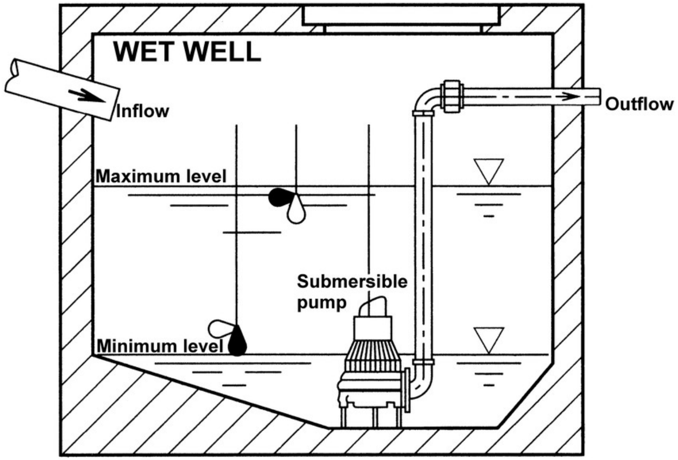
MESCC – CSLAB

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

Ricardo Mendes & Arthur Gerbelli



CONTENT

- **Requirement Specification**
  - System requirements
  - Hazard Analysis
  - Traceability

- **Implementation:**
  - CCSYA - Assembly
  - RTAES - Concurrency and RealTime Scheduling
  - COMCS - Communication Infrastructure

- **Prototype**

REQUIREMENTS SPECIFICATION – System Requirements

WPS:

- SN-1.1 ✓
- SN-1.2 ✓
- SN-1.3 ✓
- SN-1.4 ✓
- SN-1.5 ✓
- SN-1.6 ✓

RSS:

- SN-2.1 ✗ The status of all WPS shall be displayed on all RSS.
- SN-2.2 ✓
- SN.2.3 ✗ The RSS shall have an independent power supply from the WPS
- SN.2.4 ✗ The alarm on the RSS shall have an independent power supply

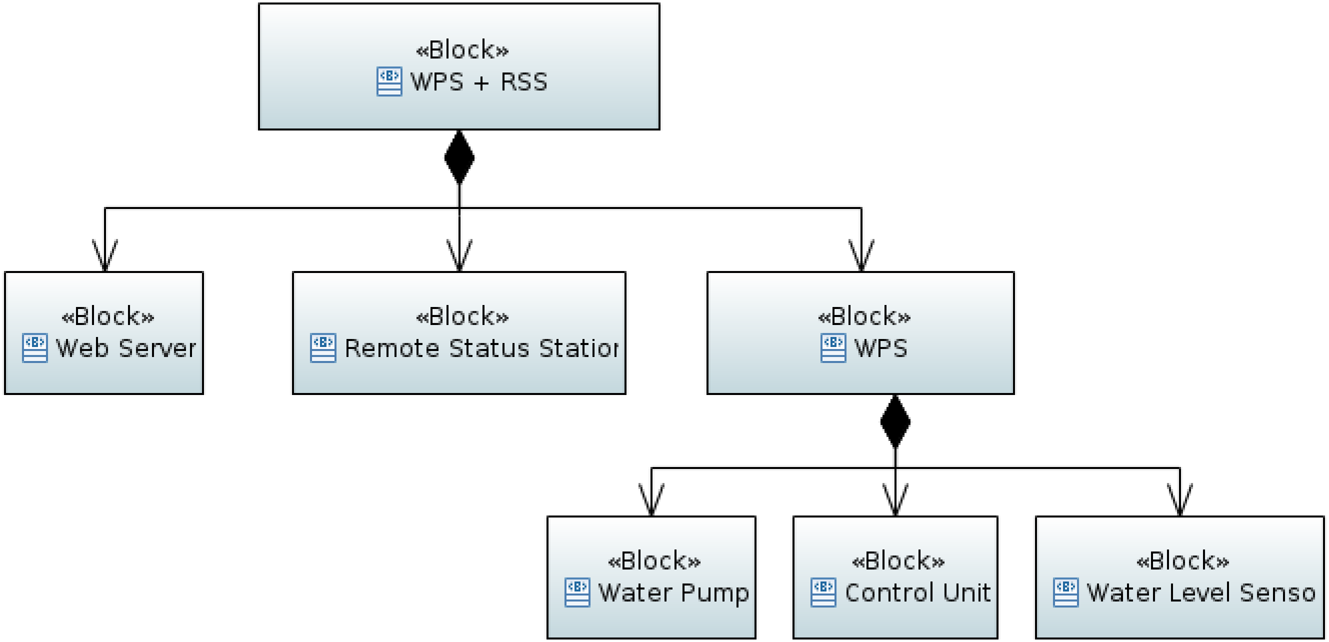
Web Server:

- SN-3.1 ✓

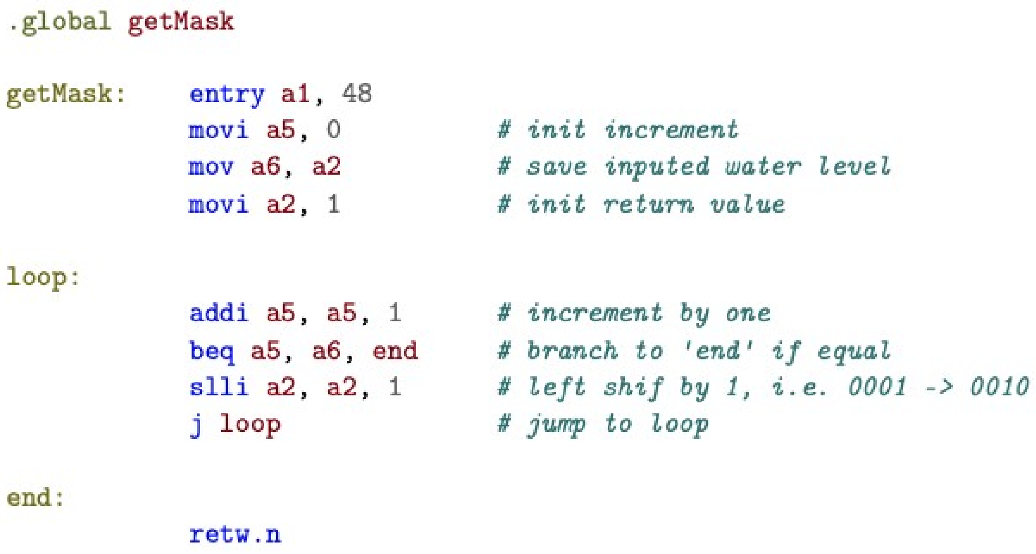
REQUIREMENTS SPECIFICATION – Hazards Analysis

- H-1 ✓
- H-2 ✓
- H-3 ✗ A pump doesn't turn OFF when the water level in bellow minimum.
- H-4 ✓
- H-5 ✗ Independent power supplies
- H-6 ✓
- H-7 ✗ RSS stops working.
- H-8 ✗ Control Unit stops working.
- H-9 ✗ Rapid wear of the first water pump.

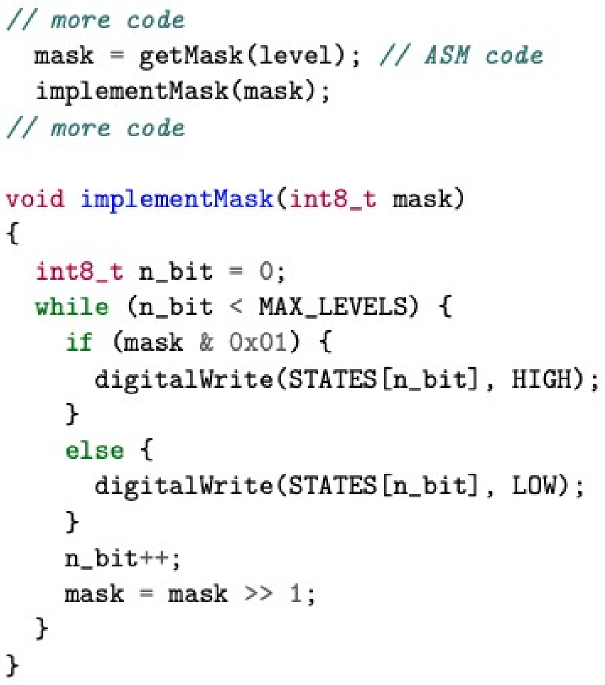
REQUIREMENTS SPECIFICATION – Traceability



IMPLEMENTATION – CCSYS Assembly

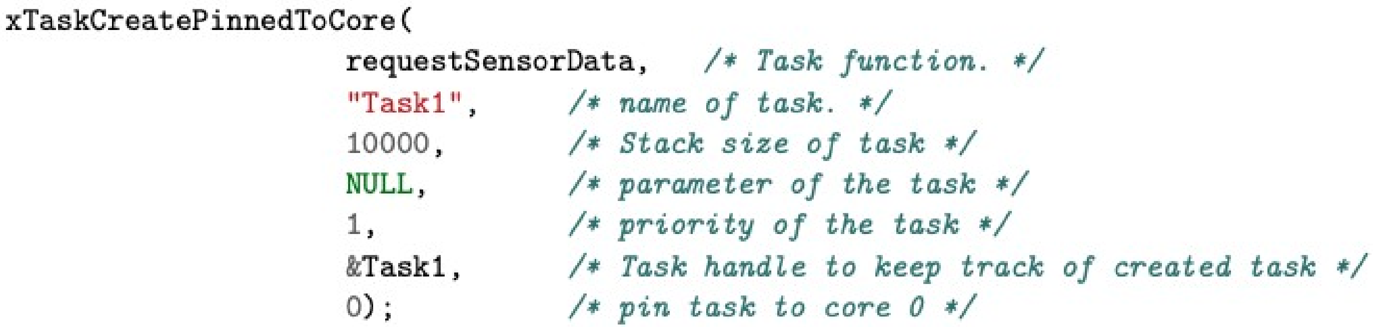


```
.global getMask

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 equal
            slli a2, a2, 1      # left shif by 1, i.e. 0001 -> 0010
            j loop              # jump to loop

end:
            retw.n
```



```
// more code
  mask = getMask(level); // ASM code
  implementMask(mask);
// more code

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;
  }
}
```

IMPLEMENTATION – RTAES RT Scheduling



```
xTaskCreatePinnedToCore(
                    requestSensorData,   /* Task function. */
                    "Task1",        /* name of task. */
                    10000,          /* Stack size of task */
                    NULL,           /* parameter of the task */
                    1,              /* priority of the task */
                    &Task1,         /* Task handle to keep track of created task */
                    0);             /* pin task to core 0 */
```

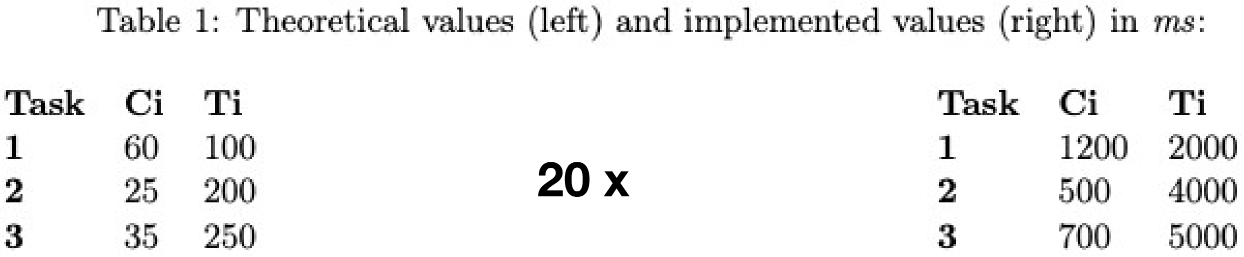


Table 1: Theoretical values (left) and implemented values (right) in *ms*:

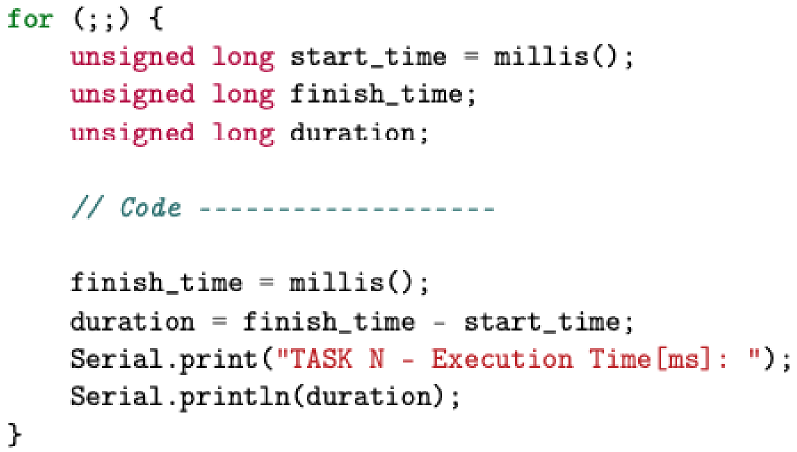
Task	Ci	Ti
1	60	100
2	25	200
3	35	250

20 x

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

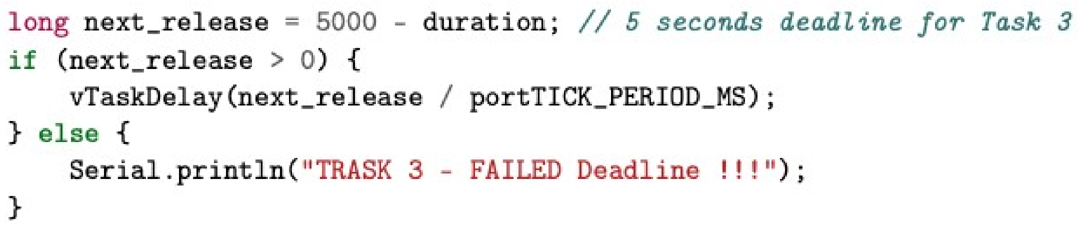
IMPLEMENTATION – RTAES RT Scheduling

Calculate Execution time:



```
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);
}
```

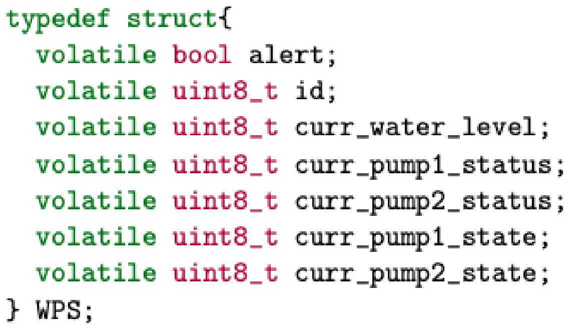
Warning when deadline fails:



```
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 !!!");
}
```

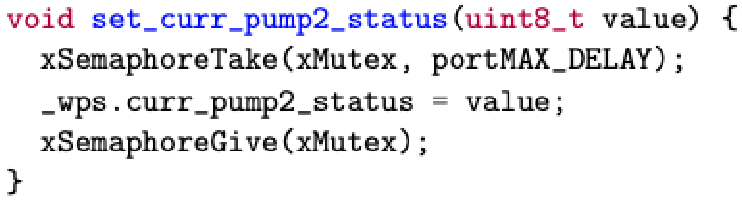
IMPLEMENTATION – RTAES Concurrency

WPS status:



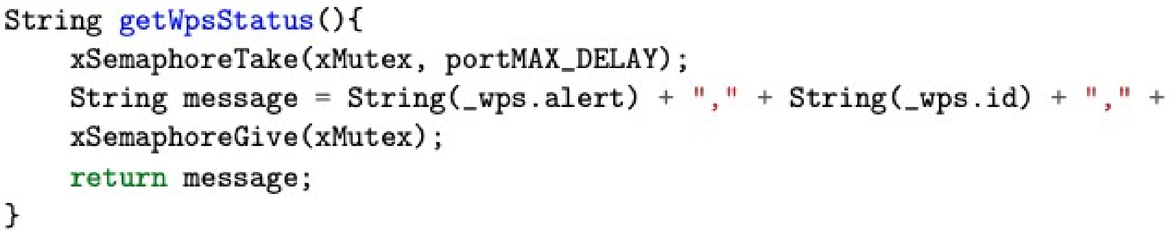
```
typedef struct{
  volatile bool alert;
  volatile uint8_t id;
  volatile uint8_t curr_water_level;
  volatile uint8_t curr_pump1_status;
  volatile uint8_t curr_pump2_status;
  volatile uint8_t curr_pump1_state;
  volatile uint8_t curr_pump2_state;
} WPS;
```

Update WPS status:



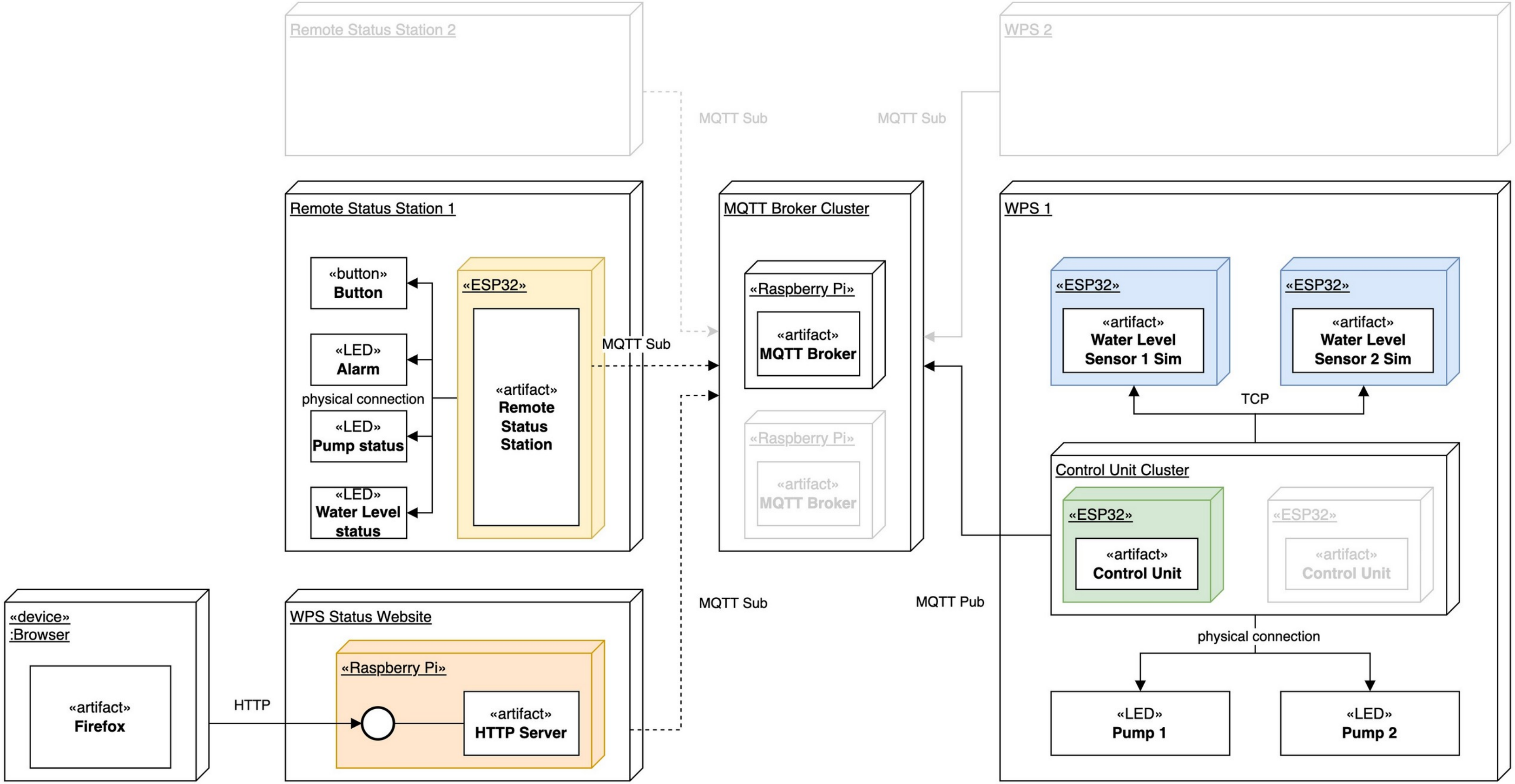
```
void set_curr_pump2_status(uint8_t value) {
  xSemaphoreTake(xMutex, portMAX_DELAY);
  _wps.curr_pump2_status = value;
  xSemaphoreGive(xMutex);
}
```

Read WPS status:

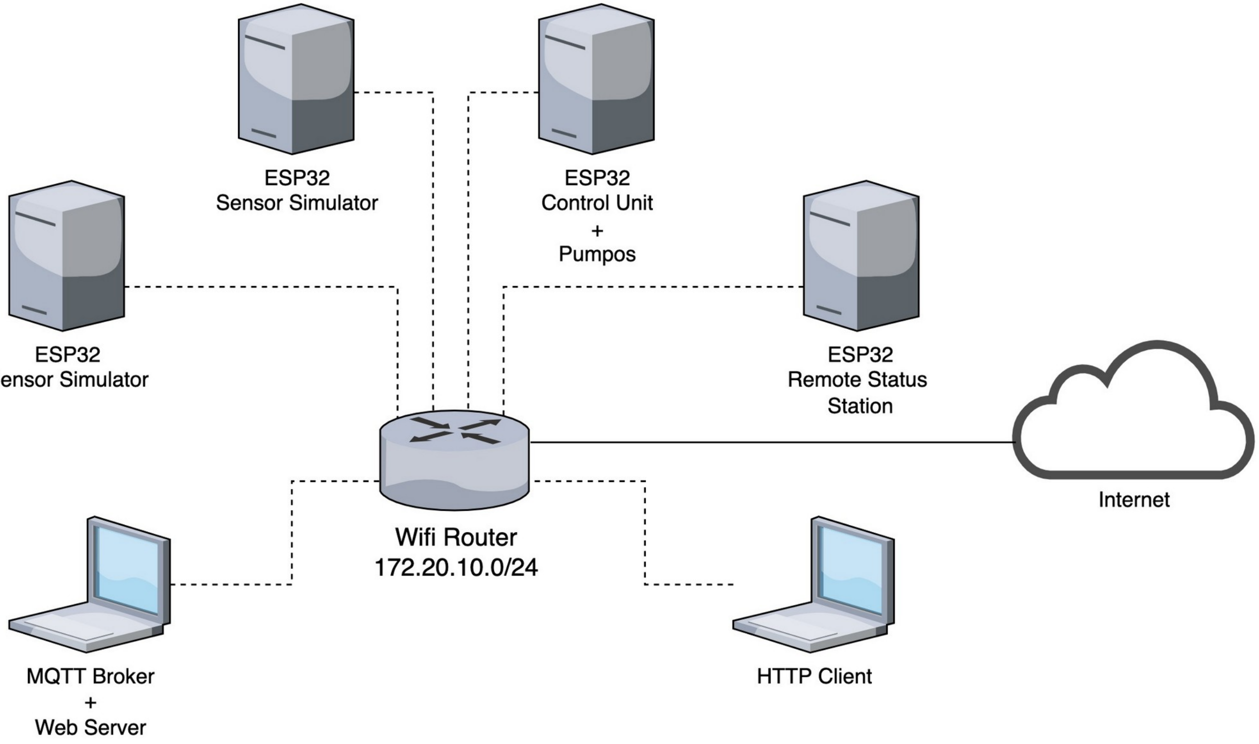


```
String getWpsStatus(){
    xSemaphoreTake(xMutex, portMAX_DELAY);
    String message = String(_wps.alert) + "," + String(_wps.id) + "," +
    xSemaphoreGive(xMutex);
    return message;
}
```

IMPLEMENTATION – COMCS Communication



PROTOTYPE – Network diagram



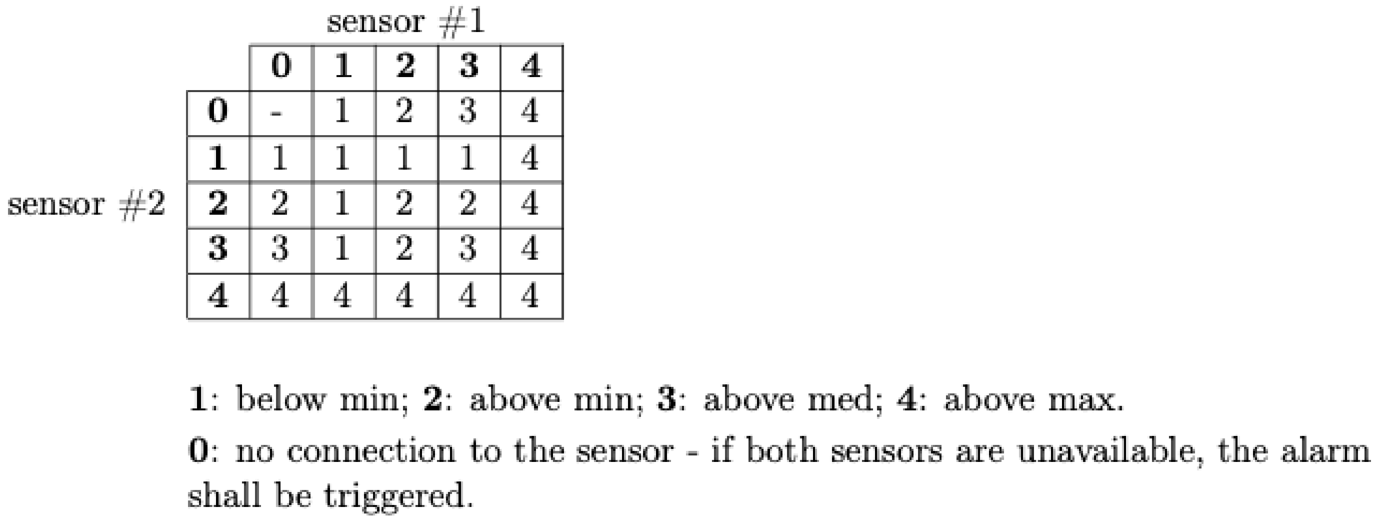
PROTOTYPE – Test Cases

- **SR-1.1** While the water level is above the minimum level, WPS shall have a pump working.
- **SR-1.2** When the water level is below the minimum level, WPS shall have all pumps stopped.
- **SR-1.3** If the water level is above the maximum level, then the WPS shall trigger an alarm at the Remote Status Station (RSS).
- **SR-1.4** A second pump shall be turned on only when the water level is above 2/3 the maximum water level.

PROTOTYPE – Test Cases

- **SR-1.5** When only one pump is available, the maximum water level shall be reduced to 2/3.

- **SR-1.6** If the readings of the sensor are uneven, the system shall choose the worst case scenario, following the table below:



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

PROTOTYPE – Test Cases

- **SR-2.2** If the alarm is ON, the button in the RSS shall only disable it.
- **SR-3.1** The status of all WPS shall be visible on a web page.
- **H-2** Both pumps stopped working.
- **H-6** RSS are not getting information from WPS.