid by the female hatch after four weeks and give birth to microscopic larvae. Their growth takes place on the gills of a fish for a period that can vary from fifteen days to two months. This depends mainly on the living conditions of the fish. The larvae will then leave the fish in question once they have reached their final form. Nevertheless, the size of the larvae is small since it is 2 to 3 millimeters.

Breeding:

Concerning the breeding of this mussel, it lives towards the bottom of the water and buries itself half in mud, sand or gravel. Moreover, freshwater mussels depend on fish for their survival. Indeed, the mussel larva must find a fish that will serve as a host and allow it to continue its development. The mussel lives on its host for a period of ten to thirty days.

Diseases:

One of the first diseases that the freshwater mussel can get is called "glochidiasis". It is not a real disease because the small mussel that starts its growth without being on the gills of a fish, does not suffer from it. However, it is necessary to be careful with freshwater mussels because they can be a vector of diseases. In addition to that, it does not support chemical treatments against diseases that affect fish. These mussels can therefore inhibit diseases when they perform water treatment. In addition to this, the mussel can be a security against bacterial infections.

Things to managed:

One of the main characteristics of this mussel is that it destroys the environment where it lives. In addition to this, it is preferable to raise it in an aquarium that is very large and also deep. In fact, the minimum volume to raise this type of mussels is about 150 liters. In addition to that, another important characteristic is to offer to the mussel an aquarium of a certain depth and a minimum width of 80 centimeters of frontage.

Can they replace plants in an aquaponics system? Purifiers of water? How do they handle nitrites/nitrates?

The freshwater mussel has a large water filtration capacity (up to 50 L per day). It is therefore very useful when we raise several fish in an aquaponic environment because it allows to clarify the water. The mussel cannot replace the plants in their totality. They have the same role, but it is still important to have natural plants in an aquarium to purify and oxygenate the water naturally. In addition to this essential role, the plants present in an aquaponic environment allow to reproduce the natural environment of the fish. Fish need to live in an environment that is as close as possible to their natural environment. This is why it is difficult to replace plants with freshwater mussels. They have the same objective concerning water and its purification, but the presence of plants brings something essential for the fish to reproduce their living conditions.

(Center of biological diversity, sd) (Wikipedia, 2018) (Xerces, sd)

9.1.2 *Macrobrachium rosenbergii*



Figure 86. *Macrobrachium rosenbergii*

This species is found in freshwater and is therefore adapted to live in an aquaponic system, a system consisting of freshwater. The *Macrobrachium rosenbergii* is a night feeder. It is a large crustacean that starts out measuring only a few millimetres and grows to 10-15 cm in nine months and its adult size can approach 30 cm. It is therefore the largest shrimp raised in an aquaponics environment.

The aim of the current aquaponic system is to be able to sort the fish in order to sell them and market them. So, we need fish that are edible for humans and fish that feed mainly on vegetation. Indeed,

the aquaponic system that we have on the roof of Technobotnia University is a system in which fish and plants are raised together. The latter are used to feed the fish that live in the tank provided for this purpose.

With all these criteria, we realize that the *Macrobrachium rosenbergii* is perfectly adapted to all these criteria since it is omnivorous (so it can feed on plants) and also edible. All of this is in line with our current system and the aim of the final project.

Nutritional needs:

It feeds mainly at night and is omnivorous (algae, aquatic insect larvae, flesh of dead fish or other crustaceans...).

Temperature:

This type of crustacean lives in fresh-water and it's mainly used in the aquaculture. It lives in quite warm water between 23°C and 28°C.

pH: acid pH between 7,5 and 8,5

Dissolved oxygen: between 3 and 7 ppm

Light:

This species of shrimp tries to find shade when in a natural, shallow environment to avoid sunlight during the day.

Nitrite/nitrate/ammonia production: Nitrate levels should remain below 50mg/L.

Life cycle:

This species is reproduced in captivity, usually in an aquarium. This crustacean lives mainly at night and prefers the seabed. The life span of this species is relatively short (about 2 years).

Breeding:

It is best to keep them in a large aquarium of about 600 L. This crustacean lives in groups of at least 3. It reproduces in brackish water.

Problems:

This type of crustacean is not suitable for living in a small aquarium due to its large size (up to 20 or 30 cm).

Diseases:

Whitish muscle disease of *Macrobrachium rosenbergii* (whitish disease) is a disease of the giant freshwater shrimp *Macrobrachium rosenbergii* caused by a nodavirus that has not yet been officially classified. There is no treatment for the disease and prevention is the only way to keep the disease out of shrimp farms. Breeders and post-larval individuals should be screened regularly.

The major problem of diseases affecting *Macrobrachium rosenbergii* generally occurs because of poor incoming water quality, poor husbandry, over-stocking, poor hygiene and the inadequate procedures. We can find a number of diseases such as viruses, bacteria or fungi.

Things to managed:

To keep the water clean and unpolluted, plan for a monthly renewal of 20% to 30% of the water volume because it often lives in very turbid waters. In addition to this, we also need to know that with this species of shrimp, there is a very specific and particular hierarchy among the males. This has an impact on their sociability, cohabitation with several males and their behavior. Omnivorous by nature, this giant shrimp eats everything but is also carnivorous with small crustaceans (it is quite capable of cannibalism towards young shrimp of its own species).

Another important piece of information about *Macrobrachium rosenbergii* is that they should be fed regularly, 2 to 6 times a day, making sure to give a larger amount of food in the evening hours.

Compatibility with plants, fruit, strawberries?

Knowing that this species of shrimp is compatible with plant food, we conclude that it can feed on plants present in the Greenhouse. Indeed, we will have plants such as lettuce or raspberries available. These large shrimps can therefore feed on this, but we must ensure that they have meat-based food from time to time. This is justified because they are omnivores, so in order to develop properly, they need a relatively varied diet.

(Wenger, 2013) (Nandlal & Pickering, 2006) (Schäfer, 2018)

9.1.3 Freshwater shrimp



Figure 87. Freshwater shrimp

Using freshwater shrimp for aquaponics (crustaceans' fish) can be a beneficial addition to our fish tank. Shrimps are popularly used since they are not only tasty seafood items but are also an efficient cleaning crew for the tank. More importantly, caring for shrimps in aquaponics systems does not require much effort since they are hardy and easy to raise.

Nutritional need:

Shrimp will eat anything they are opportunistic omnivores, which means they will eat both plants and animals, whether they are dead or alive. As they grow, they will also eat algae, dead and living plants, worms (even decaying worms), fish, snails and even other dead shrimps.

Temperature:

Shrimp can thrive between 13,5 and 40 C, but the optimum temperature is between 25 and 31 C

pH:

It should be good to maintain a pH level between 7,8 and 8,5. As always, we need to use a quality pH meter.

Oxygen requirement:

Dissolved oxygen concentration should be maintained to at least 3 mg/L. Anything higher is ideal, while anything below that is stressful for the shrimp.

Shrimp diseases:

Shrimp in all life stages is susceptible to different viruses, which will cause mortality or stunted growth. Fungal infections usually affect the larval stage, while bacterial infections may result when the

crustaceans live in unfavorable conditions. One important thing to note is that the viral disease called white spot disease is considered the most serious threat to shrimp culture.

Can they be mixed with other breeds?

An important thing to note when caring for shrimp in aquaponics systems is that they are not suitable tankmates for aggressive fish types that have big mouths. One perfect and popular fish species to mix with shrimp is tilapia. Many aquaponic gardeners have successfully raised these species together without problems. One common practice is to put a 0.25-inch wire mesh between the fish and shrimp.

Shrimp water filters:

Filters for shrimp tanks need to come with a sponge as juveniles can easily be sucked into the filter.

(Madore, 2021)

9.1.4 Freshwater crabs



Figure 88. Freshwater crabs

Typical appearance and behavior:

Active, nocturnal (active at night), Crabs can climb almost any surface, including airline tubing and intake tubes; secure holes in aquarium hood to keep crabs inside. All crabs molt as they grow; a crab lying on their back may be molting. Some will feed on their old shell for calcium, some species of crabs live in low brackish levels and will benefit from freshwater salt.

Temperature: Freshwater crab can live 22 - 25°C

Nutrient requirement:

A well-balanced freshwater crab diet consists of:

Flakes, freeze-dried, sinking pellets and wafers or frozen food. Feed a variety of food to ensure complete nutrition. We need to remember when we are feeding the crab:

- Feed a small amount once daily
- Make sure a small amount of food reaches the bottom of the tank; if unsure, drop sinking pellets into the tank at night; alternate protein and algae-based pellets.
- Thaw frozen food before feeding and Crabs need calcium to grow their shells.

pH: The pH balance of the water should be around 6.2 to 7.2.

Life cycle and breeding:

Fertilized female fiddler crabs carry hundreds to thousands of eggs under their abdomen. When the eggs are ready, the mother goes into the water and allows the eggs to hatch into microscopic free-swimming larvae. At the end of the final larval stage, the larvae Molt into immature crabs. The amount of time spent as a swimming larva (hatching to true crab stage) varies among species, but ranges from a few weeks to a few months. As they grow larger and turn into adults, the secondary-sexual characteristics (e.g., the asymmetric claws) begin to develop. Adult crabs' mate and the cycle start over. They can live like 3 years or more. it depends also on species.

Can they be mixed with other breeds?

No more than one crab per square foot is recommended because Male crabs can be very territorial. Do not keep with fish who will harass or eat them, such as certain cichlids. Carbs may catch and eat small fish, dwarf African frogs, snails and other tank mates who live or sleep on the bottom of the tank

Depending on the crab species, may be compatible with:

- Gourami
- Tetras
- Hatchers
- Swordtails
- Rainbowfish
- Danios
- Barbs

Diseases:

Some health issues can be:

- The Loss of appendage. An iodine supplement can help with the molting process.
- Body or surface erosion. Test and treat water immediately; maintain proper diet. Little is known about diseases that affect crabs; if environmental conditions and food are adequate; crabs are resistant to disease. (Wikipedia, 2021)

9.1.5 Langouste



Figure 89. Langouste

Langouste is also call by spiny lobsters. The langouste is found in almost all warm seas, including the Caribbean and the Mediterranean Sea, but are particularly common in Australasia, where they are referred to commonly as crayfish or sea crayfish and in South Africa.

Temperature: The best temperature is around 15° and 20° C.

Nutritional needs:

Lobsters are generally regarded as opportunistic. The adult lobster diet consists of slow-moving and sedentary animals including bivalves, chiton, gastropods, small molluscs, worms, sea urchins, small crustaceans and echinoderms but larva have different diets than the adult lobsters. Larvae eat most of time other planktonic marine organisms such as chaetognaths, euphausiids, fish larvae, medusae, or ctenophore. They are considered omnivorous, as there are records of them occasionally eating vegetation. Langouste use their front legs to bring food close to them and then crush it with their mandibles.

Life cycle:

The life history of the spiny lobster consists of four phases: planktonic phyllosome larvae, swimming post larval pueruli, benthic juvenile, and adult. Each has a distinctive behavior and habitat that is characteristic of that stage in the life cycle.

A female releases 500,000 to 2 million eggs, once or twice each season. When eggs are ready to be fertilized, a female will scratch open the spermatophore deposited by the male, resulting in external fertilization (some consider this a form of delayed fertilization). A female carries fertilized eggs on her pleopods for about a month, until they are ready to hatch. Increasing embryonic pheromone levels indicate readiness to hatch and trigger more vigorous pleopodal pumping by the female, helping the eggs to hatch. The maturity estimation is two years. Approximately the animal can live 12-20 years in the wild; age is typically estimated by size.

Diseases:

Spiny lobsters have few pathogens, parasites, and symbionts and all these are common in the post larvae or juvenile' stages. The langouste can have some following diseases:

- Nicothoidae – parasitic copepods
- Digenetic trematode infections
- Microsporidi
- Peritrich ciliates
- Oomycetes
- Vibriosis
- White spot syndrome virus

(Pitcher, sd) (Houlihan & Wood, sd) (J.D.Shields, 2011)

9.1.6 *Larges species*

Guppies are lively and sociable community fish that get along well with other species similar in size and temperament, but the counsel is to make sure checking possible compatibility issues between different species, prepared to avoid any problems with the fish. keeping guppies with other large fish that may mistake them for food. Predatory fish and aggressive fish are also a bad match for guppies.

9.2 Theoretical research on the underwater plants

For our project we are using guppy fish. Therefore, we oriented our research towards plants that can live underwater with guppy fish.

9.2.1 Guppy grass

This is part of the best live plants for guppies.



Figure 90. Guppy grass

There are several reasons behind it:

First of all, guppy grass gets its name because it is very famous among guppy breeders as it provides a lot of hiding places for guppy fry.

Secondly, it is very easy to care for a fast-growing plant. Also, it can tolerate a wide range of water parameters. It is highly recommended when we are the beginning, to start with this plant Guppy grass also helps to maintain good water quality by removing harmful toxins like ammonia, nitrate, nitrites and other heavy metals. It is also a very good oxygenating plant. For proper and faster growth of guppy grass, we should supplement it with some liquid fertilizers. We should add these fertilizers after the weekly water change. guppy grass can tolerate a wide range of water parameters.

The ideal temperature for the proper growth of guppy grass is from 10 to 30 degrees Celsius. The pH tolerance range is from 6 to 7.

9.2.2 Green foxtail



Figure 91. Green foxtail

Green foxtail is a fast-growing, tall plant that can grow as tall as 24 inches. This plant is very easy to care for which makes it ideal for beginners. The ideal temperature for the proper growth of green foxtail is from 22 to 27 degrees Celsius. The pH tolerance is 6,5 to 7,5. Fertilizers are not necessary for green foxtail.

9.2.3 Java Moss



Figure 92. Java Moss

Java Moss is a very popular plant in the aquarium hobby. This plant is very easy to care for and it can tolerate a wide range of water parameters which makes it ideal for beginners. This is a fast-growing plant, and it provides a lot of hiding place and protection to the fish. It is also a very good oxygenating plant, and it improves the overall water quality of the aquarium. Java Moss is very undemanding when it comes to light.

This plant can grow in almost any lighting condition. We can keep this plant in very low light, or we can keep it in very highlighting condition as well.

The ideal temperature for the proper growth of Java Moss is from 15 to 30 degrees Celsius. The pH tolerance range is from 5 to 8 and the light demands is between low to medium.

9.2.4 Hornwort



Figure 93. Hornwort

It is a very hardy plant and can tolerate a wide range of water parameters which makes it ideal for beginners. This plant can tolerate temperature as low as 15 degrees Celsius, so we can easily keep it in a cold-water aquarium. Hornwort can grow very quickly and can provide a lot of hiding places for guppy.

The ideal temperature for the proper growth of hornwort is from 15 to 30 degrees C. The pH tolerance is between from 6 and 7,5.

9.2.5 Anacharis



Figure 94. Anacharis

When we plant multiple anacharis plants close to each other, it creates lots of hiding places for guppy fish. we can plant anacharis in the substrate of our aquarium or can also let it float in the aquarium. Some people have noticed that it grows faster when we keep it floating in the aquarium as it will get lighter than when it is planted.

The ideal temperature for the proper growth of this plant is from 15,5 to 27 degrees Celsius. The pH is between from 6,5 to 7,5.

9.2.6 *Salvinia natans*



Figure 95. *Salvinia natans*

Salvinia is known as floating water moss, water butterfly, floating fern or floating moss. This plant is very easy to care for and it can tolerate a wide range of water parameters, so we can easily keep it in a cold-water aquarium.

This plant also helps to remove toxic heavy metals from the water. So, it will work as a very good water purifier for our aquarium and improve water quality. The ideal temperature for the proper growth of this plant is between 12 and 30 degrees Celsius. The pH tolerance range is between 6 and 8.

9.2.7 *Red root floater*



Figure 96. *Red root floater*

Red root floater is native to South America, and it is now reported spreading in Florida. Red root floater is easy to maintain and a fast-growing floating aquarium plant. This plant does well in low water flow aquariums, and it cannot tolerate a high level of surface agitation. This is a very good oxygenating plant, and it propagates very quickly. This plant is ideal for open-top aquariums because it does not do well in too much humidity or moisture.

The ideal temperature for the proper growth of red root floater is from 21 to 27 degrees Celsius and the pH tolerance is between 6,5 and 7,5.

9.2.8 Duckweed



Figure 97. Duckweed

Duckweed is a fast-growing floating aquarium plant. This is very easy to maintain plant which makes it a good option for beginners. People have mixed opinions about this plant. On the other hand, this plant does have some benefit if we keep its growth under control. This plant can grow very quickly so we need to keep an eye on it and regularly trim it. If maintained well, this plant can work as a very good water purifier for the aquarium. It can tolerate temperatures as low as 5.5 degrees to up to 32 degrees Celsius. (Tom, 2019) (Healthy Betta, sd)

9.2.9 Proportion between underwater plants and guppy fish

Plants are a critical part of the guppy aquarium or tank. Besides simulating their natural habitat, plants help clean and oxygenate the water. More importantly, the plants which have many leaves provide guppy fry with places to hide and cover from adults until they grow up.

9.2.10 Commercial value for underwater plants

Many underwater plants are using in ornamental and restoration way:

- Ornamental plant:

Submerged plants are used to enhance the beauty of lakes and ponds as well as provide habitat for other forms. Ordinary types of freshwater aquarium plants include Anacharis, Camboma, Aponogeton, Anubias, Cryptocoryne, Egeria densa, Echinodorus and Vallisneria can be use as ornamental plants. Generally, they are available bunched, bare-root or potted, depending on growth characteristics and value.

- Restoration plant:

In our research specially, we find some plants which are compatible with the guppy, but we have others Aquatic plants that can be used by ecologists, engineers, and other professionals to restore damaged wetlands or create water storage filtration areas for stormwater runoff. Restoration species include both freshwater but also saltwater plants such as cattail, bulrush, native lily, pickerelweed, seagrasses, and mangroves. (Services, s.d.)

9.2.11 Humans benefit

Some underwater plant can be a vegetable or fruits that human can beneficiate. Strawberry and lettuce(can be place into a shallow bowl with water) are some kinds of plant that is recommend as underwater plant which can be eat by human.

9.2.12 Fish benefit

Herbivore and omnivore fish can be feed with some kind of underwater plants but we recommend to try out some criteria as the plant must look good and have the capacity to grow fast enough to keep up with the efforts of the fish to nibble them down and able to be sufficient for them .It not all the plants that the fish can eat, They must be study enough to be edible, have a nutriment required for the fish and soft enough to provide food for the fish. Probably, the best ways to provide growing plants for fish to eat is to attach them to driftwood or rocks using cotton thread. As these plants grow, they naturally adhere to the rock or driftwood and provide excellent food plants for fish. In the addition, the plants need to receive enough the right amount of light to produce nutriment necessary for the fish. Most plants benefit from full-spectrum lighting, and they generally need about 10 to 12 hours of light per day. (Editorial, 2011)

9.3 Cheapest/robust webcams for cold and damp environments

Another question from our client was what type of camera is suitable for use in a wet/marine environment with sometimes low temperatures. We need to take pictures of the fish in the aquarium. For this purpose, we turned to a technique using a camera placed on a pipe that keeps the camera system out of the water. This solution is not unique and we have multiple options for successfully taking pictures of the fish to determine their size and then sorting them.

So, to answer one of our client's questions, we did some research on cameras that could be used to meet our needs.

Following various research, we were able to find information to give advice and ideas to our client. We looked for robust cameras for wet and cold environments, but also for cameras that could meet these criteria at a relatively low price.

Robust camera ideal for cold environments:

To begin with, we were interested in the temperature criterion. In Finland, and especially during the Finnish winter, temperatures are rather low and can reach as low as -25°C.

We found a camera that can withstand such extreme temperatures. On top of that, the camera has features that keep it running smoothly during periods of frost or rain. The name of this camera is the PTZ camera. Nevertheless, the price of this camera is high because it has some very special features. For this reason, we do not want to use this type of camera for our system because the technology is too advanced to be able to take simple photos of fish in a greenhouse. (fast, s.d.)



Figure 98. PTZ camera

Rechargeable and cold-resistant camera:

In order to withstand the extreme temperatures that can occur in Finland, especially during the winter, it is best to use a camera with heated lenses. These prevent frost from forming on the camera lens. In addition to this feature, they also prevent condensation from forming inside the camera housing and around the camera.

We have found several camera references that can withstand extreme temperatures down to -30°C. The one that is most suitable for such temperatures is the SkylinkNet HD 720p IP camera. This camera is capable of recording 30 pictures per second. In addition to this, it has a night vision function that allows clear images despite the darkness of the environment. All of these features are perfect for us because they are features that are unique to the Nordic country in winter.

We can also find this camera at affordable prices around 50 euros and 100 euros. This is not negligible and is a positive point for this camera that is perfectly adapted to extreme conditions. (Controls, s.d.)



Figure 99. SkylinkNet HD 720p IP camera

Waterproof camera:

Continuing our search, we found another style of camera, the AEE S80 camera. This one can be put in water up to 1 meter deep, has Wi-Fi, 4 hours of autonomy and a G-sensor. It is capable of taking a large number of photos per second. This is a positive point to be able to take a picture of the fish and to have a sufficient number of pictures to select the best one to measure the size of the fish.

The price of this camera is more affordable than the camera presented above. Indeed, we can find one with a price oscillating from 120 euros to 250 euros. (PNJ, s.d.) (advance, s.d.)



Figure 100. AEE S80 camera

GoPro camera:

This type of camera is also suitable for taking pictures and videos underwater with a very high resolution. This camera is also called an "action camera" because it has the ability to take sharp pictures even though things are moving.

One of the recommended GoPro cameras is the GoPro HERO9. It is capable of taking pictures with near perfect resolution as well as videos. It is able to automatically select the best picture taken with

the best resolution. This can also be an advantage for our project. We want to get a picture of fish that can pass quickly in front of the camera. This option will be very useful for the realization and optimization of our system to save time and to have a perfect image processing. In addition to these features, it is also a waterproof camera up to several meters deep and robust. Nevertheless, the price of this camera remains high even if it remains proportional to the services that the camera can deliver. (GoPro, s.d.)



Figure 101. GoPro HERO9

Waterproof sports camera:

We can also find waterproof sports cameras at low prices (between 50 euros and 100 euros). This is a camera that also has a good image resolution. It is a wireless camera that is easily adjustable and adaptable. It can be waterproof up to a depth of 30 meters, which is sufficient for us as the aquarium or even the tank in the Greenhouse is not that deep. (Gembird, s.d.)



Figure 102. Waterproof sports camera

10 General research on aquaponics plants

To begin with, there are plants that do not need sunlight to grow. However, these are mainly houseplants and not plants made for the aquaponics environment.

In an aquaponics system, the production of food in this system is the main objective, but it is essential to understand that this is based mainly on the balanced management of the ecosystem that lives in the aquaponics unit and which is made up of three main organisms: fish, plants and bacteria. Indeed, the fish farm must provide the necessary nutrients for the development of the plants and in return, the plants must filter the water sufficiently for the fish. Finally, it is important to ensure that the volume of water introduced into the system is well oxygenated and sufficient so that the circulation of water from one tank to another can be done properly.

Concerning the plants that can be adapted to this cycle for aquaponics, we find vegetables which are the main production of an aquaponic system. Indeed, it is essential to cultivate healthy and resistant plants. In addition to this, it is recommended to cultivate plants with a fast growth.

Lettuce is a very suitable plant for the first crops/plantations in an aquaponics system. It is a plant that requires the least amount of nutrients. The plants can be grown with a slightly closer spacing between each plant than in a soil-based system, since in an aquaponics system the plants are not competing for water and nutrients. However, it is necessary that the plants have enough space to reach their adult size and avoid competition for light with neighboring plants.

Thus, in order to grow lettuce, a minimum amount of light will be needed to make it profitable on the energy production that it will require to obtain a good growth. Indeed, the plants grow faster with the presence of a minimum of light. We have also seen during our tests in the Greenhouse tank that the presence of light allows the lettuce to grow properly and to have the color that will allow it to be sold at a correct price. (Food and Agriculture Organization of the United Nations, 2022)

11 Budget

For our project we had a budget of 325,77 euros. During the semester we spent 204,2 euros. This was useful to buy the fish, the different maintenance products for the aquarium, the materials and products useful to build the fish classifier (plexiglass, pipes, etc. ...) and also for the construction of the door (screws, hinges, handles, wood ...).

To conclude, we can say that we have respected the budget at our disposal because there is a total of 121.57 euros not used. Indeed, we did not realize big expenses because we had at our disposal an important number of materials already bought during the previous semesters and the parts of our fish classifier were realized in 3D printing with the resources available at the university.

Global budget (euros)	325,77
Remaining budget (euros)	121,57
Budget spent (euros)	204,2
Tie ribs	6
Leaking box	12
Fish and accessories for the aquarium	43
Plexiglass + accessories for the Greenhouse	49,2
Door accessories	94

Table 2. Expenditure on the project during the semester

12 Conclusion

In this part the conclusion on different parts of the project is given. There are three parts where the team can make a conclusion about: the remote monitoring system, the fish classifier and the greenhouse.

12.1 The remote monitoring system

The first topic that will be discussed in the conclusion is the remote monitoring system of the greenhouse. In general, there can be said that this part was successful. There were a lot of improvements made to the monitoring system. First the monitoring system is now better organized. This helps in getting a good overview of what is happening inside the greenhouse. Also, there are now less external addons in the system and more home assistant components integrated in the system. The last improved part is that the monitoring system is now remotely accessible. This makes it way more practical and usable.

All these things combined make that the monitoring system is now more robust and gives a better overview of the greenhouse.

12.2 The fish classifier

The next topic is about the fish classifier. For the fish classifier the conclusion is a little more complex. This is because the classifier is not finished, but there were a lot of things learned about this concept.

The first thing to talk about the classifier is the current state of the classifier. The camera setup is working, and it is possible to capture pictures of the fish with the camera. With these pictures it is possible to do the calculations of how big the fish itself is. When combined with a servo motor and a door this could lead to a proper classifying concept.

Anyway, this concept didn't work because the fish's behavior made it really hard. The fish were not really behaving as we hoped it would. This is because when they went into the pipe to classify them, they stayed in the pipe instead of going through directly. This is not a weird thing, since fish like places to shelter. The pipe can be seen as a nice place to shelter and so was the place they stayed in.

Another problem the team faced was the death of most of the fish. This made the observing of the fish behavior a little harder since a fish on his own is probably behaving other than when he has a lot of other fish around. The fish probably died because there was too much chlorine in the aquarium water. This happened because the water needed to be refreshed a lot and when refreshed there came chlorine in the aquarium as well. This chlorine could be removed by using a supplement for the water. This meant that one fish could survive.

12.3 The greenhouse

In the last part the greenhouse conclusion is discussed. For the greenhouse were no clear goals other than repairing what is broken and try to improve the place. This made it hard to say if the team succeeded in this task. Anyway, there can probably be said that the team did a good job on this one since the customer was happy with the improvements made.

The first improvement was the cleaning part. In this part the greenhouse was cleaned, and all the present junk was removed. The biggest change was demolishing the drip racks for more space.

The next improvement was getting the second tank to the greenhouse. This tank will be used for the plants to grow inside. This will come in handy in the winter when it's really cold. When isolated well the tank can keep the plants at the right temperature even though it's too cold outside. Getting this tank inside was not the easiest, but the team succeeded with not a lot of damage to the greenhouse.

To follow up the next improvement was about isolation. This was the biggest part within the greenhouse. The isolation was necessary so the greenhouse could run in the winter months. The isolation concerned the isolation on the inside and trying to isolate the outside as well. In general, there can be said that this was also succeeded. The isolation on the inside is in most places really stable and hopefully helps keep the warmth inside.

The last part is about the door construction. The old door was not fitting in the frame anymore and was also falling apart a little. The team wanted to fix this and came up with a new door. This door was made lighter so it would not change place as much as the other door. Also, it got its own locking system on the door itself, which was an improvement when looking back at the old door. With this door the locking system was a piece of wood keeping the door in the frame. This was all but optimal.

13 Recommendations

In this part the recommendations will be discussed for the three different parts of the project. The three parts are: the remote monitoring system, the fish classifier and the greenhouse.

13.1 The remote controlling system

For the remote controlling system there are the following recommendations. The first one is to add the components for the new tank so they can also be monitored. This is important to make sure that the whole greenhouse is monitored. The integration of this things is the following recommendation. This means that the new components are showed well in the dashboard.

13.2 The fish classifier

For the recommendations of the fish classifier there are also a lot of points worth to discuss. The first one is to find a solution to force fish through the pipes. The solution of the project group was to flush them through or force them with a moving wall. Anyway, these concepts are just brainstorming ideas and need to be overthought by the next group. Also finding other solutions would be great. There also need to be thought of a solution where fish can be classified without going through a pipe. This is because it can be a problem when fish need to go through to classify as mentioned earlier.

The next recommendation will be combining machine vision and the classifying with the gates. This part was not included in this project, because there was no time left. This part is important to make sure the fish can be classified.

Finally, the setup needs to be overthought. The setup is not completely stable at the moment and the capturing area is a little bit small. The borders from the capturing area are in sight, which is not the best.

13.3 The greenhouse

Finally, the recommendations for the greenhouse. The first recommendation will be to improve the isolation any further. For now, the heating experiment showed that there is still a lot to do on the isolation.

The first step in this can be an experiment with a match searching for airgaps. Whenever an airgap is found the next group can close it to m sure the greenhouse will be airtight. When the greenhouse is airtight the isolation should work on its best.

After this a new heating experiment is needed to see if the improved isolation steps worked. This experiment is not the easiest to do as can be read in the report.

A further recommendation is to make sure the plants tanks are ready to work with. The start of this setup is already made during this semester but needs to be worked on anymore. Making the plants' tank ready is a big step for the future, because the greenhouse can run whole year then. In the plants tank the bioball tank needs to be integrated as well.

Finally, when the classifier is working this one need to be integrated. This might be the last step for the greenhouse setup. This will also be a hard part since there are a lot of things that can go wrong. This needs to be taken into account when doing this.

14 Table of figures

Figure 1. Hygrow Aquaponic Team - Spring 2022.....	2
Figure 2. Fish classifier drafts.....	13
Figure 3. Aquarium.....	16
Figure 4. Improvement of the aquarium	16
Figure 5. Leaking sensor.....	17
Figure 6. First 3D model.....	18
Figure 7. Junction between 2 pipes	19
Figure 8. 3D printed part to create the junction.....	20
Figure 9. Improvement of the junction with the 3D printed part	20
Figure 10. 3D model of the cubic elbow included the gate	21
Figure 11. Servomotor to rotate the gate.....	21
Figure 12. Two different positions for the gate	22
Figure 13. 3D model of the final solution	23
Figure 14. part to hold the LEDs for the machine vision.....	24
Figure 15. Assembly of 3D parts for the machine vision	25
Figure 16. White piece for the background	25
Figure 17. Support for the LEDs	25
Figure 18. Main part of the machine vision assembly	26
Figure 19: 3D part linking Raspberry and the pipe	26
Figure 20. Final 3D model of the fish classifier	27
Figure 21. Final version of the fish classifier	28
Figure 22. Foot to maintain the structure (pipes and 3D link).....	29
Figure 23. Plexiglass holder.....	29
Figure 24. Parts in the 3D printer.....	30
Figure 25. Settings of the 3D printer.....	30
Figure 26. The 4 supports in the aquarium.....	31
Figure 27. Vision of the support under the pipe and the holding of the plexiglass.....	31
Figure 28. Size distortion in traditional and telecentric lenses (Gregory Hollows, s.f.).....	33
Figure 29. Raspberry Pi HQ Camera with telephoto lens mounted on it	33

Figure 30. Representation of backlit machine vision (Effilux, s.f.).....	34
Figure 31. Fish imaged captured using direct lighting	34
Figure 32. Fish image captured using background light	35
Figure 33. Camera movement detector code, part 1	36
Figure 34. Camera movement detector code, part 2	38
Figure 35. Images captured by the movement detector	39
Figure 36. Code for loading images in python	39
Figure 37. Sample test and background images	40
Figure 38. Code for calculating image difference	41
Figure 39. Fish image difference heatmap	41
Figure 40. Code for binarizing and filtering by size.....	42
Figure 41. Binarized (left) and filtered by size (right) fish images	42
Figure 42. Code for estimating fish size	44
Figure 43. Output of the fish size estimation.....	45
Figure 44. Size estimator validation with object of known size.....	45
Figure 45. Automation diagram.....	48
Figure 46. Home Assistant dashboards. EC and pH sensor missing.....	49
Figure 47. HA File editor, configuration.yaml	50
Figure 48. Frontend theme configuration	51
Figure 49. HA configuration for thermostat element	51
Figure 50. Automation heating	52
Figure 51. Water tank with growing LEDs. Taken from final report Spring 2021	53
Figure 52. HA automation for the tank lights	53
Figure 53. Triggers configuration for the light automation	54
Figure 54. Conditions and actions for the light automation.....	54
Figure 55. ESPHome device configuration	55
Figure 56. Ultrasonic distance sensor	56
Figure 57. Pipe cap to hold ultrasonic sensor in place.....	56
Figure 58. ESPHome configuration for water level sensor	57
Figure 59. Water level sensor mounted	57
Figure 60. Let's encrypt plugin configuration	58

Figure 61. Diagram showing CG-NAT customers sharing public IP address (Huawei, s.f.)	59
Figure 62. Script to update DuckDNS record	60
Figure 63. DuckDNS automation configuration	61
Figure 64. Ping to our domain name	61
Figure 65. Nginx web server configuration	63
Figure 66. Proxy server connection diagram	66
Figure 67. HA HTTP proxy configuration	66
Figure 68. Hygrow GitHub profile	67
Figure 69. Monitor tests definition	69
Figure 70. Monitor frontend	69
Figure 71. Telegram bot backend configuration file	71
Figure 72. PM2 running monitor and Telegram bot on AWS	71
Figure 73. The previous layout of the Greenhouse	72
Figure 74. The future layout of the Greenhouse	73
Figure 75. Vision in front of the tanks	73
Figure 76. Example of the blue sheets on the inside for insulation	74
Figure 77. Example of the reinforcements to the inside isolation	75
Figure 78. Heating experiment temperature chart.	76
Figure 79. Heating experiment power usage	76
Figure 80. Construction of the door. Frame on the left and almost end product on the right.	77
Figure 81. Comparison of the old (left) and new (right) locking system.	78
Figure 82. End result of the door.	79
Figure 83. Evolution of our tests with plant growth	80
Figure 84. Focus on raspberries growth	80
Figure 85. Fresh-water mussels	82
Figure 86. Macrobrachium rosenbergii	84
Figure 87. Freshwater shrimp	87
Figure 88. Freshwater crabs	88
Figure 89. Langouste	90
Figure 90. Guppy grass	92
Figure 91. Green foxtail	93

Figure 92. Java Moss	93
Figure 93. Hornwort.....	94
Figure 94. Anacharis.....	94
Figure 95. Salvinia natans	95
Figure 96. Red root floater.....	95
Figure 97. Duckweed.....	96
Figure 98. PTZ camera.....	98
Figure 99. SkylinkNet HD 720p IP camera.....	99
Figure 100. AEE S80 camera.....	99
Figure 101. GoPro HERO9	100
Figure 102. Waterproof sports camera.....	100
Table 1. Lighting cost estimations.....	81
Table 2. Expenditure on the project during the semester	102

15 Bibliography

- advance, R. (n.d.). *S80 AEE Sports Camera*. (Robot advance) Retrieved 05 14, 2022, from <https://www.robot-advance.com/EN/art-s80-aee-sports-camera-3435.htm>
- Center of biological diversity. (n.d.). *FRESHWATER MUSSELS*. (Center of biological diversity) Retrieved 5 11, 2022, from [biologicaldiversity.org: https://www.biologicaldiversity.org/campaigns/freshwater_mussels/](https://www.biologicaldiversity.org/campaigns/freshwater_mussels/)
- Controls, H. (n.d.). *SkylinkNet Indoor Wireless Pan*. (Home Controls) Retrieved 05 14, 2022, from <https://www.homecontrols.com/SkylinkNet-Indoor-Internet-Security-Camera-SKWC400PH>
- Editorial, C. (2011, 01 01). *Aquarium Plants As Fish Food*. (BeChewy) Retrieved 05 14, 2022, from <https://be.chewy.com/plants-as-fish-food/>
- Effilux. (n.d.). Retrieved 2 25, 2022, from <https://www.effilux.com/en/products/backlight>
- ESPHome. (2022, 4 30). Retrieved from <https://esphome.io/>
- fast, I. (n.d.). *PTZ Camera*. (Comms express) Retrieved 05 14, 2022, from <https://www.comms-express.com/infozone/article/ptz-camera/>
- Food and Agriculture Organization of the United Nations. (2022). *Food and Agriculture Organization of the United Nations*. Retrieved from <https://teca.apps.fao.org/teca/en>
- Gembird. (n.d.). *ACAM-W-01*. (Gembird) Retrieved 05 14, 2022, from https://www.mythomson.com/uk_en/force_lng__uk_en/
- Google. (n.d.). *IPv6 adoption map*. Retrieved 2 25, 2022, from <https://www.google.com/intl/en/ipv6/statistics.html#tab=per-country-ipv6-adoption>
- GoPro. (n.d.). *HERO9 Black*. (GoPro) Retrieved 05 14, 2022, from <https://gopro.com/en/us/shop/cameras/hero9-black/CHDHX-901-master.html>
- Gregory Hollows, N. J. (n.d.). *The advantages of telecentricity*. Retrieved 2 25, 2022, from <https://www.edmundoptics.eu/knowledge-center/application-notes/imaging/advantages-of-telecentricity/>
- Healthy Betta. (n.d.). *10 Best Live Plants For Guppies: A Curated List*. (Healthy Betta) Retrieved 5 11, 2022, from <https://healthybetta.com/best-live-plants-for-guppies/>
- Houlihan, C., & Wood, J. B. (n.d.). *Marine Invertebrates of Bermuda*. Retrieved 5 11, 2022, from <http://www.thecephalopodpage.org/MarineInvertebrateZoology/Panulirusargus.html>
- Huawei. (n.d.). Retrieved 2 25, 2022, from <https://forum.huawei.com/enterprise/en/carrier-grade-nat-cgn/thread/746755-867>
- J.D.Shields. (2011, 1 5). *Diseases of spiny lobsters: A review*. (Sciencedirect) Retrieved 5 11, 2022, from <https://www.sciencedirect.com/science/article/pii/S0022201110002193>
- Madore, I. (2021, February 2). *A Guide for Keeping Freshwater Shrimp*. (Buceplant) Retrieved 05 14, 2022, from <https://buceplant.com/blogs/news/a-guide-for-keeping-freshwater-shrimp>
- Final report | Hygrow Aquaponic System

- Nandlal, S., & Pickering, T. (2006). *Freshwater prawn Macrobrachium rosenbergii farming in Pacific island countries*. (Secretariat of the Pacific Community; The University of the South Pacific) Retrieved 5 11, 2022, from <https://spccfpstore1.blob.core.windows.net/digitallibrary-docs/files/40/408631e103b84d0a408ccdda018b9a34.pdf?sv=2015-12-11&sr=b&sig=dn6rwWtPtpjF%2FWpwDdyu6mkG8AKJOIH9JBp40Iw%3D&se=2022-11-07T10%3A55%3A01Z&sp=r&rsc=public%2C%20max-age%3D864000%2C%20ma>
- Novia. (n.d.). *About us*. Retrieved 5 14, 2022, from <https://www.novia.fi/about-us>
- OpenCV. (n.d.). Retrieved 2 23, 2022, from <https://opencv.org/releases/>
- Pitcher, C. R. (n.d.). *Spiny Lobster*. Retrieved 5 11, 2022, from <https://spccfpstore1.blob.core.windows.net/digitallibrary-docs/files/60/609bb41f09736b2e99f043d75f073482.pdf?sv=2015-12-11&sr=b&s>
- PNJ. (n.d.). *AEE S80 Action cam*. (PNJ share your vision) Retrieved 05 14, 2022, from <https://www.pnj.fr/en/shop/aee-s80-action-cam/>
- Schäfer, F. (2018, 8 13). *Macrobrachium rosenbergii*. (Aquarium Glaser) Retrieved 11 5, 2022, from <https://www.aquariumglaser.de/en/fish-archives/macrobrachium-rosenbergii-3/>
- Services, F. D. (n.d.). *Aquatic Plants*. Retrieved 05 14, 2022, from <https://www.fdacs.gov/Agriculture-Industry/Aquaculture/Aquatic-Plants>
- Statista. (2021). *Statista*. Retrieved from <https://www.statista.com/statistics/418122/electricity-prices-for-households-in-finland/#:~:text=In%20the%20first%20half%20of,17.67%20euro%20cents%20per%20kWh.>
- Tom. (2019, 11 9). *10 Best Live Plants For Guppies*. (Aqua Goodness) Retrieved 5 11, 2022, from <https://aquagoodness.com/best-live-plants-for-guppies/>
- Wenger, A. (2013, 10). *Freshwater Prawns*. (Tropicalfish magazine) Retrieved 5 11, 2022, from <https://www.tfhmagazine.com/articles/freshwater/freshwater-prawns-full-article>
- Wikipedia. (2018, 10). *Freshwater bivalve*. (Wikipedia) Retrieved 5 11, 2022, from [Wikipedia.org: https://en.wikipedia.org/wiki/Freshwater_bivalve](https://en.wikipedia.org/wiki/Freshwater_bivalve)
- Wikipedia. (2021, 03 28). *Freshwater crab*. (Wikipedia) Retrieved 05 14, 2022, from https://en.wikipedia.org/wiki/Freshwater_crab
- Wikipedia. (2022, 5 3). *Nginx*. Retrieved from Wikipedia: <https://en.wikipedia.org/wiki/Nginx>
- Xerces. (n.d.). *About Freshwater Mussels*. (Xerces) Retrieved 5 11, 2022, from About Freshwater Mussels: <https://xerces.org/endangered-species/freshwater-mussels/about>