UNIVERSITÉ CÔTE D'AZUR
SI5/M2 INFO
2024-2025

Julien Deantoni, Nicolas Ferry

Architecting IoT Systems, Beyond Functional Correctness

Smart Vineyard Farming

⚠ May be refined in the next few weeks

Project delivery

This project concerns the delivery of the *Architecting IoT Systems, Beyond Functional Correctness* course. It entails the development of a comprehensive system designed to automate the management of a smart vineyard. Teams will be responsible for delineating the specific operational environment for the system. In addition to ensuring functional correctness, participants are expected to discern potential threats (to any extra-functional concerns - eg. performances, scalability, energy) and proactively devise mitigation strategies for their solutions. It is important to note that no a priori extra-functional requirements have been specified for this project, necessitating teams to independently elicit and address them.
Deliveries are expected by email (to Julien Deantoni: firstname.lastname@univ-cotedazur.fr, with [IoT_BFC] as object prefix) followed by "team X project" where X is the name of your team (as used in the slack dedicated channel). The delivery is expected **before** the 19^{th} of January 2025 at 10:00PM Paris Time. The delivery is expected as a PDF paper That follows classical scientific papers format.
The paper must contain :

- the name of the members of your team
- a link to the code of your system (typically a link to the git repository. Note that this git should clearly explain how to setup and use the project)
- an introduction section specifying the context of use of the system, its functional requirements and what are the extra functional requirements you elicited as being the most important; together with an explanation of why (to be further detailed in the "proposed solution" section).
- a critical description of existing solution ("state of the practice" section); with the pros and cons of each identified solution;
- a "proposed solution" section specifying:
  - the differentiating extra functional requirements you elicited ;
  - the main risks you identified and how you mitigated them. It can be done by rationalizing the choices you did in your architecture. Note that the architecture should be specified and rationalized in terms of:
    1. the application architecture;
    2. the hardware architecture;
    3. the deployment specification;

  - a critical analysis of your own solution, specifying what you did right and what could be improved (and how)

- An "implementation and result" section highlighting how and why your solution is actually a good one (or not)
- A conclusion resuming the main pros and cons of your architecture, the responsibility of each member in the team with respect to the delivered project; as well as prospective on potential evolution.

1 Project Description: Smart Vineyards Management System

1.1 Objective

The project addresses a smart farming system to optimize production management in a large vineyard, covering several dozen hectares. The vineyard is not connected to the national electricity network. The goal is to equip each vine plant in the field with programmable modules, which perform essential functions for both monitoring environmental conditions and automating the distribution of resources (such as water, fertilizers, or pesticides). This system aims to improve production efficiency, reduce resource waste, and minimize environmental impact.

1.2 System Overview

At the heart of this system is a digital twin dashboard, a dynamic, virtual replica of the vineyard that mimic the actual field conditions and the modules operational status. Each programmable module is in charge of collecting the desired environmental data, including but not limited to soil humidity, temperature, sunlight exposure, and possibly others such as leaf wetness or how ripe the grapes are.
This digital twin dashboard consolidates all incoming data, visualizing the vineyard's physical environment, pinpointing the exact status of each module, and offering insights into field conditions as if the farmer were inspecting the fields in person. In addition to environmental measurements, the dashboard reflects the health and functionality of each module, indicating any issues or maintenance needs and enabling remote troubleshooting.

1.3 Resource Distribution Strategy

The system includes a controlled distribution mechanism for plant resources (e.g., irrigation, fertilizers). The digital twin dashboard must enact precise control over resource distribution, aligning with actual environmental conditions and weather forecasts. In other words, the quantity and timing of resource distribution must depend on forecasted weather data and actual environmental conditions, ensuring the vineyard receives only what is necessary, thus conserving resources and adapting to natural environmental condition variations. For example, if rain is forecasted, the irrigation modules should reduce or suspend water distribution to prevent overwatering and resource waste. Such strategy requires integrating weather forecast data into the decision-making algorithm and the flexibility to adjust distribution parameters remotely, based on updated predictions and farmers' experience.

1.4 System Architecture Constraints

Given the large area of the vineyard, this project imposes strict requirements on system architecture and non-functional aspects, such as energy consumption, data transmission usage and reliability or fault tolerance. The system must also be designed to work autonomously for long periods, minimizing the need for frequent human intervention.
Each module must operate with minimal energy consumption to prolong battery life or optimize solar panel efficiency if used. Modules should be able to operate under energy constraints while maintaining data collection and communication capabilities. Also, the system architecture should support the addition of new modules without compromising the network's performance or reliability. This includes managing

increased data traffic and maintaining consistent communication with all modules.Given the outdoor environment, modules must be resilient to various weather conditions and potential mechanical issues. The system must be able to identify and respond to module failures, either through self-diagnosis or via alerts in the Digital Twin. It should be possible to update the system's resource distribution strategy and the operational logic to answer changes in weather forecasts and other external factors (e.g., climate change, extra ordinary phenomenon). This requires a secure mechanism to update each module's operational parameters remotely, enabling remote adjustments and minimizing the need for physical intervention.

1.5 Expected Outcome

The final product will be a functional prototype of a representative part of the smart vineyard management system that demonstrates key points of your system, possibly ranging from on the field modules to the Digital Twin management.
It is mandatory to validate the operational environment and the solution envisioned with one of the school representatives based on actual architecture descriptions
Technical choices in terms of languages, libraries, frameworks or technologies are not imposed and you are free to choose the one(s) that seem(s) the most suitable to your team.
As available hardware, you'll have access to:

- Raspberry PI 3 and 4
- Arduino boards with a shield with classical sensors/actuators (leds, buttons, temperature sensors, bluetooth, RFID, NFC, infra red, ...)
- raspberry hat for arduino sensor usage
- a pi camera module.
- a USB ampere-meter (to be shared among teams)
- some radio frequency emitter/receivers
- the laptops of your team members to be used as you feel is appropriate

Regrettably, access to a real vineyard for testing purposes is unavailable. As a result, the demonstration the smart farming system will be executed through alternative methodologies, such as mocking or simulating. To replicate any additional required hardware, you may employ basic sensors and actuators from Arduino as part of the mock setup.

Important note

Quality of your code is of course important in general but will not be taken into account for this project
Usability in term of graphical design and or physical ergonomics is of course important but will not be taken into account for this project
Told differently, the following extra functional properties will not be considered in this project: maintainability of your code, correct versioning of your code, genericity of your code, UI design and ergonomic aspects[1].

[1] I really like qualitative code but my feeling is that this is too much demanding for a 8 weeks project.