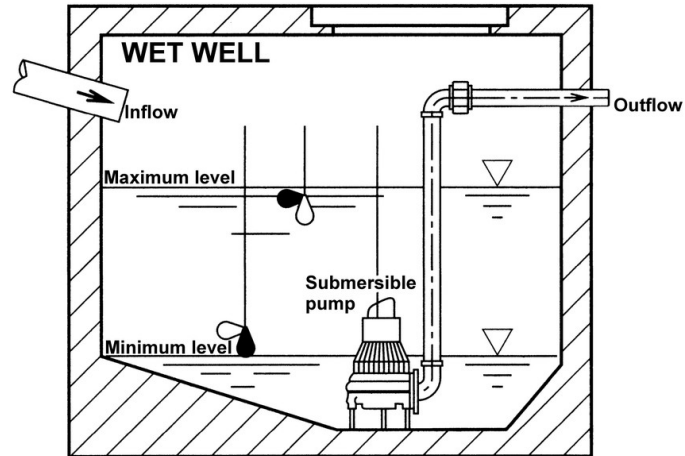


# MESCC – CSLAB

## Water Pumping System (WPS) + Remote Status Station (RSS)

Ricardo Mendes & Arthur Gerbelli



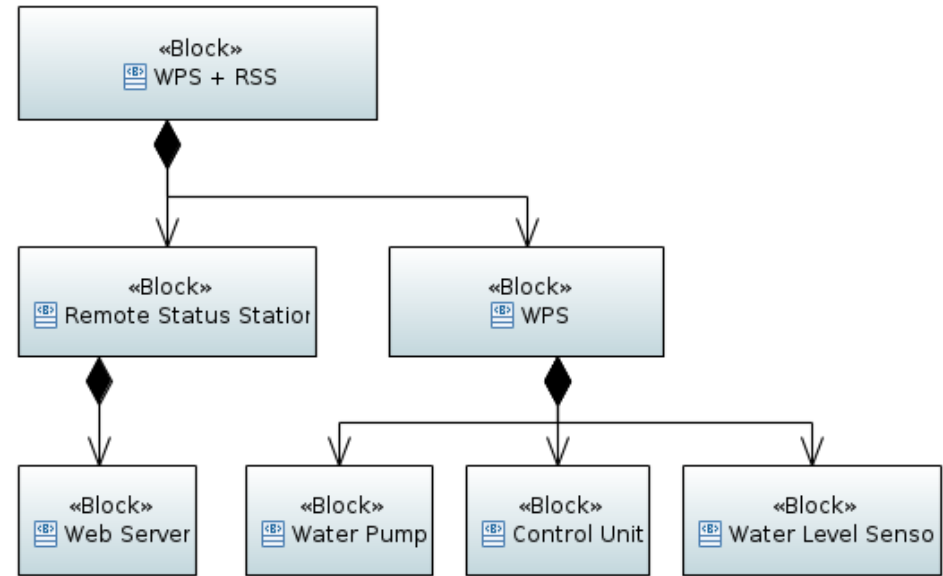
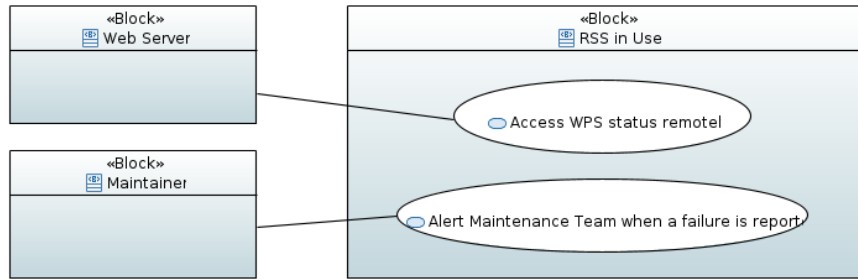
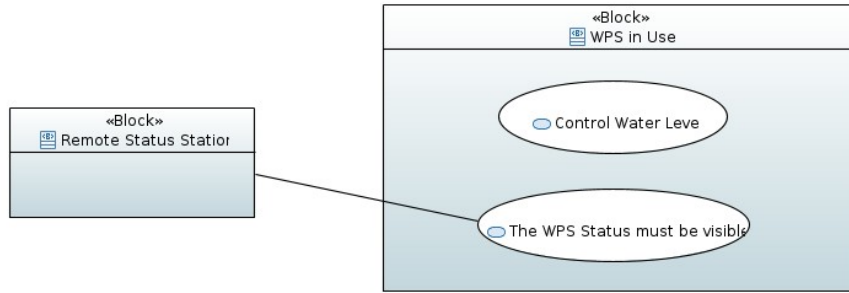
# CONTENT

- Requirement Specification [Update]
- Implementation:
  - Concurrency and RealTime Scheduling
  - Communication Infrastructure
  - Implementation using Assembly
- Demo

## **REQUIREMENTS SPECIFICATION – Stakeholder Needs**

- **SN-I.3** Every WPS will have two pumps and two water level sensors to achieve a certain level of redundancy and reliability on the system.
- **SN-I.4** To improve the system's performance, and given that we have one unused water pump, this pump should be used when the water level is above  $\frac{2}{3}$  of the well max capacity.
- **SN-I.5** In case that only one water pump is operational, the max capacity of the WPS shall be reduced.

# REQUIREMENTS SPECIFICATION – Use Cases & System's Structure



## REQUIREMENTS SPECIFICATION – Hazards Analysis

- **Description:** The two level sensors give contradictory readings.
  - **Cause:** Sensor malfunction, connection issues.
  - **Effect:** Inappropriate system behavior.
  - **Mitigation:** 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. Trigger the alarm if the system is unable to achieve a consensus.
- 
- **Description:** Control Unit stops working.
  - **Cause:** Malfunction, bug.
  - **Effect:** Total failure of the system.
  - **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.

## REQUIREMENTS SPECIFICATION – Requirements Update

- 1.4: A second pump shall be turned on only when the water level is above  $\frac{2}{3}$  the maximum water level.
  - 1.5: When only one pump is available, the maximum water level shall be reduced to  $\frac{2}{3}$ .
  - 1.6: If the readings of the sensor are uneven to a level of 20cm, the system should choose the worst case scenario.
- 
- 2.3: The RSS shall have an independent power supply from the WPS.
  - 2.4: The alarm on the RSS shall have an independent power supply from the RSS itself and from the WPS.
- 
- 4.1: To improve the system's communication reliability, a cluster of 2 MQTT brokers shall be deployed.

## IMPLEMENTATION – Concurrency and Real Time Scheduling

Hardware specification for an ESP-WROOM-32:

- Clock rate: 80 to 240 MHz
- Clock cycle per instruction (CPI): 2 clock cycles — for instructions following ALU and branch instructions. 4 clock cycles — in other cases.

```
$ avr-objdump -m avr -S control-unit.ino.cpp.o | wc -l  
output -> 448
```

$$CPU\ Time = \frac{Instruction\ count * CPI}{Clock\ Rate}$$

$$CPU\ Time = \frac{448 * 4}{80000} = 0,0224\ seconds$$

## IMPLEMENTATION – Concurrency and Real Time Scheduling

The Control Unit will run three tasks:

	Task	Ci	Ti
• T <sub>1</sub> : Retrieve data from sensors (TCP Client);	1	60	100
• T <sub>2</sub> : Process data and give instructions to pumps;	2	25	200
• T <sub>3</sub> : Publish WPS status to MQTT broker (MQTT client).	3	35	250

... and **share one specific resource**: a data structure with the WPS status, pump status, water level status and alert status.



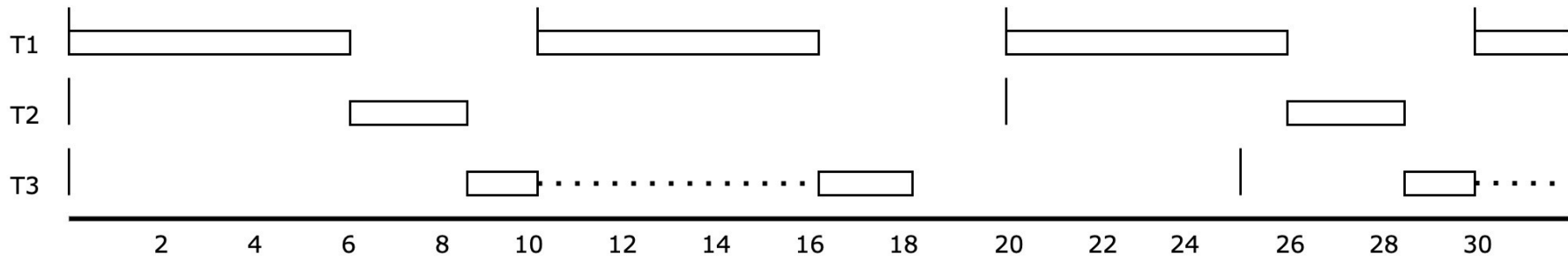
## IMPLEMENTATION – Concurrency and Real Time Scheduling

$$U = \frac{60}{100} + \frac{25}{200} + \frac{35}{250} = 0,865$$

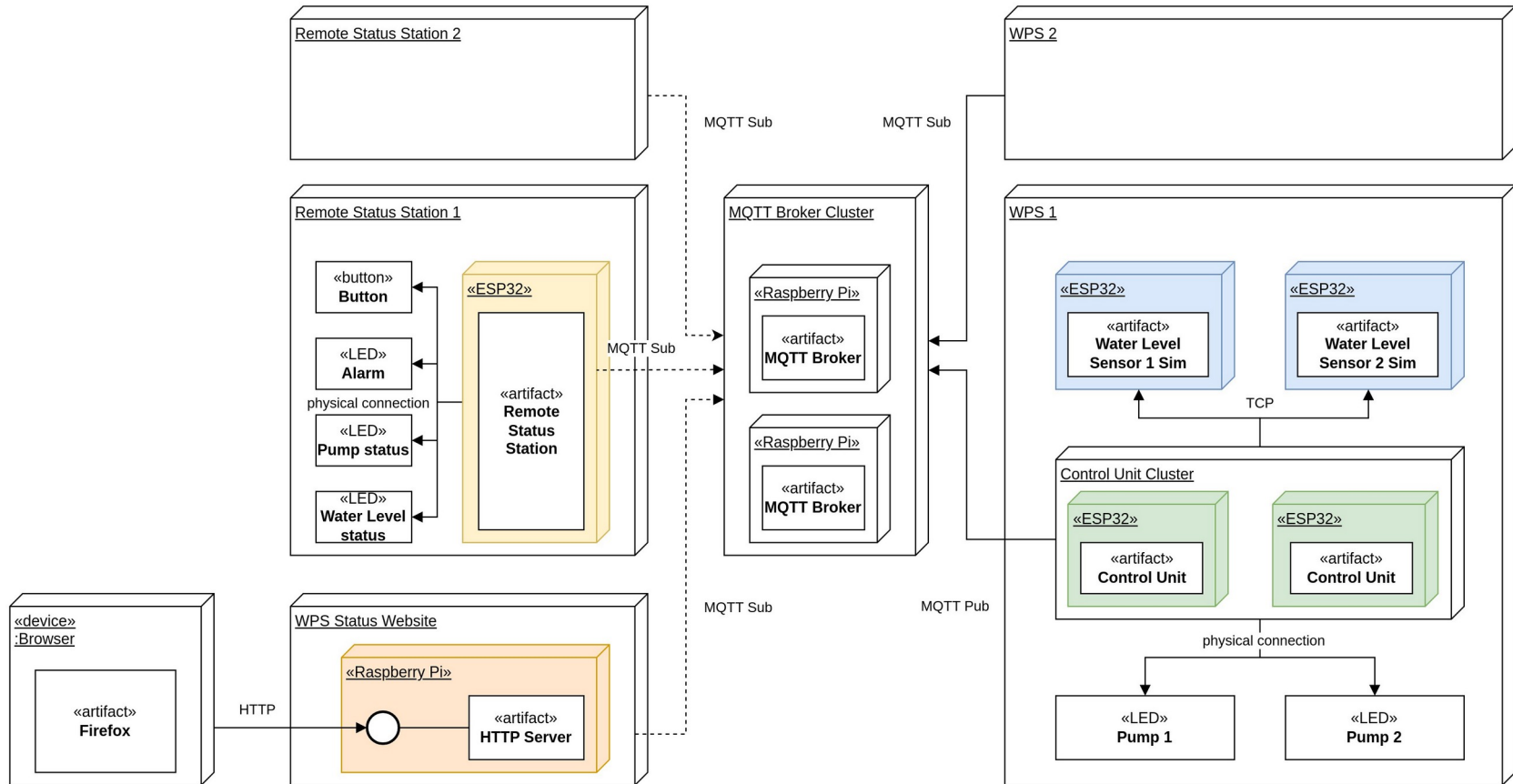
CPU utilization doesn't exceed 100%  
but cannot confirm that it is schedulable!

$$LUB = 3(2^{\frac{1}{3}} - 1) \approx 0,78$$

$$1 \geq 0,865 \geq 0,78$$



# IMPLEMENTATION – Communication Infrastructure



# IMPLEMENTATION – Assembly Implementation

```
.global getMask

getMask:    entry a1, 48
            movi a5, 0          # init increment
            mov a6, a2          # save inputed water level
            movi a2, 1          # init return value

loop:
            addi a5, a5, 1      # increment by one
            beq a5, a6, end     # branch to 'end' if not equal
            slli a2, a2, 1      # left shif by 1, i.e. 0001 -> 0010
            j loop              # jump to loop

end:
            retw.n
```

```
void implementMask(int8_t mask)
{
    int8_t n_bit = 0;
    while (n_bit < MAX_LEVELS) {
        if (mask & 0x01) {
            digitalWrite(STATES[n_bit], HIGH);
        }
        else {
            digitalWrite(STATES[n_bit], LOW);
        }
        n_bit++;
        mask = mask >> 1;
    }
}
```

**DEMO**

