Smart River Monitoring

Ameri Javid, D'antino Matilde, Diotallevi Federico

IoT 2023/2024

Contents

1 System design

The Smart River Monitoring System is an IoT system implemented using Arduino and ESP32. The project's goal is to simulate a system whose principal components are:

- Water Level Monitoring Subsystem (based on ESP)
  - Embedded system to monitor the water level of a river
  - Interacts with the River Monitoring Service subsystem via MQTT
- River Monitoring Service Subsystem (backend - running on a PC server)
  - Service functioning as the main unit governing the management of the smart river system
  - Interacts through the serial line with the Controller
  - Interacts via MQTT with the Water Level Monitoring
  - Interacts via HTTP with the Dashboard
- Water Channel Controller Subsystem (Arduino)
  - Embedded system controlling the gate/valve of a water channel
  - Interacts via serial line with the River Monitoring System
- River Monitoring Dashboard Subsystem (Frontend/web app on the PC)
  - Front-end to visualize and track the state of the river monitoring
  - Interacts with the River Monitoring Service

1.1 User Modes

The River Monitoring System uses a user mode that can be either manual, automatic, or remote. Each system's component has to adapt to the user mode, in particular:

- In auto mode, the service communicates to the water channel controller the valve opening level based on the water level measurements.
- In manual mode, the user can manually set the valve opening level. Note that this mode has priority over the others.
- In remote mode, the user can set the valve opening level using a dashboard provided by the system.

1.2 Alarm States

The River Monitoring System has different alarm states based on the water level measured:

- When the water level is between 0cm and 6cm, the system is considered to be in ALARM TOO LOW STATE.
- When the water level is between 6cm and 16cm, the system is considered to be in NORMAL ALARM STATE.
- When the water level is between 16cm and 22cm, the system is considered to be in PRE ALARM TOO HIGH STATE.
- When the water level is between 22cm and 30cm, the system is considered to be in ALARM TOO HIGH STATE.
- When the water level is over 30cm, the system is considered to be in ALARM TOO HIGH CRITIC.

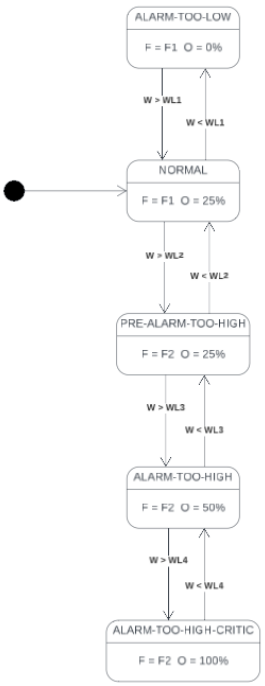

Figure 1: Alarm States

2 Water Level Monitoring Subsystem

The Water Level Monitoring subsystem is responsible for continuously monitoring the level of the water. It performs the following tasks:

- Monitoring the water level.
- Sampling the water level and sending it to the River Monitor Service with a specific frequency.
- Indicating network problems .

2.1 Water Level Measurements

The ESP32 performs water level measurements using a sonar sensor, which is passed to the FreeRTOS function that creates a task in the operating system. The frequency at which measurements are taken is determined by the backend.

2.2 Communication with Water Monitoring Service

Communication with the backend occurs through the MQTT protocol using the broker "broker.mqtt-dashboard.com". The Water Level Monitoring Subsystem connects to the WiFi and subscribes to the topics "water_level" and "frequency":

- Through the "water_level" topic, the water level monitoring subsystem publishes the water levels recorded by the sonar. The messages are read by the backend.
- Through the "frequency" topic, the water monitoring service publishes the frequency at which the sonar should repeat the measurement. The frequency is calculated based on the water level recorded by the sonar. The frequency is read by the water level monitoring subsystem.

In case of WiFi connection problems or issues with communication, the LED on the ESP32 will be red. If it is successfully connected to the network and the MQTT broker, the LED will be green.

3 River Monitoring Service

The river monitoring service decides the overall policy and behaviour of the river monitoring system, depending on the water level as measured by the Water Level Monitoring subsystem.

3.1 System Architecture

The RiverMonitoringService is the controller of the system. It is responsible for orchestrating all hardware components of the system in a consistent manner with the system state, using appropriate protocols to:

- Communicate with ESP32 to obtain water level measurements and update the frequency of water level measurements based on the system state.
- Communicate with Arduino to update the valve opening level and be notified when the system is switched to manual mode.
- Communicate with the Dashboard displaying the latest available data and handling remote control requests.

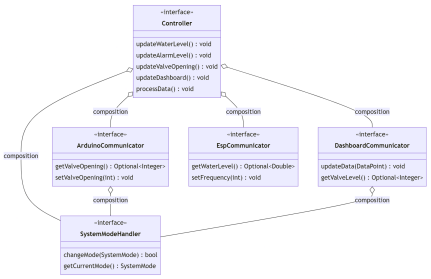

Figure 2: River Monitoring Service UML

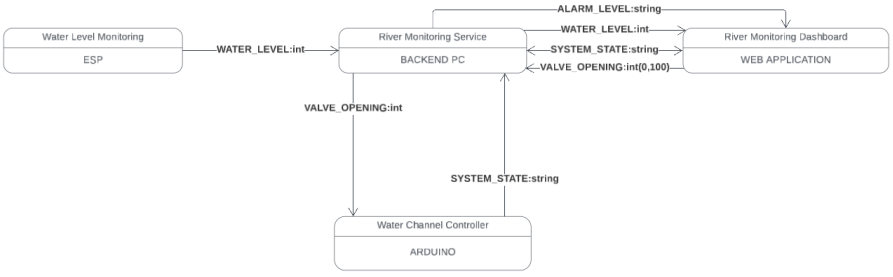

Figure 3: Message schema

3.2 Communication with Arduino

Communication with Arduino is managed via serial communication implemented using libraries provided in the course labs. This communication runs on a dedicated thread responsible for sending the correct valve opening levels when in remote or automatic mode and handling requests for manual mode.

3.3 Communication with ESP32

Communication with the ESP32 is managed using the MQTT protocol, employing the Vertx library[1].
For MQTT communication, the broker `broker.mqtt-dashboard.com` is used along with two distinct topics:

- **water_level**: where the backend acts as a subscriber, reading the water level measurements published by the ESP.
- **frequency**: where the backend acts as a publisher, posting the reading frequencies of the water level consistent with the system.

3.4 Communication with the Dashboard

Communication with the Dashboard is managed using the HTTP protocol, implemented through the Vertx library[1]. An HTTP server is created to handle external requests from the Dashboard via various HTTP methods to three different paths:

- **/api/data**: to receive a JSON containing the current data in the system or a 404 error if the system is unable to retrieve data to send.
- **/api/mode**: to request a switch from AUTO mode to REMOTE mode or vice versa. The handler returns a JSON containing the mode set at the end of the request along with a status code 200 if the change was successful, or 403 if the change was forbidden.
- **/api/valve**: to set the level of valve opening. In this case, the system expects a JSON in the request body containing the field VALVE_OPENING and its corresponding opening value. The request ends with a status code 400 if the parsing of the value was not performed correctly.

The server listens on port 8080 to handle these HTTP requests.

[1] See `https://vertx.io`

4 Water Channel Controller Subsystem

The Water Channel Controller subsystem is responsible for controlling the Arduino part with all its devices, more specifically the valve, establishing how much water should flow to the channels, and this is determined by the valve opening-level: from full closed (0), up to full open (100). It performs the following tasks based on the three different states of the system (AUTOMATIC, REMOTE, MANUAL):

- Setting the opening-level of the water valve by communicating with the backend (this happens in AUTOMATIC and REMOTE modes).
- Setting the opening-level of the water valve by using a potentiometer (this happens in MANUAL mode).
- Switching the state from AUTOMATIC/REMOTE to MANUAL by pressing a button.
- Displaying the current state of the program on the LCD Display.

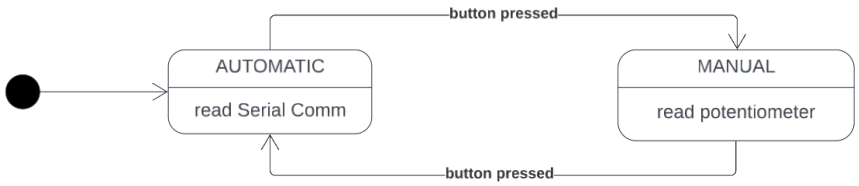

Figure 4: State of Water Channel Control Task

4.1 AUTOMATIC/REMOTE mode

The subsystem receives messages, from the backend, in which it's written the value of the opening-angle of the water valve, extracts this value and maps it into an int, it then calls the specific function to open the valve of that exact value.

4.2 MANUAL mode

The subsystem switches to this mode when the button is pressed. During this mode the opening-level of the valve is controlled by a potentiometer. If the button is pressed again, the subsystem switches back to the AUTOMATIC/REMOTE mode.

4.3 DISPLAYING

During the entire program the LCD Display displays the current state (mode) in which the system is. So the text is updated every time the subsystem switches mode.

5 River Monitoring Dashboard Subsystem

The river monitoring dashboard has two main responsibilities:

- To visualize the state of the River Monitoring system.
- To allow a user for controlling manually, from remote, the valve opening level.

5.1 Frontend

The River Monitoring Dashboard Subsystem is implemented through a web application that displays a chart, implemented using the Chart.js library provided by JavaScript, illustrating the water level trend. Additionally, it displays the current system status and valve opening. The second part of the web page features a button to switch to remote mode, which is enabled only when the system mode is not set to "MANUAL". When the system mode is set to "REMOTE", a slider bar is then displayed, allowing control over the valve opening.

5.2 Communication with backend

Communication with the Water Monitoring Level Service occurs through the use of the HTTP protocol. The dashboard updates information on the water level trend and system status every "frequency" seconds (in our case, frequency = 5 seconds) by sending an HTTP request to the backend to obtain the latest available data in JSON format, which is retrieved using the path api/data. If the "Remote Control" button is pressed, the backend is notified via an HTTP request using the path api/mode. If the backend determines that entering the requested mode is not possible, an error message will be returned. To communicate the valve opening during the remote mode an HTTP request is sent using the path api/valve.

6 Schema

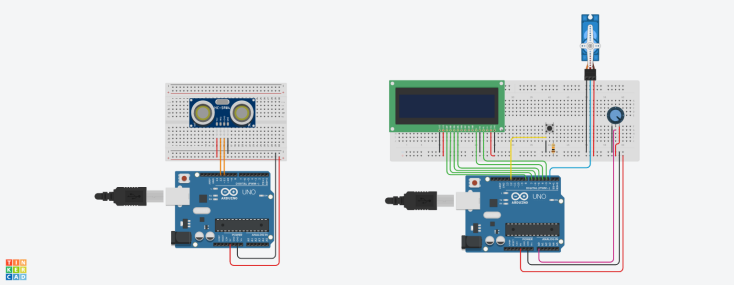

Figure 5: TinkerCard Arduino and ESP32 Schema