Critical Systems Lab - MESCC
Water Pumping Automated System

Ricardo Mendes 1201779

Arthur Gerbelli 1220201

ISEP, January 2024, **Third Delivery**

Contents

1 Introduction

As this is the last report, we will try to summarize all the previous analyses, with the main focus being to achieve a conclusion and not only repeat the specificities of the last reports.

Although the following chapters are tightly defined, the report should be read as a whole.

2 Requirement Specifications

2.1 System Requirements

There was no change in the system requirements. Below, we listed all the identified system requirements and a comment about its implementation on the prototype.

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

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

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

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

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

.6: If the readings of the sensor are uneven, the system shall choose the worst case scenario, following the table below: Implemented

sensor #2 \ sensor #1	**0**	**1**	**2**	**3**	**4**
0	-	1	2	3	4
1	1	1	1	1	4
2	2	1	2	2	4
3	3	1	2	3	4
4	4	4	4	4	4

1: below min; **2**: above min; **3**: above med; **4**: above max.

0: no connection to the sensor - if both sensors are unavailable, the alarm shall be triggered.

SR-2 .1: The status of all WPS shall be displayed on all RSS. Partially Implemented: the hadware available was not enough to add a second WPS and RSS.

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

.3: The RSS shall have an independent power supply from the WPS. Not Implemented

.4: The alarm on the RSS shall have an independent power supply from the RSS itself and from the WPS. Not Implemented

SR-3 .1: The status of all WPS shall be visible on a web page. Implemented

SR-4 .1: To improve the system´s communication reliability, a cluster of 2(two) MQTT brokers shall be deployed. Not Implemented

2.2 Hazard Analysis

We added one more entry to the hazard analysis, H-9. Below, we added some comments about the implementation of the identified mitigations in the prototype.

H-1: see SR-1.5

H-2:
- **Description:** Both pumps stopped working.
- **Mitigation:** Trigger alarm. Implemented

H-3:
- **Description:** A pump doesn't turn OFF when the water level in bellow minimum.
- **Mitigation:** Trigger alarm. Not Implemented

H-4: see SR-1.6

H-5: see SR-2.3 and SR-2.4

H-6:
- **Description:** RSS are not getting information from WPS.
- **Mitigation:** Implement a cluter of MQTT Brokers or remove this single point of failure by adopting DDS. Trigger alarm. Partially Implemented: the alarm is triggered

H-7:
- **Description:** RSS stops working.
- **Mitigation:** Have redundancy by having multiple RSS and each one displaying all statuses from all WPS. Not Implemented

H-8:
- **Description:** Control Unit stops working.
- **Mitigation:** Implement redundancy by having a cluster of nodes running the Control Unit. If the number of nodes is 3 we can implement a voting system and run the same process with the same input in parallel. This would improve the system's fault tolerance. Not Implemented

H-9:
- **Description:** Rapid wear of the first water pump.
- **Mitigation:** Distribute evenly the work done by the pumps. This can be done, by choosing a random pump when one is need first. Not Implemented

2.3 System Structure and Traceability

This last chapter tries to map the implementation of the prototype to the model developed during the previous two deliveries.

During the implementation of the system, we noticed that the Web Server is a subsystem independent of the RSS.

Although, both are ways to visualize the WPS status, they have very different objectives and levels of criticality. The RSS must alert the maintenance team; the web

version is, in our understanding, only a nice thing to have. And most of all, we can remove one service without affecting the other. Because of this, we made some changes to the System Structure diagram, as it is possible to visualize below.

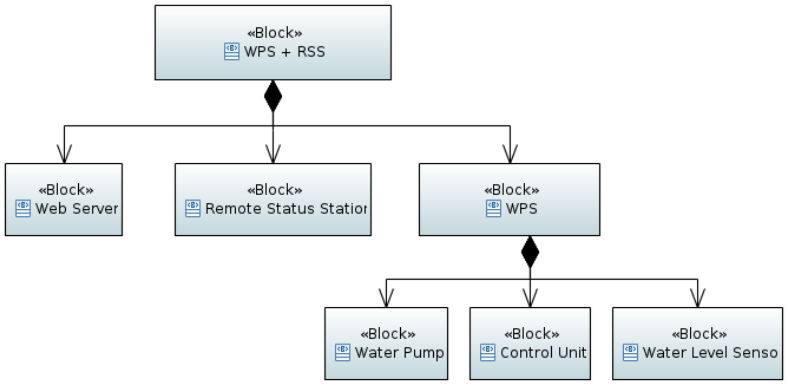

Figure 1: System Structure diagram

Looking at the Deployment diagram for the implementation of the prototype, we can clearly see that the output of the modeling traces nicely to the implementation. The grayed parts of the diagram don't make up the prototype.

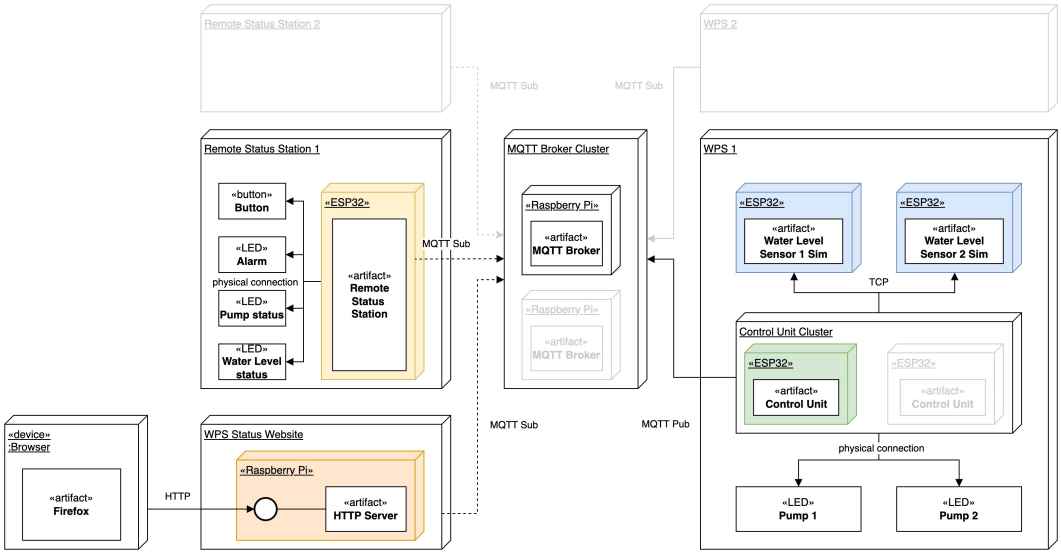

Figure 2: Deployment diagram

3 Implementation

3.1 CCSYA - Assembly

- mostrar codigo da implementacao - um dos comentarios no codigo de assembly está mal (beq)

3.2 RTAES - Concorrency and Real Time Scheduling

3.2.1 Real Time Scheduling

Table 1: Theoretical values (left) and implemented values (right) in *ms*:

Task	Ci	Ti
1	60	100
2	25	200
3	35	250

Task	Ci	Ti
1	1200	2000
2	500	4000
3	700	5000

The values on the right side table were achieved by running the code in different situations. A closer look, shows that we incresed the values **x 20**.

To track the execution time we added the following code to each task:

```
for (;;) {
    unsigned long start_time = millis();
    unsigned long finish_time;
    unsigned long duration;

    // Code --------------------

    finish_time = millis();
    duration = finish_time - start_time;
    Serial.print("TASK N - Execution Time[ms]: ");
    Serial.println(duration);
}
```

And here is some of the outputs:

```
TASK 3 - Duration[ms]: 40
TASK 1 - Duration[ms]: 917
TASK 2 - Duration[ms]: 475
TASK 1 - Duration[ms]: 658
TASK 1 - Duration[ms]: 754
TASK 3 - Duration[ms]: 770  < above WCET
TASK 2 - Duration[ms]: 476
TASK 1 - Duration[ms]: 649
TASK 1 - Duration[ms]: 799
TASK 3 - Duration[ms]: 29
TASK 2 - Duration[ms]: 477
```

```
TASK 1 - Duration[ms]: 176
TASK 1 - Duration[ms]: 997
TASK 2 - Duration[ms]: 481
TASK 1 - Duration[ms]: 187
TASK 3 - Duration[ms]: 157
TASK 1 - Duration[ms]: 990
TASK 2 - Duration[ms]: 475
TASK 1 - Duration[ms]: 1036
```

The execution time of task 2 that was calculated in the last report is very close to the measured values. Because of this, we needed to add a sleep function to increase its execution. However, task 1 and task 3 execution time was poorly estimated. This is because of the latency during the TCP and MQTT requests.

To simulate the conclusions from the schedulability analysis, we also increased the periods - as visible in the table 1.

Also, to control that the tasks didn't extended beyond its deadline, we added the following code:

```
long next_release = 5000 - duration; // 5 seconds deadline for Task 3
if (next_release > 0) {
    vTaskDelay(next_release / portTICK_PERIOD_MS);
} else {
    Serial.println("TRASK 3 - FAILED Deadline !!!");
}
```

In some cases, task 1 failed the deadline by a couple of miliseconds. Because we defined a timeout value of 2 second, when the request to the two sensors fail, we get this result.

```
TASK 1 - Connection failed: 172.20.10.13
TASK 1 - Connection failed: 172.20.10.5
TASK 1 - Duration[ms]: 2009
TASK 1 - FAILED Deadline !!!
```

connection timeout = 1s that gives a lot of timeouts. specially with one of the sensors mitigation.

3.2.2 Concorrency

- mostrar codigo da implementacao
    - onde mutex

3.3 COMCS - Communication

- TCP pull strategy
    - Arthur

3.4 Prototype

3.4.1 Overview

- diagram if implementation

3.4.2 WPS

- explicar a tabela dos inputs dos sensores e levantar alarm caso (A tabela agora é diferente da ultima vez) - strategy for cluster of control unit nodes
    - dois timeouts dos sensores equivale a max level.... nao devia estar assim

3.4.3 MQTT Broker

- one topic per WPS

3.4.4 RSS

- reads broker every 1 sec - sends alert after 15 sec without message - subscribes to each WPS topics (iter) - log who clicked the button and maintain LED on... because the system heals it self

3.4.5 Web Server

4 Team Work

References

[1] Espressif documentation: https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/system/freertos_idf.html#id23/