A Deliverable 2 Report on

Vertical Farm Control System

CS3500

Software Engineering

Computer Science

2022-2023

Submitted by:

Group 10

Jakab-Gyik Sarolta - 122118473

Smith Deirbhle - 120338191

Varga Zoltán - 122118466

Veres Noémi - 122118476

University College Cork

Ireland

Table of contents

Table of Figures

Table of Tables

This is to enforce that everyone has contributed to the making of the deliverable.

1. Abstract

Vertical farms are indoor farms that grow vegetables stacked on the vertical axis. In this manner, more crops can be cultivated on a smaller footprint than in traditional agriculture. The controlled environment offered by an indoor farm eliminates the risk of diseases and insects. Combined with hydroponics it increases 10 times the crop yield compared with traditional agriculture and reduces water consumption by 90% since it is recirculated in the system. Hydroponics means that the roots of the vegetables are placed in water enriched with nutrients instead of soil.
Our team has decided to create a control system for an indoor vertical farm that uses hydroponics. For the sake of simplicity this farm grows butterhead lettuce exclusively but can be extended to manage the environment of other vegetables as well.
Keywords: agriculture, energy efficient, environmentally friendly, green farm

2. Introduction

2.1 Background

Some of the vital reasons why people started experimenting with vertical farming system was connected to the exponential growth of the population and the inefficiency of the traditional agricultural methods. Most places around the world still use the same methods in growing crops then our ancestors did years ago. Which constitutes a major problem: these methods depend on various factor that the farmers cannot control. Insects, diseases, the nutrients in the soil, the general structure of the soil changing from year to year, the limitation of space and other issues.

To address them, vertical farming requires only a fraction of a farm's space to grow just as many vegetables. The insecticides and the biochemical materials that farmers use could be avoided. Since they are also harmful for the human body, this is a considerable side effect of traditional agriculture. The consistency of the soil or water that the plants are growing in is controlled by machines, the sensors are measuring the nutrient level from time to time, ph level, air and water temperature and the salinity of the water which can be determined by the EC level (electrical conductivity). This way the water can provide the ideal environment for all kinds of plants, carefully determined for every species, what would be the range of the EC level that results in the fastest growth.

2.2 Different Possible Approaches

These systems, however, do not use soil to plant the seeds in. They use special growing systems, as mentioned above, which come in 6 different forms. (seen in Figure 1 Hydroponics in vertical farming systems)

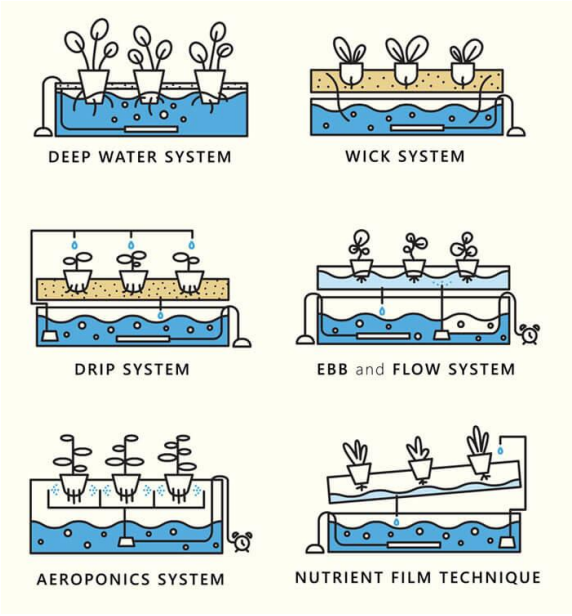

Figure 1 Hydroponics in vertical farming systems

1. ***Deep Water system:*** the roots of the plants are floating in a nutrient-dense water with oxygen. A container acts as a reservoir for this solution.
2. *Wick system:* does not consume electricity but it uses the wicks of the plants to provide the nutrient-dense water for the plants growing in the soil. The least complex and most energy efficient approach.

3. *Drip system:* the nutrient solution is dripped either to the roots of the plants or to the leaves, coming from above. It depends on the types of plants the farm wants to cultivate.
4. *Ebb and Flow system (also called flood and drain):* the roots of the plants are periodically flooded with nutrient dense water.
5. *Aeroponics system:* the roots of the plants are sprinkled with the nutrient solution.
6. *Nutrient Film technique:* the system works with pipes that hold the nutrient solution for the plants, then the water is collected from them by letting it flow back to the reservoir from these pipes. The pipes are moving; the incline makes the water flow down and then a new amount is pumped of into the pipe. The water is recycled this way; less water consumption.

In this control system the first approach is taken since it does not require electricity and it is one of the simpler ways to realize the physical structure of the project.

2.3 Pros and cons of vertical farming

Among the **advantages** of vertical farming and hydroponics, we can mention the small footprint of the growing area, the reuse of water, the elimination of pesticides and harming substances. These all contribute to the growth of healthy and nutrient-rich crops. The production can be made carbon neutral by capturing rainwater and installing solar panels. Plants can be cultivated in a stable and predictable manner, isolated from the unforeseeable outside environment.

Vertical farms and hydroponic cultivation have **drawbacks** as well. Among the biggest disadvantage we can mention the limited variety of plants that can be grown in such an environment, and the natural processes, such as pollination, the system needs to replace. Artificial lighting can increase the energy consumption by a great extent. Also, the initial phase of fine-tuning and high dependence on technology can carry possible vulnerabilities too.

3. Requirements

Requirements are essential in the process of creating a product, they set achievable goals for the team and provide a tool to measure progress. Also, they provide a perspective on the project by mentioning what needs to be done. Further on, we detailed two types of requirements: non-functional requirements, and functional requirements.

3.1 Non-Functional Requirements

Non-functional requirements contain demands that are concerning conceptual properties of a product. They do not say what to do, but what properties the system needs to have while doing the actions stated by functional requirements. The system should be:

1. *Intuitive and easy to use by the administrator:* The GUI of the application should not be a burden for the administrator to use. It should be clean and intuitive.
2. *Reliable results and reports:* The reports generated by the system on the request of the administrator should reflect the reality. Therefore, they should be taken in real-time and displayed in an accurate way.
3. *Precise:* Sensors should be calibrated before use and maintained so. Unprecise data would have terrible effects on the environment and on the yield. The whole system relies on the data provided by the sensors.
4. *Deal with incorrect user inputs and display messages that can help the admin find the issue:* The application should always validate the input of the admin. The initial parameters

provided are the base of the environment that will be maintained; therefore, no error should occur in this phase.

5. *Efficient*: The whole point of vertical farms and hydroponics is to be as efficient as possible with the scarce resources that are available. Thereby the control system should manage these resources as well as possible.
6. *Highly productive*: The high efficiency is obtained not only by the good resource management, but also by maximizing the yield and the space used.
7. *Environmentally friendly*: Combined the two requirements above, environmentally friendliness is achieved. No waste and efficient use of resources contributes to this aspect.

3.2 Functional Requirements

Functional requirements say what the system should be able to do, but without mentioning how that should be achieved. They should be compliant with the SMART requirement's properties; they should be specific, measurable, attainable, realistic, and traceable. Our functional requirements are presented below. Note that these requirements change with respect to the given parameters of the system administrator.

Table 1 Contains actions that are triggered based on intervals given as system parameters.

Environment property	When to do	What to do	Until when to do
Air temperature	< *[minimum_air_temperature]* – [*temperature_treshold]*	R1. Start heating the air	>= *[minimum_air_temperature]*
	> *[maximim_air_temperature*] – [*temperature_treshold]*	R2. Start cooling the air	<= *[maximum_air_temperature]*
Water temperature	< *[minimum_water_temperature]* – *[temperature_treshold]*	R3. Start heating the water	>= *[minimum_air_tempreature]*
	< *[maximum_water_temperature]* – *[temperature_treshold]*	R4. Start cooling the water	<= *[maxium_air_tempreature]*
Humidity	< *[minimum_humidity]* – *[humidity_treshold]*	R5. Start humidifier	>= *[minimum_humidity]*
	> *[maximum_humidity]* – *[humidity_treshold]*	R6. Start dehumidifier	<= *[maximum_humidity]*
pH level	< *[minimum_pH]* – *[pH_treshold]*	R7. Start alkaline dispenser	>= *[minimum_pH]*
	> *[maximum_pH]* – *[pH_treshold]*	R8. Start acid dispenser	<= *[maximum_pH]*
Electrical conductivity	< *[minimum_EC]* – *[EC_treshold]*	R9. Increase EC	>= *[minimum_EC]*
	> *[maximum_EC]* – *[EC_treshold]*	R10. Decrease EC	<= *[maximum_EC]*

The *[parameter_name]* is representing the corresponding system parameter. For example, the heating of air will be turned on when the temperature drops below the minimum temperature - a threshold. And is continuing heating until the temperature gets above the minimum temperature. The system has other requirements such as:

R11. The system will check the air temperature every *[air_temperature_check_interval]*.
R12. The system will check the water temperature every *[water_temperature_check_interval]*.
R13. The system will check the nitrogen(N) level in the water every *[nitrogen_check_interval]*.
R14. The system will check the phosphorus(P) level in the water every *[phosphorus_check_interval]*.
R15. The system will check the potassium(K) level in the water every *[potassium_check_interval]*.
R16. It will ensure the levels are kept to an NPK ratio of *[NPK_ratio]* by adding nutrients as needed.
R17. The system will turn the lights on for *[light_ON_time]*.
R18. The system will turn the lights off for *[light_OFF_time]*.
R19. The system will check the humidity level in the room every *[humidity_check_interval]*.
R20. The system will check the pH level in the water every *[pH_check_interval]*.
R21. The system will check the EC level in the water every *[EC_check_interval]*.
R22. If the system cannot solve an issue it will alert the administrator.

4. Design Diagrams

4.1 Use-Case Diagram

The use-case diagram (seen in Figure 2) is aimed to graphically capture the system's actors, i.e. the administrator and the actions which can be performed: initialize the system parameters, adjust system parameters, generate reports, receive alerts and reset the system. This diagram provides a structure for our application as it helps to identify components in the design phase. It also helps to capture the requirements which are presented in detail.

The system administrator should be a biologist/botanist who can supervise the growing phase, detect problems in time and adjust the parameters accordingly. He plants the seeds and initializes the system. Also, he harvests the plants and resets the system.

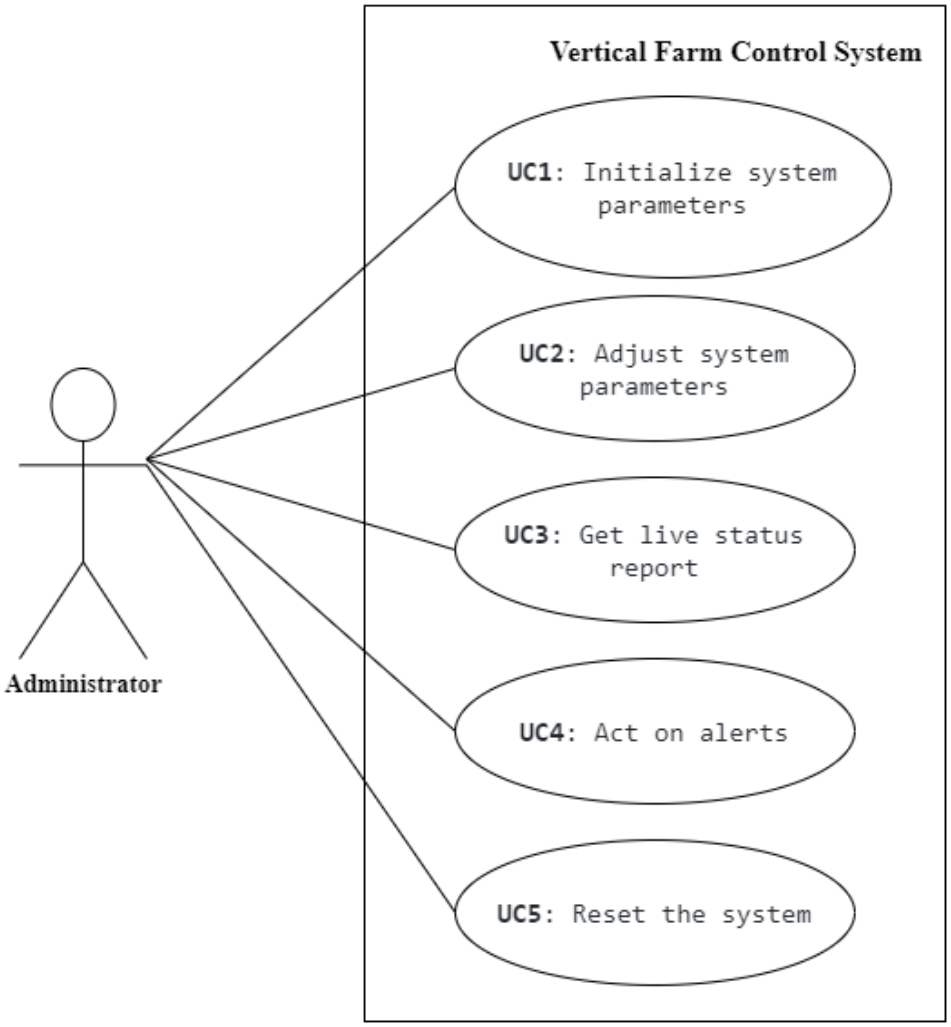

Figure 2 Use-case diagram of the Vertical Farm Control System

Now the use cases will be presented:

Use Case 1: initialize system parameters *(for sequence diagram see Figure 3)*
Primary Actor: administrator
Main Success Scenario:
1. The administrator inserts the values for the: time, air temperature, water temperature, nutrient levels, light intensity, light spectrum, humidity, pH level and conductivity level
2. The administrator clicks on the "validate input parameters" button
3. The application validates the data and displays a message informing the administrator to start the system
4. The administrator clicks on the "start" button
5. The application starts the system
Alternative Sequence: Invalid values for the setup parameters
The administrator inserts invalid values for the application's setup parameters
The application displays an error message and requests the administrator to insert valid values
The scenario returns to step 1

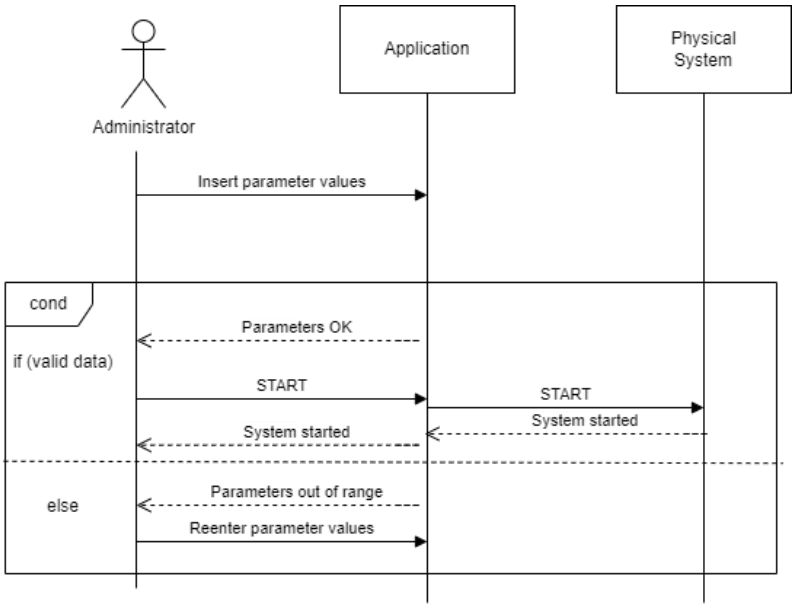

Figure 3 Sequence diagram for use case 1

Use Case 2: adjust system parameters *(for sequence diagram see Figure 4)*
Primary Actor: administrator
Main Success Scenario:
1. The administrator selects the parameter to edit
2. The administrator clicks on the "Edit" button
3. The administrator enters the new parameter
4. The new parameter is saved
5. A confirmation message is displayed
Alternative Sequence: Invalid values for the parameters
The administrator inserts invalid values for the application's parameters
The application displays an error message and requests the admin to insert valid values
The scenario returns to step 3

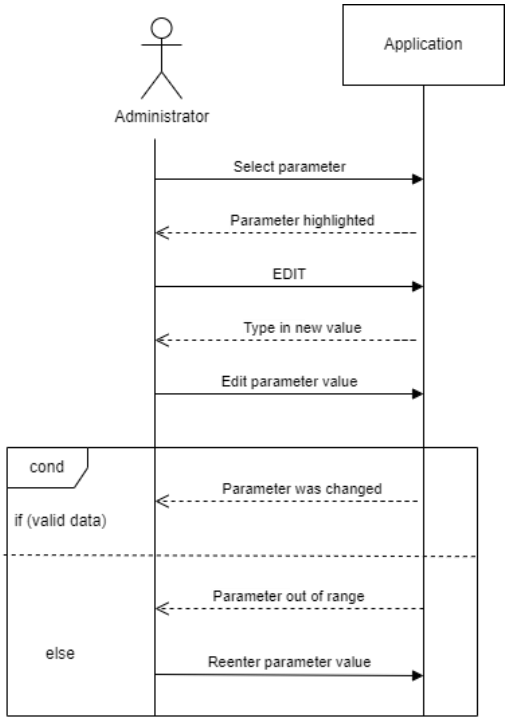

Figure 4 Sequence diagram for use case 2

Use Case 3: get live status report *(for sequence diagram see* Figure 5*)*
Primary Actor: administrator
Main Success Scenario:
1. The administrator should select the data based on which they want to create the reports
2. The administrator presses the “Generate report” button
3. A report is generated

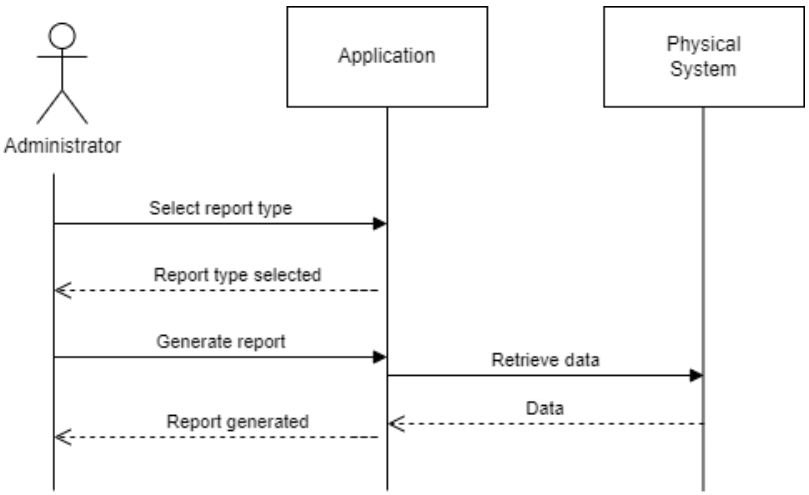

Figure 5 Sequence diagram for use case 3

Use Case 4: act on alerts *(for sequence diagram see* Figure 6*)*
Primary Actor: administrator
Main Success Scenario:
1. The application senses unusual behaviour which requires further action from administrator
2. The application displays an alert message informing the administrator about some specified faults
3. The administrator reads the alert and takes further action.

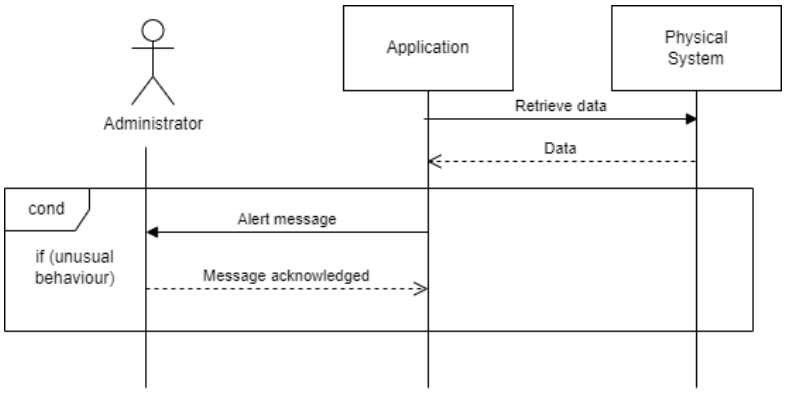

Figure 6 Sequence diagram for use case 4

Use Case 5: reset the system *(for sequence diagram see Figure 7)*
Primary Actor: administrator
Main Success Scenario:
1. The administrator should press the "Reset" button
2. The application returns to an initial phase

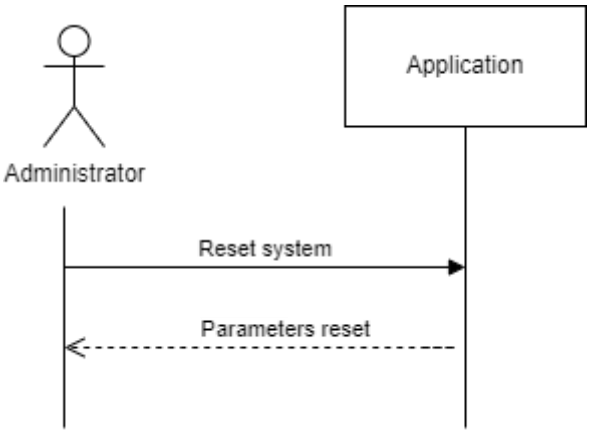

Figure 7 Sequence diagram for use case 5

4.2 Control System Diagram

The block diagram of the system (seen in Figure 3) shows the interaction between the components of the system (sensors, actuators, the control algorithm, the plant). Each block represents one of these components. Since the input depends also on the measured quantities, we are using a feedback loop which returns the measured values from the sensors.

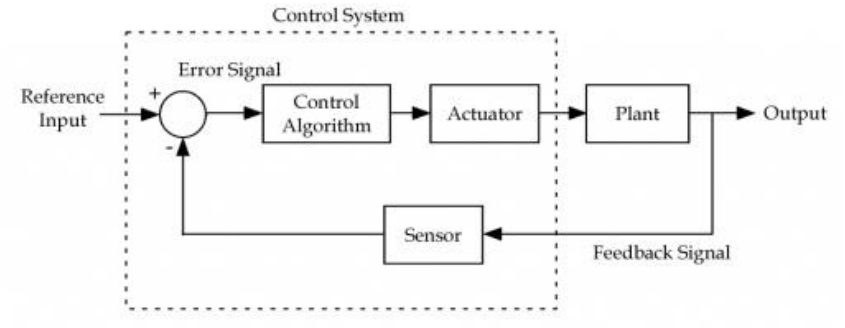

Figure 8 Feedback Control System diagram

4.3 Finite State Machine

The finite state machine diagram below (seen in Figure 9) captures the transitions between the states of the system. It starts off with the "system initialized" state, after the admin inputs all the necessary system parameters, and then it continues until the crops are harvested. The system has 6 states in total: "system initialized", "data collection", "environment balancing", "balanced environment", "alerted" and "mature crops".

The "alerted" state is one of the more complex states that needs two measurements and has 2 different cases in which the system needs to enter this state. Firstly, in the positive-value case, the system determines by subtraction if the current measurement is less than the target value. If it is, then it takes a second measurement value after if has taken some action to correct the problem. This is the m2 value. Then it compares the first measured value and the second one. If the target value of the measurement has not improved (in less than or equal to the first measurement), then something in the system must be broken. Thus, an alert must signal this to the admin. In the negative-value case, if we need to decrease a measured value but the second measurement seems to be greater than or equal to the first one, then the system sends an alert.

Symbols

T = predefined time interval when the sensors off, the interval at which we make measurements in balanced state
t = the current time since the system has entered the current state
x_{target} = the target value that the system is aiming to achieve
x_{m1} = the value of the first measurement
x_{m2} = the value of the second measurement
predefined_time = the interval at which we make measurements in environment balancing state

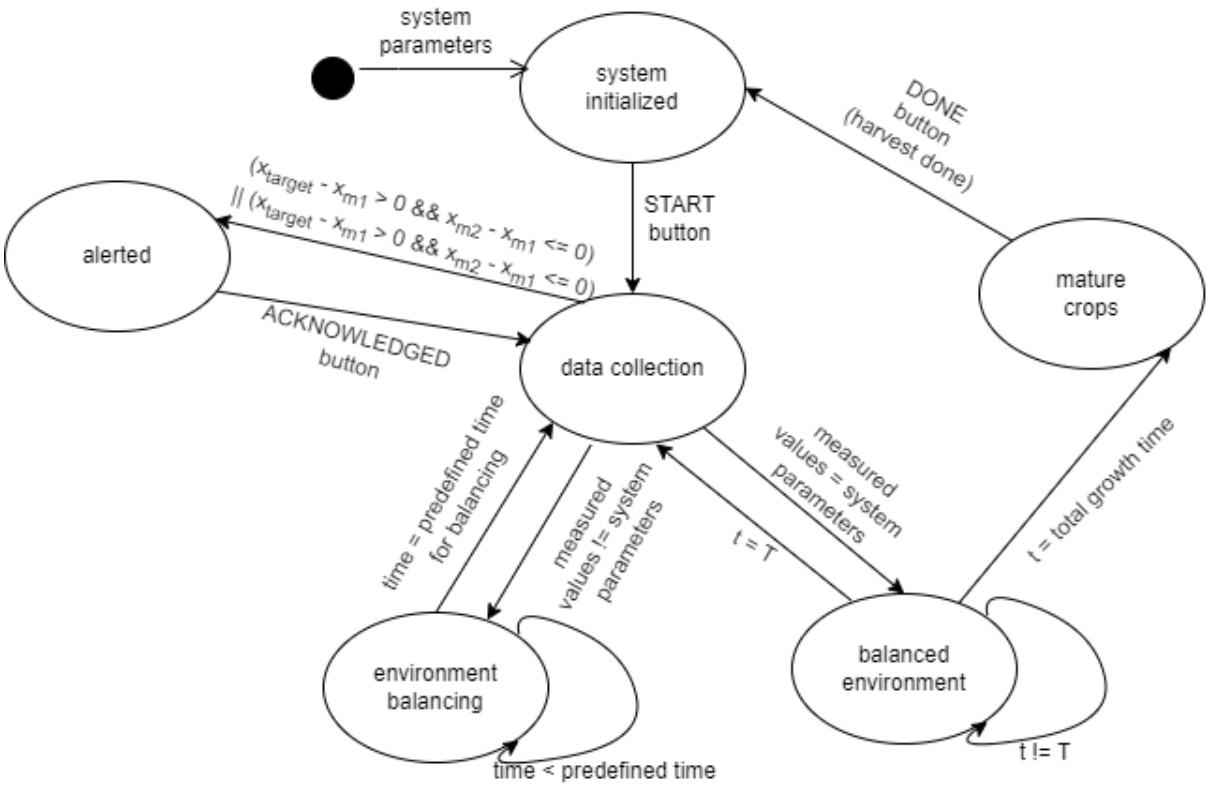

Figure 9 Finite state machine diagram

4.4 Data flow

The flow of data in our system is illustrated by the Data Flow Diagrams below. They capture the movement of data between external entities, processes, and data stores.

On **Level 0** there is a **Context Diagram** *(see* Figure 10*)* which is a high-level data flow diagram. It shows that we have an *administrator* and the *physical system* as entities that interact with the *application* which represents the whole control system as a process. The data sent by the *administrator* to the *application* are the initial parameters. It gets back from the *application* the status report. On the other hand, we have the *physical system* as an entity as well. It gets the start command from the *application*, and it sends back the measured values

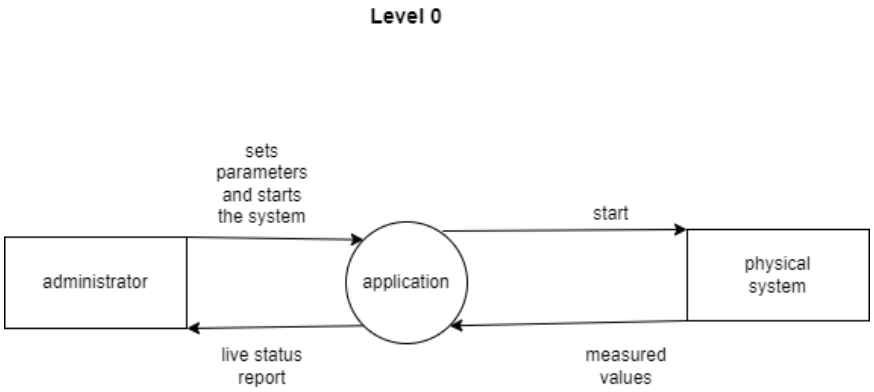

Figure 10 Level 0 (Context) data flow diagram

The **level 1** *(see* Figure 11*)* diagram puts more detail in the context diagram. It highlights the main functions of the system. In addition to the diagram a level below it adds the *database* which is used by the *start system* process to save measurement data into it. It is also queried by the *generate live report* process for which it provides the measured values in the past. The *generate live report* process than sends the report to the *administrator*.

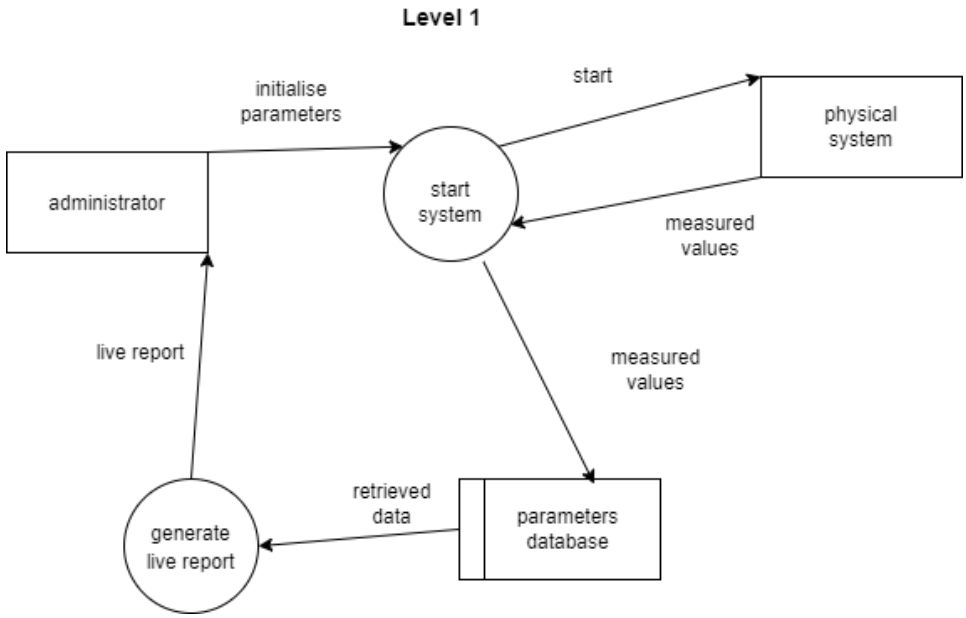

Figure 11 Level 1 data flow diagram

The **Level 2** *(see Figure 12)* Data Flow Diagram adds even more detail the previous ones. The *start system* process is divided further into *validate input* process that receives the input from the user upon initialization, sends the data to the database and to the *environmental balancing* process. This process constantly compares the values from the sensors, using a new process called *retrieve sensor values*, to the ones set by the admin. In case the expected values are different from the actual values the process sends a command data to the physical system to start the required actuators. If something unusual is detected this process sends the unusual values to the *show alerts* process to generate an alert for the administrator. Another process that resulted from the initial *start system* process is the *periodically compare ideal and actual state* process. This will get its data from the database (ideal state) and from the sensors through *retrieve sensor values* process (actual state) and if those do not match it sends the values that are wrong to the *environment balancing* process. The *show parameters and report* process gets the data from the database and sends a report or just the plain data to the administrator to take further action.

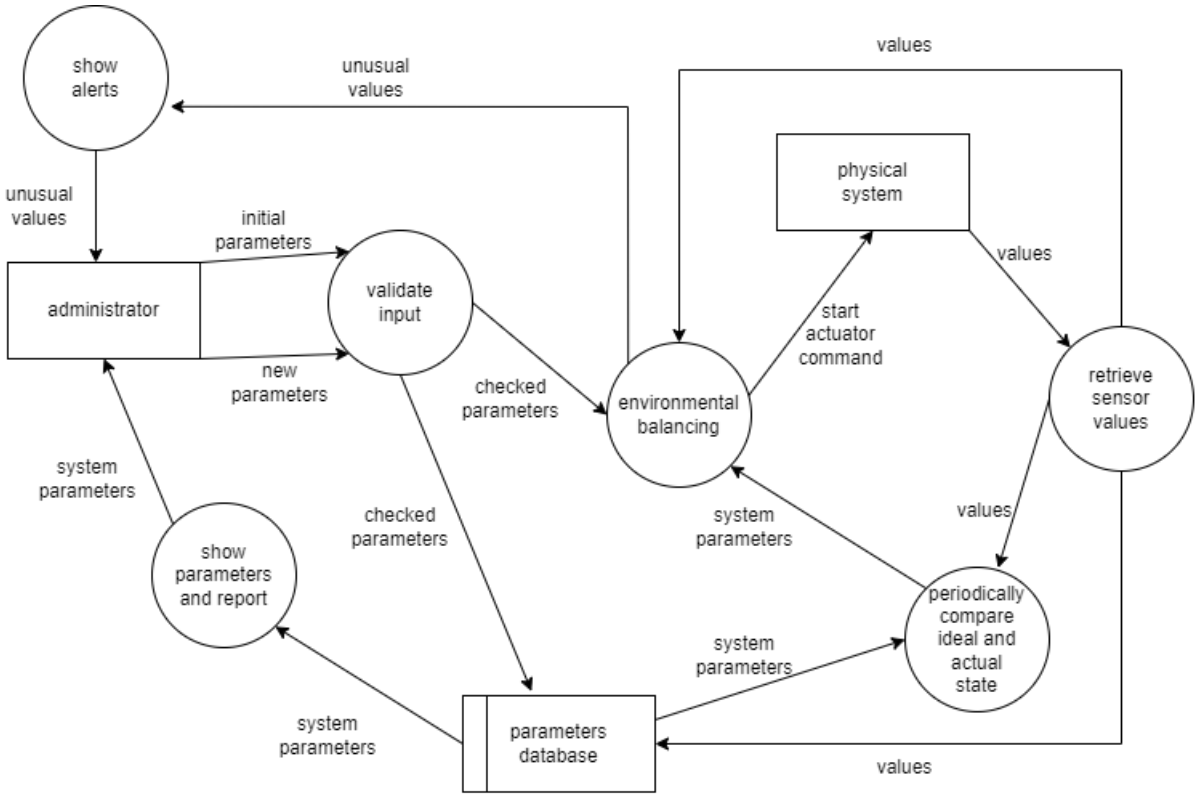

Figure 12 Level 2 data flow diagram

5. References

[1] "IoT based Hydroponic Farms" [Google Scholar]
Retrieved from: https://ieeexplore.ieee.org/document/8748447
[2] "Nutrient Use in Vertical Farming: Optimal Electrical Conductivity of Nutrient Solution for Growth of Lettuce and Basil in Hydroponic Cultivation" [Google Scholar]
Retrieved from: https://www.mdpi.com/2311-7524/7/9/283/htm
[3] Data Flow Diagrams:
Retrieved from: https://www.youtube.com/watch?v=euI2AFobS0w
https://www.geeksforgeeks.org/levels-in-data-flow-diagrams-dfd/