Assignment 02 - "Smart Waste Disposal System"

"Embedded Systems e Internet of Things" final report

Lorenzo Cinelli

April 11, 2025

Contents

1 Analysis **2**
1.1 Description 2
1.2 Domain Model 3
1.3 Requirements 3
1.3.1 Mandatory: 3
1.3.2 Optional: 4

2 Design **5**
2.1 Architecture 5
2.2 Detailed Design 5
2.2.1 Container management task: 6
2.2.2 Filling monitoring task: 7
2.2.3 Temperature monitoring task: 7
2.2.4 Operator communication task: 8

3 Develop **9**

Chapter 1

Analysis

1.1 Description

The system required is a smart waste disposal system for - potentially dangerous - liquids. It has to be composed by a container having some sensors to check the temperature of the waste and the filling level. It has also two leds and a screen to signal if the user can pour the waste or not in case of some problems. To pour the waste there is a door controlled by two buttons - one to open it and one to close it. There is also an operator dashboard where operators can handle problems - like emptying the container - or consult information about the container. If no user is pouring waste the system waits a timeout and then goes in *sleep mode*.

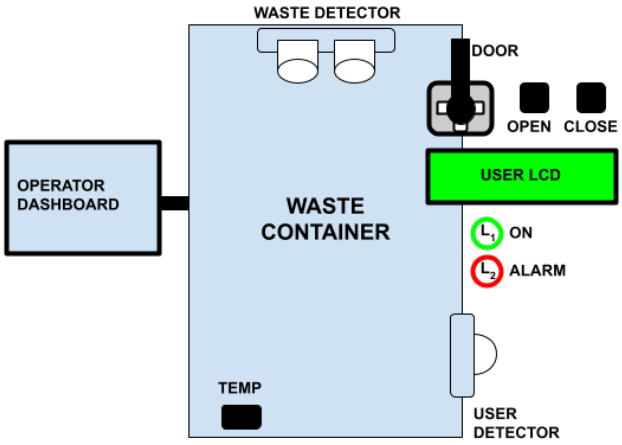

Figure 1.1: Smart Waste Disposal System

Here is the complete description for the assignment.

1.2 Domain Model

The waste container will manage an user detector. When an user is detected if the system was asleep it would be awaken. If for some time no one is detected the system will turn in *sleep mode* in order to consume less energy. Moreover it will manage the door, which can be opened by the user by pressing the "Open" button and can be closed by the user by pressing the "Close" button, otherwise it will close after a timeout. The door will automatically close if sensors detect that the container is full or the temperature of the liquid is too high. The door can be open and close also when the operator interact with the empty or restore button in his dashboard. Furthermore it will manage the information devices: two leds and a display. The green one denote that there are no problems in the container and a user can pour his waste, while the red one indicates either the container is full or the liquid inside is too hot. The display will show a message to the user depending on the state of the container.

The operator dashboard lets the operator see the current data of the container's sensors, which are the filling percentage and the temperature of the liquid inside. It offers also a button to empty the container, and a button to manage temperature problem inside the container. Data are stored in an history graph.

The waste container and the operator dashboard have to communicate. More specifically the dashboard will read from the waste container sensors data and will send a signal to empty or restore the container.

1.3 Requirements

1.3.1 Mandatory:

- The system has to turn in *sleep mode* if no user is detected for T_{sleep} seconds.
- The system has to wake up whenever an user is detected
- When the user presses the "Open" button the container's door opens to let him pour waste.
- When the user presses the "Close" button or T_1 seconds are elapsed from the opening of the container, the door will close.

- The filling sensor is constantly checking for the filling level of the container, sending it to the operator dashboard.
- Whenever the filling sensor detects that the container is full, it doesn't accept anymore waste until an operator empties it. If the door is open when the temperature exceeds its threshold it will immediately close.
- The temperature sensor constantly checks the waste temperature sending it to the operator dashboard.
- Whenever the temperature sensor detects a too high temperature in the liquid, the container doesn't accept anymore waste until the intervention of an operator. If the door is open when the temperature exceeds his threshold it will immediately close.
- The display has to show different messages depending on the container's state.
- When entering the pour is available it will be on the green led, while when is not available - in case of full container or waste with too high temperature - it will be on the red led.
- The operator dashboard displays the current state of the container, in particular the filling percentage and the waste temperature.
- An operator can empty the container through his dashboard pressing the button "Empty the container". When the container has to empty the door will open for T_3 seconds. After emptying the container will be available for users.
- In case of high temperature in the waste the operator can restore the container by pressing the "Restore" button in the operator dashboard, making the container available for users.

1.3.2 Optional:

- The operator dashboard can store data to make an history graph.

Chapter 2

Design

2.1 Architecture

The system is composed by two modules: the container module and the operator module. The two modules can exchange information between each other by a serial line.

The container module is a task-based architecture. Tasks are concurrent and can interact with one another by setting and reading shared flags. Tasks are executed by a cooperative scheduler. More specifically there are four tasks to manage the following behaviors: container management, filling monitoring, temperature monitoring and communication with the operator dashboard. Having multiple tasks offer a greater modularity, reusability and the possibility to have concurrent tasks.

The operator module is an active GUI showing the container's information and saving them to produce an history graph. It offers two buttons to manage issues of the container, namely the container full or the high temperature of the waste.

2.2 Detailed Design

The container module is composed by four different and concurrent tasks which are managed by a cooperative scheduler.

2.2.1 Container management task:

The container management task provides to user-related actions. This task is described by a `synchronous finite state machine`.

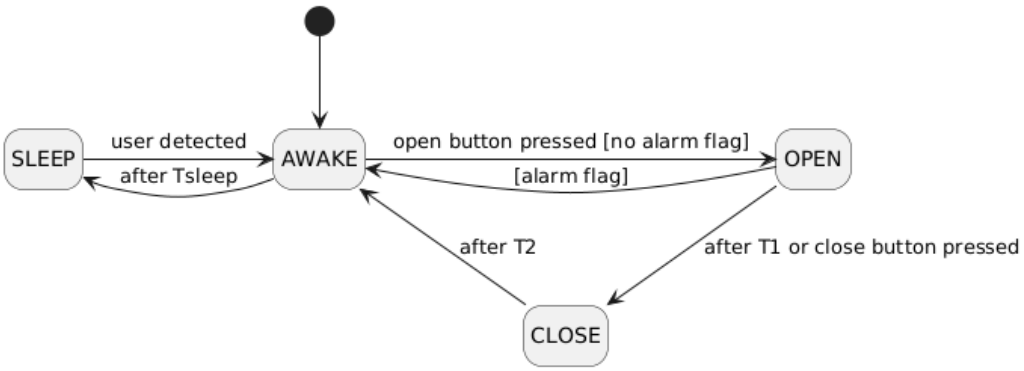

Figure 2.1: Container management task

- SLEEP: in this state the container is turned into `sleep mode` in order to reduce power consumption. It can be awaken by detecting an user.
- AWAKE: it's the initial state. In case of no actions performed by the user for T_{sleep} seconds it will go in *sleep* state.
  If there is no alarm set the container the green led is on, the red led is off and the display is showing "press open to enter the waste". In this case the container can go in *open* state after an user pressing the open button.
- OPEN: in this state the container's door is open, the display showing "press close when done" and the user can pour his waste inside the container. If T_1 seconds are elapsed or the user has pressed the close button, the state will turn in `close`, closing the door. In case an alarm is detected by another task, and an alarm flag is set, the container will immediately turn back in `awake` state closing the door, turning on the red led and off the green one, and showing on the display an error message depending on the alarm.
- CLOSE: in this state the door is closed and "waste received" is shown on the display for T_2 seconds, then the container will go in `awake` state.

2.2.2 Filling monitoring task:

The filling monitoring task provides to periodically check the filling level of the container. This task is described by a `synchronous finite state machine`.

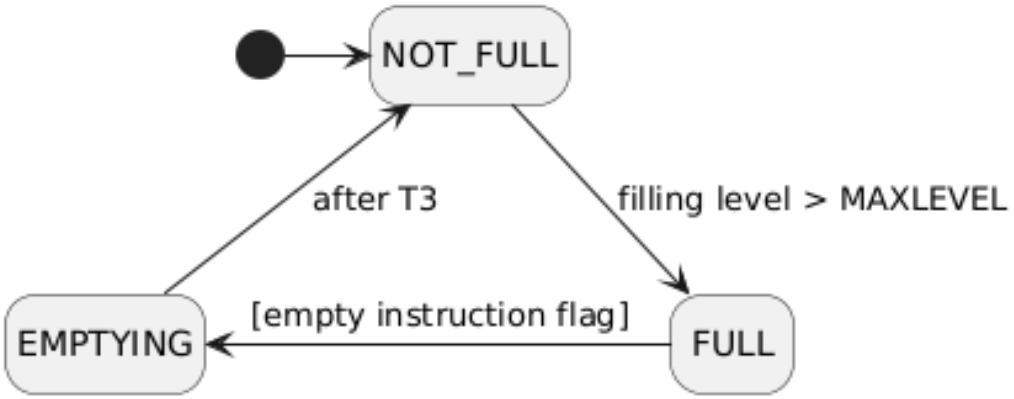

Figure 2.2: Filling monitoring task

- AVAILABLE: it's the initial state. It monitors the filling of the container and if it surpasses the maximum capacity of the container it turns in `full` state setting a `full alarm flag`, turning on the red led, off the green one and displaying on the scree "container full".
- FULL: in this state is checked if the `empty instruction flag` is set. In this case the state is set to `emptying`, the door is opened and the instruction flag is removed.
- EMPTYING: this state remains for T_3 seconds in order to let the container empty and then the state is set to `available` and the full alarm flag removed. If there is no other alarm flag set it will also turn on the green led, off the red one and set the default "press open to enter the waste" message on the display.

2.2.3 Temperature monitoring task:

The temperature monitoring task provides to periodically check the temperature of the waste. This task is described by a `synchronous finite state machine`.

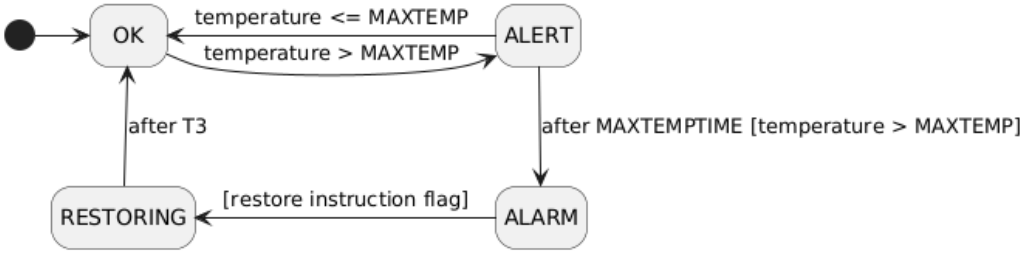

Figure 2.3: Temperature monitoring task

- OK: it's the initial state. It monitors the waste temperature and if it surpasses $MAXTEMP$ is set `alert` state.
- ALERT: in this state the task checks if the temperature keeps being higher than $MAXTEMP$ for $MAXTEMPTIME$. In affirmative case the `temperature alarm flag` is set, is set the `alarm` state, the red led is turned on, the green led turned off and the message on the display is "problem detected". In negative case the `ok` status is put back.
- ALARM: in this state the task checks if the `restore instruction flag` is set and if so it puts the `restoring` state by removing the restore flag and opening the door.
- RESTORING: in this state the door stays open for T_3 seconds to let the container be emptied - in this way the temperature will no longer be a problem - and then the task's state is set to `ok`, removing the temperature alarm flag. If there is another alarm flag the display's message is changed depending on the alarm - in this case only the full alarm can be present - otherwise the green led is turned on, the red one turned off and set the default message on the display.

2.2.4 Operator communication task:

This task is simpler compared to previous ones because is not described by a finite state machine.
It waits on the serial line waiting for a request. The request can be:

- GET DATA: in this case it sends back on the serial line the current filling level of the container and the current temperature of the waste.
- EMPTY: in this case it sets the `empty instruction flag`.
- RESTORE: in this case it sets the `restore instruction flag`.

Chapter 3

Develop

The system is developed in two different languages:

The container module is developed in `c++` by using `PlatformIO` IDE with `arduino` framework. It uses two external libraries: "LiquidCrystal_I2C.h" to manage the display and "Servo" to manage the servo motor.

The operator module is developed in `python` with two threads: one for the gui and the serial communication, the other one to repaint periodically the new history graph. It uses:

- "serial" library to communicate through the serial line with the other module.
- "threading" library to run multi thread.
- "tkinter" library to manage the graphical part.
- "time" library to be able to manage time.
- "matplotlib.pyplot" to draw graphs.
- "PIL" library to open images and sending to the gui.

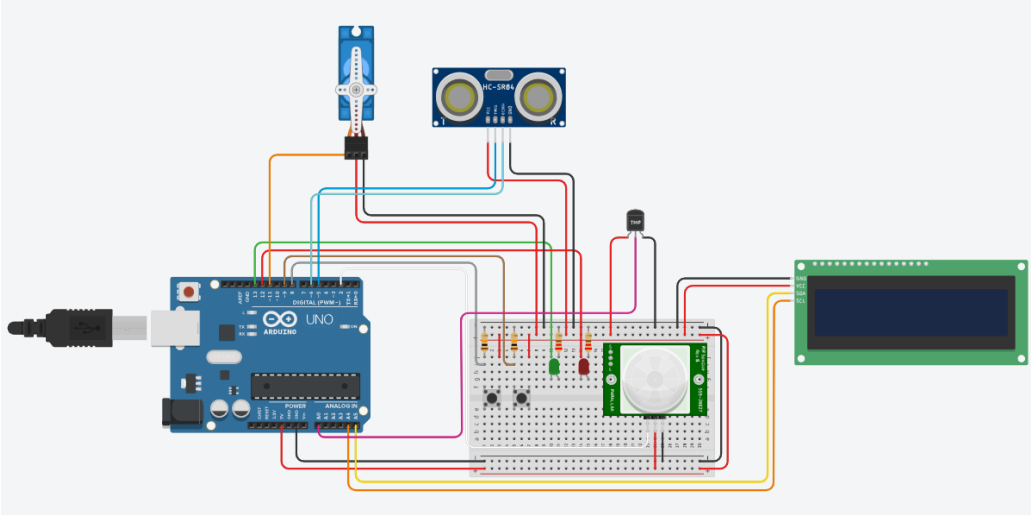

Figure 3.1: Circuit schema

Here the link to the project's GitHub repo.
Here is the link to the demonstration video.