Critical Systems Lab - MESCC
Water Pumping Automated System

Ricardo Mendes
1201779

Arthur Gerbelli
1220201

ISEP, January 2024

Contents

1 Introduction

The current document, is the result of the work done during the first delivery of the CSLAB class.

The document is divided into three parts, each one of them focused on the evaluation topics: **requirements specification** documentation, **rationale for selected technology** and **list of physical sensor and/or actuator** used for the demo.

The system that we are modeling is a Water Pumping System (WPS) for two rainwater wells. These types of systems are essentially used to move water from a lower elevation to a higher one.

A Remote Status Station is also described in the document. Its main function is to give a level of observability of the WPS and to alert the *maintenance team* for a possible failure.

There is also an additional feature. The status of the system should also be visible through a web server.

2 Requirement Specification

2.1 Problem Domain

2.1.1 Stakeholder Needs

Based on the system's description and some clarifications during the classes, we identified the following Stakeholder needs:

SN-1 .1: The water in the WPS must be pumped from a lower level to a higher one.
.2: Every WPS is an independent system; they don't have influence on each other.

SN-2 .1: The status of each element of the wet well needs to be displayed in a Remote Status Station (RSS).
.2: The RSS must display the water level, the pump status, an alarme and a button to disable the alarm.
.3: The alarm must be ON when a problem in the system is identified.

SN-3 The status information must be accessible through a web page.

SN-1 is WPS specific, **SN-2** is RSS specific and **SN-3** is Web Server specific.

2.1.2 System Context

For a better understanding of the system, we developed an external view, and so, identified external entities that do not belong to the system but interact with it. The following diagram is the output of this analysis:

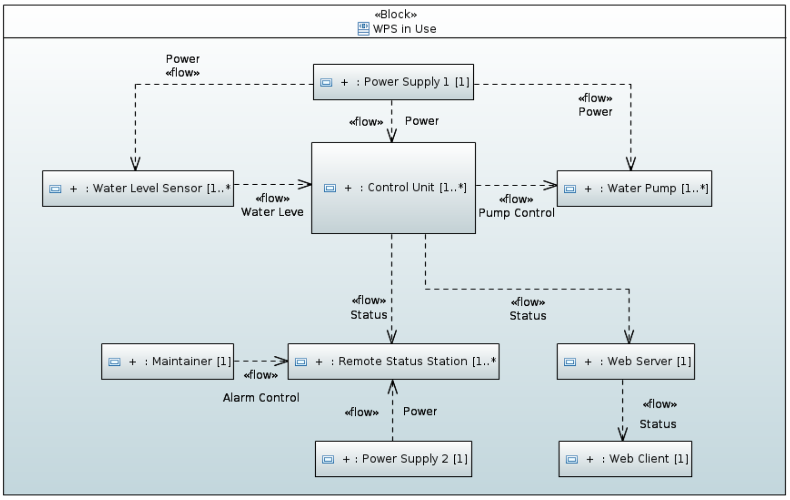


Figure 1: System Context

Although some elements represented in the diagram are part of the WPS (sensor, control unit, pump and RSS), we divide them into subsystems with their own responsibilities and interactions.

By decoupling the WPS responsibilities, we are simplifying it and turning the critical system requirements easier to grasp and model.

The main identified external entities are: **Power Supply**, **Maintenance Team** and **Web Client**.

Please notice the independent Power Supply for the Control Unit and the RSS as a way to deal with the critical requirements.

2.1.3 Use Cases

By analyzing the Stakeholder Needs, we can see that the main goal of the WPS is to control the water level inside the wet well. This goal can be captured in the model as the *"Control Water Level"* use case of the *Control Unit In Use* system context.

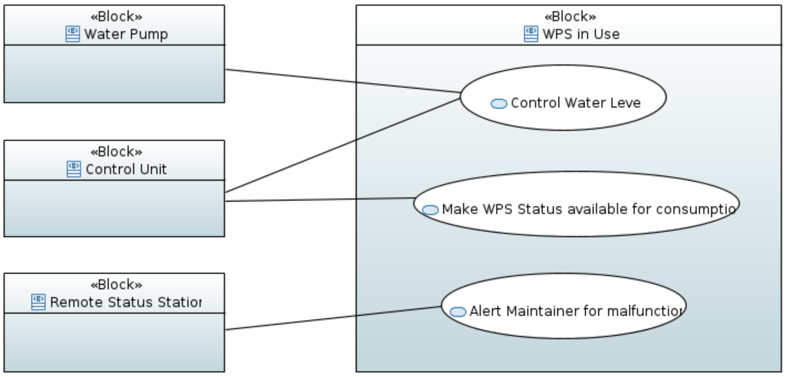


Figure 2: Use Case diagram

A closer look at the use case *"Control Water Level"* gave rise to the below activity diagram. No alternative scenario was modeled.

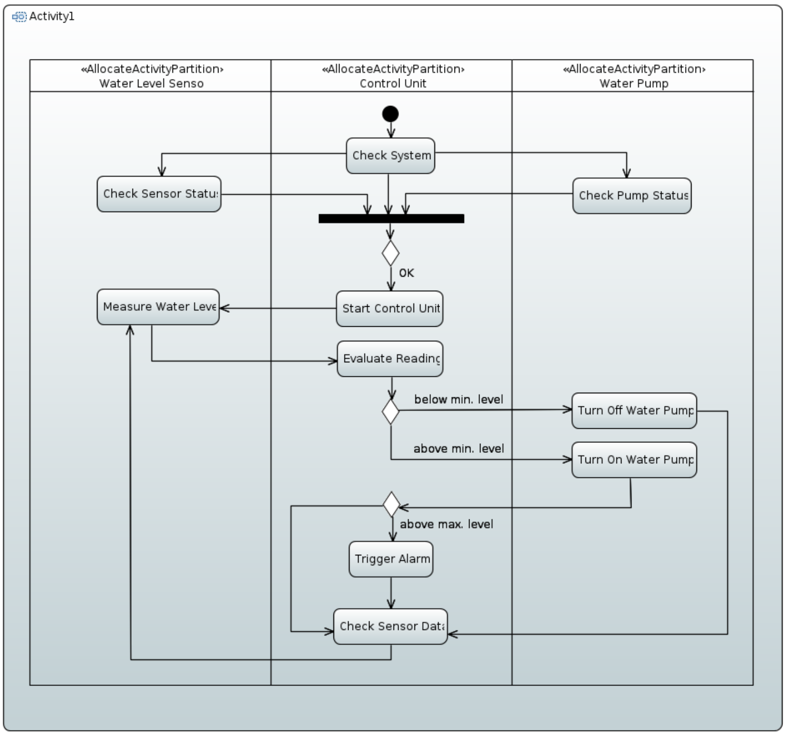


Figure 3: Use Case Activity diagram

2.1.4 Measure of Effectiveness

To be able to describe the performance of the system, some quantifiable characteristics of the WPS were identified:

- Energy Consumption in *kilowatts per hour*
- Wet Well Capacity in *cubic meters*;
- Water Inflow in *liters per second*;
- Water Outflow in *liters per second.*

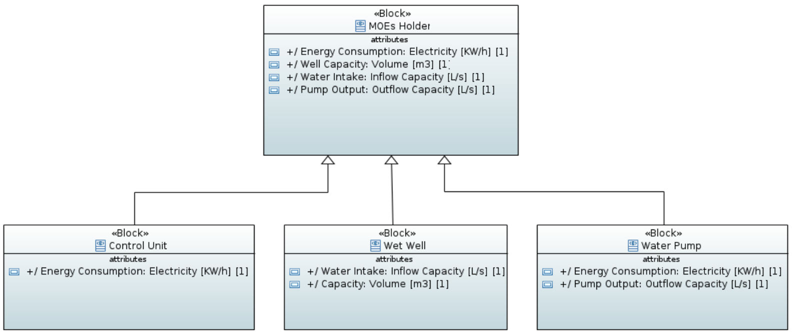

«Block»
MOEs Holder
attributes
+/ Energy Consumption: Electricity [KW/h] [1]
+/ Well Capacity: Volume [m3] [1]
+/ Water Intake: Inflow Capacity [L/s] [1]
+/ Pump Output: Outflow Capacity [L/s] [1]
«Block»
Control Unit
attributes
+/ Energy Consumption: Electricity [KW/h] [1]
«Block»
Wet Well
attributes
+/ Water Intake: Inflow Capacity [L/s] [1]
+/ Capacity: Volume [m3] [1]
«Block»
Water Pump
attributes
+/ Energy Consumption: Electricity [KW/h] [1]
+/ Pump Output: Outflow Capacity [L/s] [1]

Figure 4: Measure of Effectiveness diagram

As illustrated, each characteristic can be related to a specific element of the WPS.

2.1.5 Functional Analysis

For the functional analysis, we only focused our attention to the Control Unit; the most critical of the subsystems.

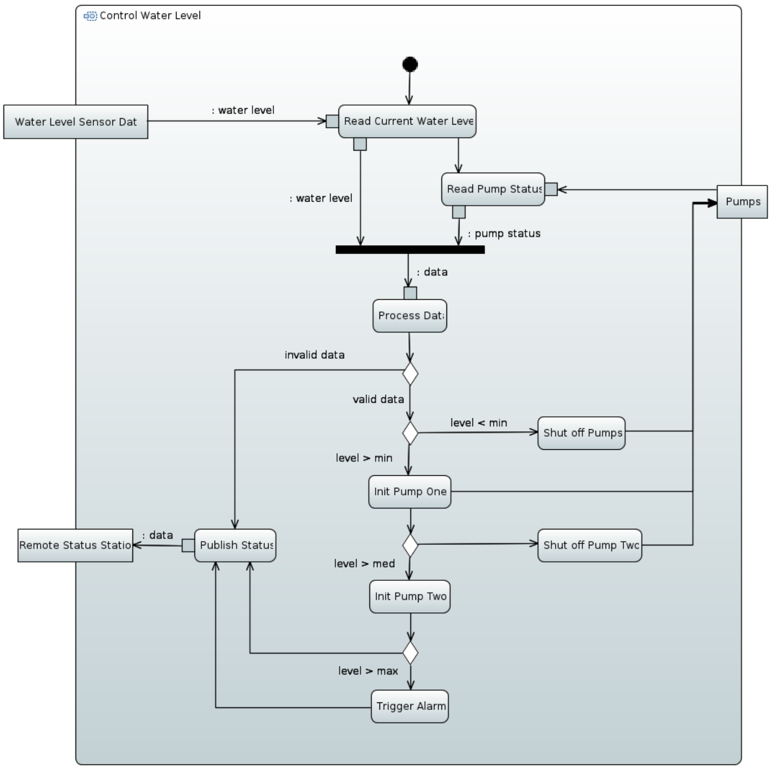


Figure 5: Functional Analysis diagram

For a reliable *Water Level Control*, the Control Unit needs to be able to read data from the sensors and also get the current status of the water pumps.

This data is used to decide the correct behavior - turn on or off the water pumps - and publish the system's status to be consumed by the RSS.

The system's status also has the information that can trigger an alarm on the RSS side.

2.1.6 Conceptual Subsystems

After identifying conceptual subsystems during the functional analysis, we tried to capture the communication between them. This was achieved by pinpointing the inputs and outputs of those subsystems.

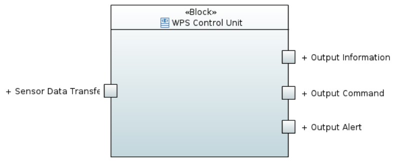


Figure 6: Communication WPS

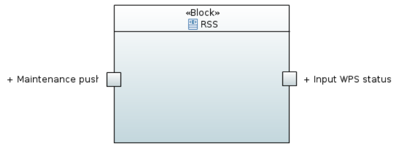


Figure 7: Communication RSS

In the diagrams above, we analyzed the WSP Control Unit and the RSS.

2.2 Solution Domain

2.2.1 System Requirements

These system's requirements were expressed using the EARS method. The objective of this approach is to state in a concise, feasible and traceable way what the system must do and its limitations.

All the requirements below are functional requirements. non-functional requirements are not being taken into account. These are specified on a later chapter.

SR-1 **.1:** While the water level is above the minimum level, WPS shall have a pump working.

.2: When the water level is below minimum level, WPS shall have all pumps stopped.

.3: If the water level is above the maximum level, then the WPS shall trigger an alarm at the Remote Status Station (RSS).

.4: A second pump shall be turned on only when the water level is above 2/3 the maximum water level.

.5: When only one pump is available, the maximum water level shall be reduced to 2/3.

SR-2 **.1:** The status of all WPS shall be displayed on all RSS.

.2: If the alarm is ON, the button in the RSS shall only disable it.

SR-3 The status of all WPS shall be visible on a web page.

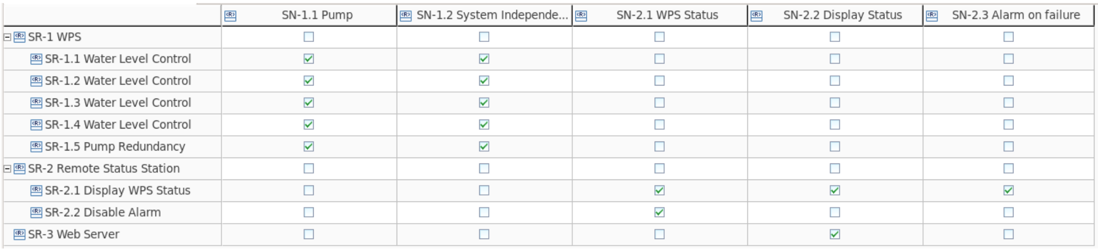
SR-1 is WPS specific, **SR-2** is RSS specific and **SR-3** is Web Server specific.

The requirements are the output of the previous analysis. If we look at the functional analysis made for the *"Control Water Level"* use case, we can clearly map it to the **SR-1**.

2.2.2 Traceability to Stakeholder Needs

In order to confirm that the output of the problem domain model is being followed, we need to establish traceability relationships.

In this particular case, we jumped the system's functions during the white-box analysis, and focused on the identified requirements and confirmed that they respond to the Stakeholders needs.



	SN-1.1 Pump	SN-1.2 System Independe...	SN-2.1 WPS Status	SN-2.2 Display Status	SN-2.3 Alarm on failure
⊟ SR-1 WPS	□	□	□	□	□
SR-1.1 Water Level Control	☑	☑	□	□	□
SR-1.2 Water Level Control	☑	☑	□	□	□
SR-1.3 Water Level Control	☑	☑	□	□	□
SR-1.4 Water Level Control	☑	☑	□	□	□
SR-1.5 Pump Redundancy	☑	☑	□	□	□
⊟ SR-2 Remote Status Station	□	□	□	□	□
SR-2.1 Display WPS Status	□	□	☑	☑	☑
SR-2.2 Disable Alarm	□	□	☑	□	□
SR-3 Web Server	□	□	□	☑	□

Figure 8: Traceability to Stakeholder Needs

As illustrated, the requirements are answering the Stakeholder needs.

2.2.3 System Structure

Having the system requirements, we can start thinking about the logical architecture of the WPS.

In this case, the subsystem of the solution domain, don't differ from the subsystems identified in the problem domain.

Although the Web Server was not explicitly described during the problem domain, this fact only happened because of a more focused analysis of the critical parts.

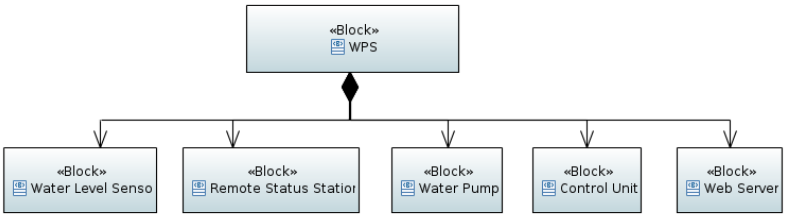


Figure 9: System Structure Diagram

2.2.4 System Behavior

Mealy Finite State Machine to describe the behavior of the WPS.

The only input to the machine is the current water level. The state of the machine changes according to these values.

There are two main states are ON and OFF, corresponding to water being or not pumped outside the well. If the water level in above the minimum, a water pump must be working (ON LOW). Depending on the levels above minimum, we can have a second pump working with (ON HIGH) or without (ON ALERT) an alert being triggered.

The state machine, beneath, models this behavior.

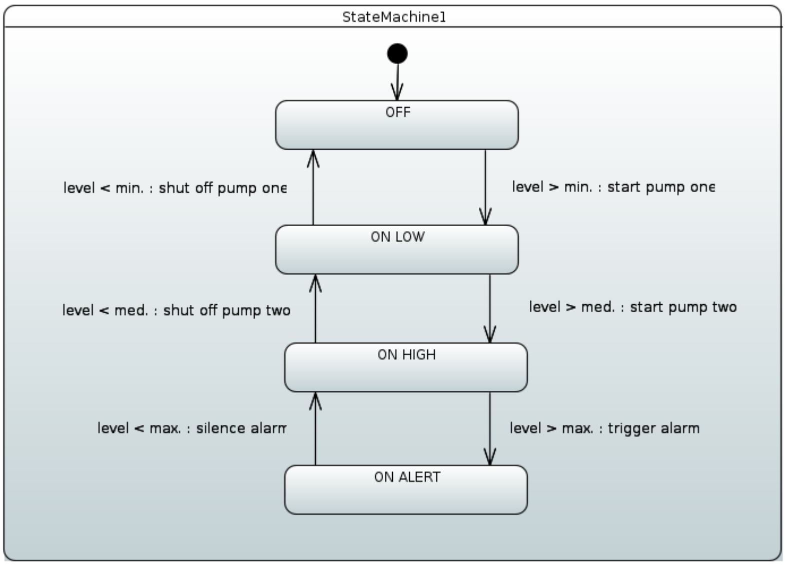

StateMachine1
OFF
level < min. : shut off pump one
level > min. : start pump one
ON LOW
level < med. : shut off pump two
level > med. : start pump two
ON HIGH
level < max. : silence alarm
level > max. : trigger alarm
ON ALERT

Figure 10: State Machine

2.3 Analysis of safety and reliability

Given that we are dealing with a critical system, the analysis of safety and reliability as bigger impact on the implementation of the solution.

H-1:

- **Description:** One of the pumps stops working.
- Cause: Mechanical problem.
- Effect: Lost of redundancy and reduction of system performance.
- **Mitigation:** Reduce the maximum water level to 2/3 and trigger alarm.

H-2:

- **Description:** The two level sensors give contradictory readings, i.e. one above max and one below min.
- Cause: Sensor malfunction, connection issues.
- Effect: Inappropriate system behavior.
- **Mitigation:** Choose a worst case or compare with the last reading to find the fault. Always trigger the alarm.

H-3:

- **Description:** Power shortage.

- Cause: Multiple causes
- Effect: Complete failure of the system.
- **Mitigation:** RSS with independente power supply and trigger alarm.

H-4:

- **Description:** Both pumps stopped working.
- Cause: Mechanical problem.
- Effect: Complete failure of the system.
- **Mitigation:** Trigger alarm.

H-5:

- **Description:** RSS are not getting information from WPS.
- Cause: Connection issues or Messagem broker stoped working.
- Effect: Unknown status of the system.
- **Mitigation:** Implement a cluter of MQTT Brokers or remove this single point of failure by adopting DDS.

H-6:

- **Description:** RSS stops working.
- Cause: Malfunction.
- Effect: Unknown WPS status.
- **Mitigation:** Have redundancy by having multiple RSS and each one displaying all statuses from all WPS.

H-7:

- **Description:** A pump doesn't turn OFF when the water level in bellow minimum.
- Cause: Mechanical problem.
- Effect: Pump overheating and complete failure.
- **Mitigation:** Trigger alarm.

Most of the hazards here listed could be handled as non-functional requirements. Because of that, a more detailed description of some of them is needed.

H-1 and **H-4** touches on the redundancy of the water pumps. Because having an unused water pump would mean a reduced performance of the system, we choose to use the two pumps even when the system is healthy. The second pump is only used when the water level is high. If, for some reason, one of the pumps is not working, the system will reduce the max capacity of the well and trigger an alarm to alert the maintenance team.

H-2 is interesting because introduces a problem that cannot be answered with a reliable voting system. If the reading of both sensor are too unequal, there must be a way to distinguish between the wrong and the correct data. There are three possible ways to deal with the issues: choose a master and a slave sensor, retain the previous input and compare it with the current one, or choose the worst case.

H-3 and **H-6** illustrates the importance of the RSS. During a localized power shortage on the well, the RSS should not be affected and so, still be able to alert the maintenance team. Having two RSS in the building also assures redundancy, specially if they have separated power supplies.

H-5 is sensible to a special limitation of having only one system dealing with the main communication. A single MQTT broker is also a single point of failure that could jeopardize the whole system. Having a cluster of brokers or using a middleware like DDS is a way to mitigate or even remove completely this risk.

3 Selected Technologies

Before describing the selected technologies, we want to emphasize that the selection was made on the basis of this being an academic exercise and that at the end a demo must be presented using microcontrollers, simple sensors and actuators.

Here are the selected communication technologies:

- **MQTT** is a lightweigh messaging protocol specially developed for the IoT. It is based on the publish/subscribe paradigm and is a perfect choise for a small code footprint and reduced network bandwidth.

  The MQTT protocol was not design to deal with authentication or authorization standards in a possible effort to avoid weaken some of its strengths i.e. low bandwidth usage, speed and low power consumption.

  Not being encrypted, MQTT supports itself on TLS/SSL for security encryption as we already wrote in a previous chapter. MQTT is not trust symmetrical and by using SSL certificates the client can have some assurance that the server is legit.

  In conclusion, the protocol includes recommendations for its implementation in application so to ensure a level of security according to today's standards

  Although the above-mentioned security issues, the main communication protocol between sensors and control units will be MQTT. Because the status of the system will be displayed in the RSS and also on a web page. The publish/subscribe method is ideal when multiple consumer are in play.

- **TCP** will be used as a fallback in case no data is being made available by the MQTT broker. If a sensor fails and the control unit confirms that no message is being sent, it can always make a request to the sensor the query about its status.

  TCP is also the main communication protocol of the IP suite. Some of the communication in the network that we will use for the demo will use it on a higher level, i.e. HTTP.

- **HTTP and REST**. The required web page, that displays the status of the system, almost forces us to use the HTTP protocol during the client-server communication. On the server side, an API following the REST guidelines will be implemented.

- **JSON**. The data between Web client and Web server will be made available in JSON. This text format is ideal for the demo because of being widely uses, language independent and easily parsed by all the main programming languages.

  Binary formats like Parquet or Protobuf could have being an interesting alternative, but the overhead of the implementation is not realistic for this exercise.

Here are the selected programming languages:

- **C**. All the main services will be written in C. This enables us a smaller code footprint an a greater control of the hardware resources.

- **Assembly** is a requirement of the exercise and will be used to simulate the Water Level Sensors.

- **Python** is certainly not used in critical systems. Because of that, this programming language will be used to implement the HTTP server that doesn't have any critical requirement.

- HTML is not a programming language! But for the sake of completeness, this markup language will be used to use a common browser as our Web client.

To have a better overview of the proposed implementation of the system, we elaborated a deployment diagram with all the discussed services.

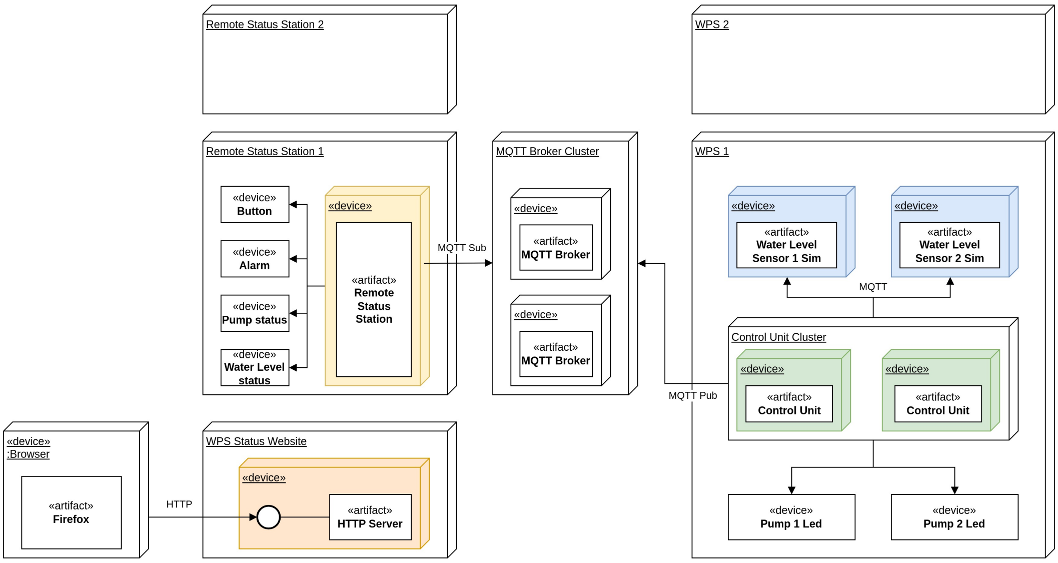


Figure 11: Deployment diagram

4 List of physical sensors/actuators

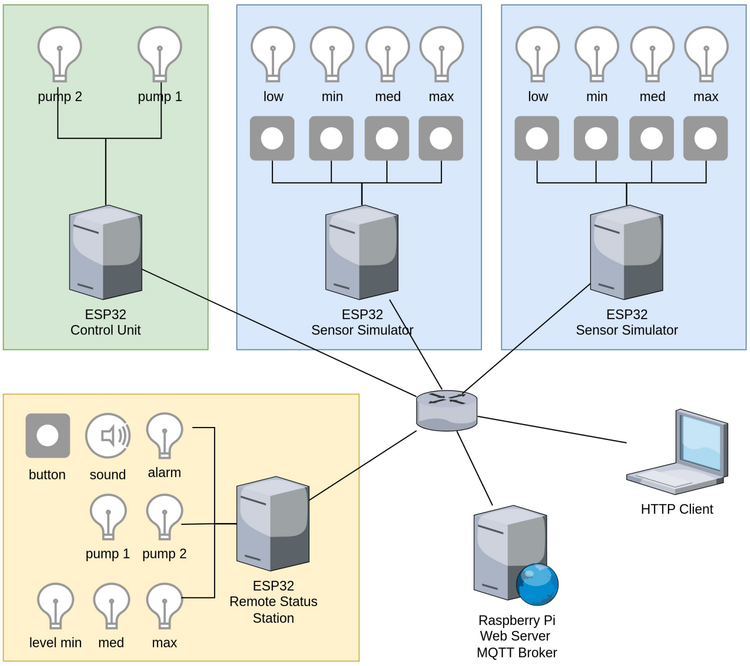


Figure 12: Network diagram