Assignment 2 - Smart Waste Disposal System

A.A 2024/2025

Elisa Yan

December 3, 2024

Contents

Chapter 1

Introduction

For the second assignment of Embedded Systems and Internet of Things, we need to develop a Smart Waste Disposal System, a system designed to manage liquid waste in a safe and efficient way. The system includes sensors to monitor waste levels and temperature, a motorised waste disposal door, and a GUI to facilitate user and operator interaction.

Chapter 2

State diagram

As a first step in analyzing the requirements, I created a state diagram to model the system's behavior, as shown in the Figure 2.1. Initially, the system begins in the **Idle** state, preparing for operation. Once the system is ready, it waits for the user to press the *Open Door* button. If no action is taken within a predefined time (time1), the system enters the **Sleep** state. It remains in this state until a user is detected, at which point it returns to Idle.

From Idle, the system moves to the **Receiving Waste** state when the user opens the door. During this state, the system monitors whether the waste level exceeds a predefined threshold, indicating that the container is full. If the user has pressed the *Close Door* button or after a specified time (time2), the system moves to the **Waste Received** state before returning to Idle.

If the container becomes full, the system moves to the **Container Full** state. In this state, it waits for the user to press the *Empty Container* button on the GUI application. Once the container has been emptied, the system returns to Idle to resume operation.

Additionally, the system includes a parallel state for handling temperature control. If the container's temperature exceeds a predefined limit for a certain duration, the system transitions to a **Temperature Alert** state. In this state, the user is required to resolve the issue by pressing the *Restore* button on the GUI dashboard.

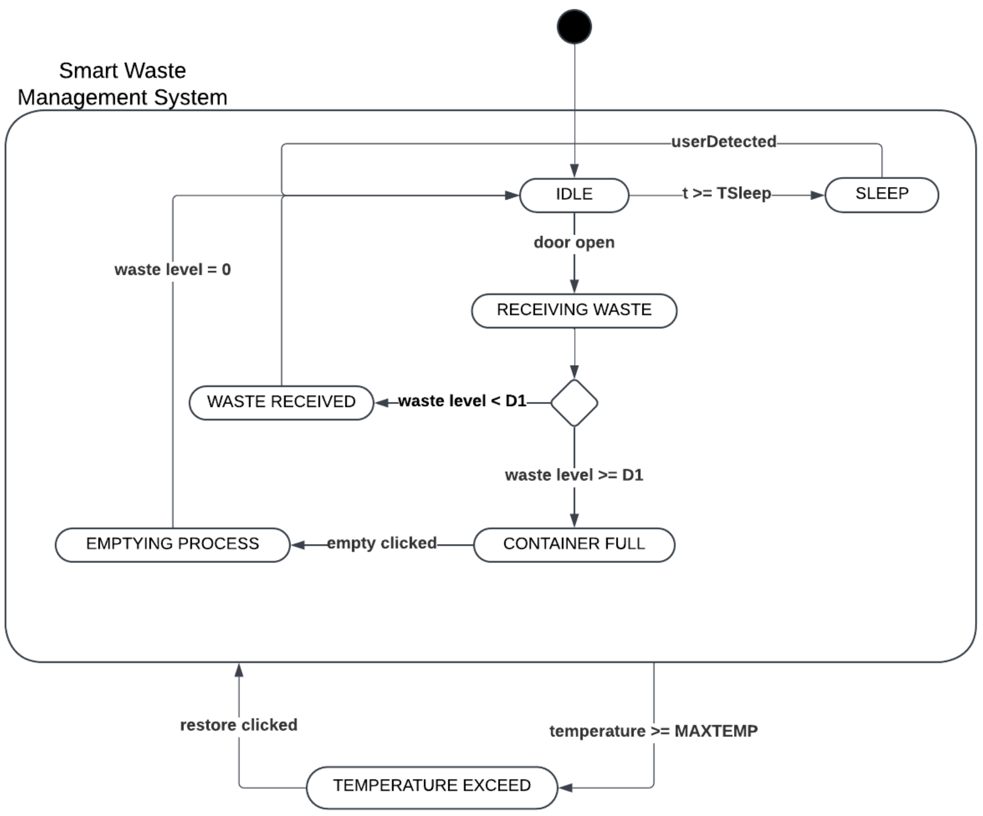


Figure 2.1: State diagram

The core of the system is the **SmartWastePlant** class. This class acts as a central controller for the waste management system, integrating hardware components and managing state transitions.

Each task within the system is built around a **Task** class managed by a **Scheduler** to ensure sequential execution. Each task contains a Finite State Machine (FSM) architecture to handle its internal actions, providing a clear and structured approach to defining and managing state specific behaviours.

2.1 Ready Task

When entering the ReadyTask, the system transitions to the **Init** state, where it prepares all necessary devices. After the preparation, the system waits for the user to press the Open button.

If the button is not pressed within the specified *TIME-SLEEP*, the system enters **Sleep mode** and can wake up only if the PIR sensor detects a user. However, if the button is pressed within the allowed time, the system transitions to the next task to start receiving waste.

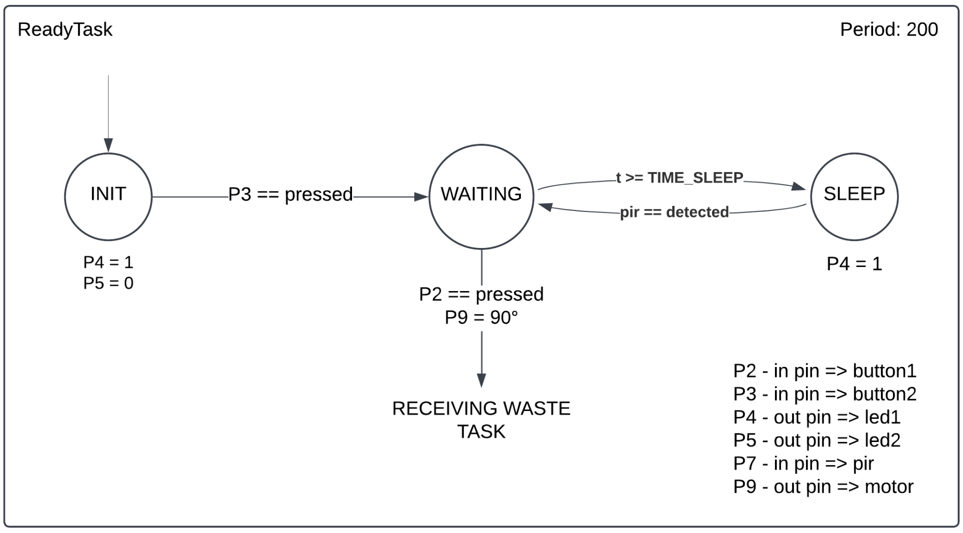


Figure 2.2: Ready Task

2.2 Receiving Waste Task

Once the user has pressed the Open button, the system enters the **Init** state to complete the setup process before moving to the **Spilling** state. In this state, the motor is activated to move the door to a 90-degree position.

The system remains in the **Spilling** state until one of the following conditions is met:

- the user presses the Close button,
- the spilling time exceeds the predefined *TIME1*, or
- the sonar sensor detects that the waste distance has exceeded the maximum limit.

If the container is full, the system transitions to the **Ready for Empty** state, indicating that it needs to be emptied. Otherwise, the system returns to the **Idle** state, ready to receive new waste.

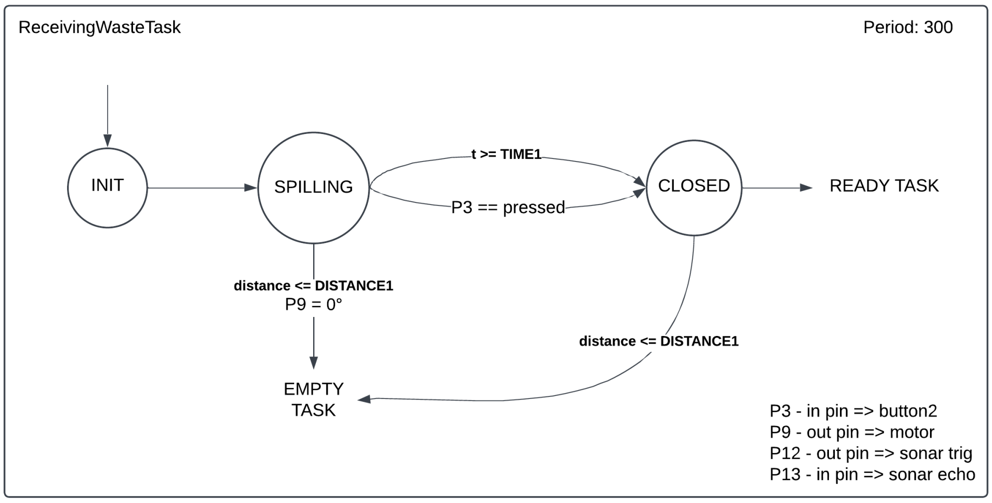


Figure 2.3: Receiving Waste Task

2.3 Empty Task

Once the sonar detects that the container is full, the system transitions to the **Empty Task**, starting in the **Init** state. At this point, the system waits for the user to interact with the dashboard.

When the user presses the Empty button, the system moves to the next state to monitor the emptying process through the GUI.

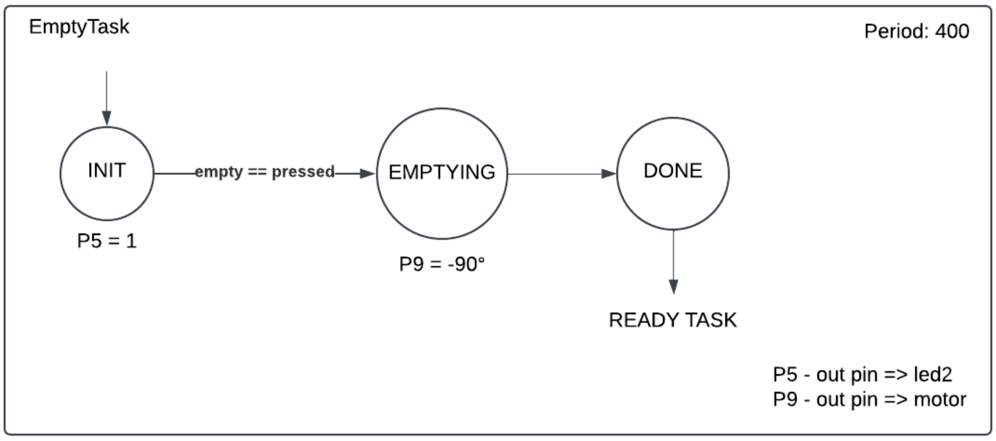


Figure 2.4: Empty Task

2.4 Read Temperature Task

This task runs in parallel with other tasks, as the temperature can exceed the threshold at any time. In the **Measure** state, the system continuously monitors whether the temperature exceeds a predefined limit for a specified duration.

If this condition is met, the system transitions to the **Alarm** state, where it waits for the operator's intervention via the GUI dashboard. Once the operator presses the Restore button, the system returns to the **Idle** state.

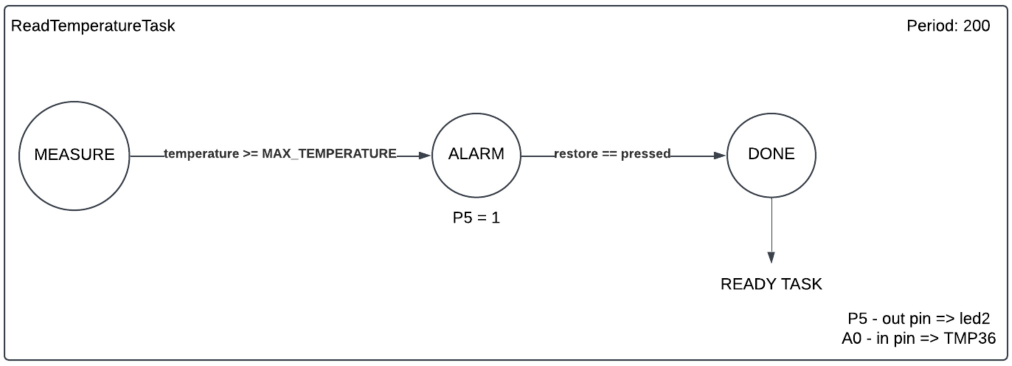


Figure 2.5: Read Temperature Task

Chapter 3

Breadboard Scheme

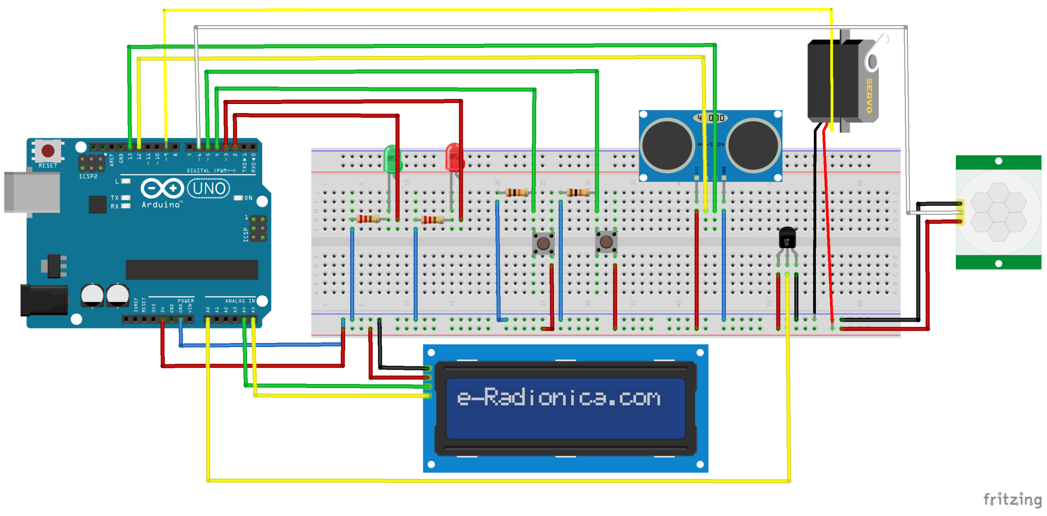


Figure 3.1: Breadboard Scheme

Chapter 4

Application

Once the Arduino system has been rebooted, as shown in Figure 4.1, the GUI displays the current filling status of the container. If the user presses the **Open** button, the container is ready to receive waste.

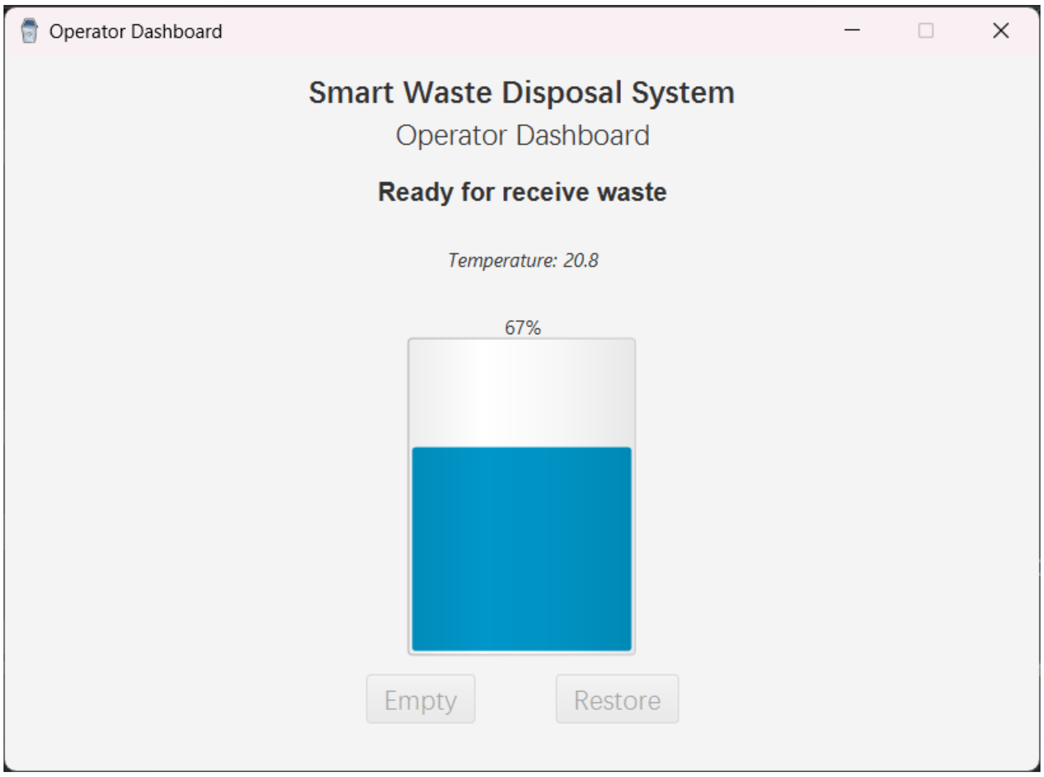


Figure 4.1: Scene of receiving waste

If the container is full, the **Empty** button will be activated, allowing the operator to perform maintenance. Once the container is emptied, the **Empty** button will return to its initial state.

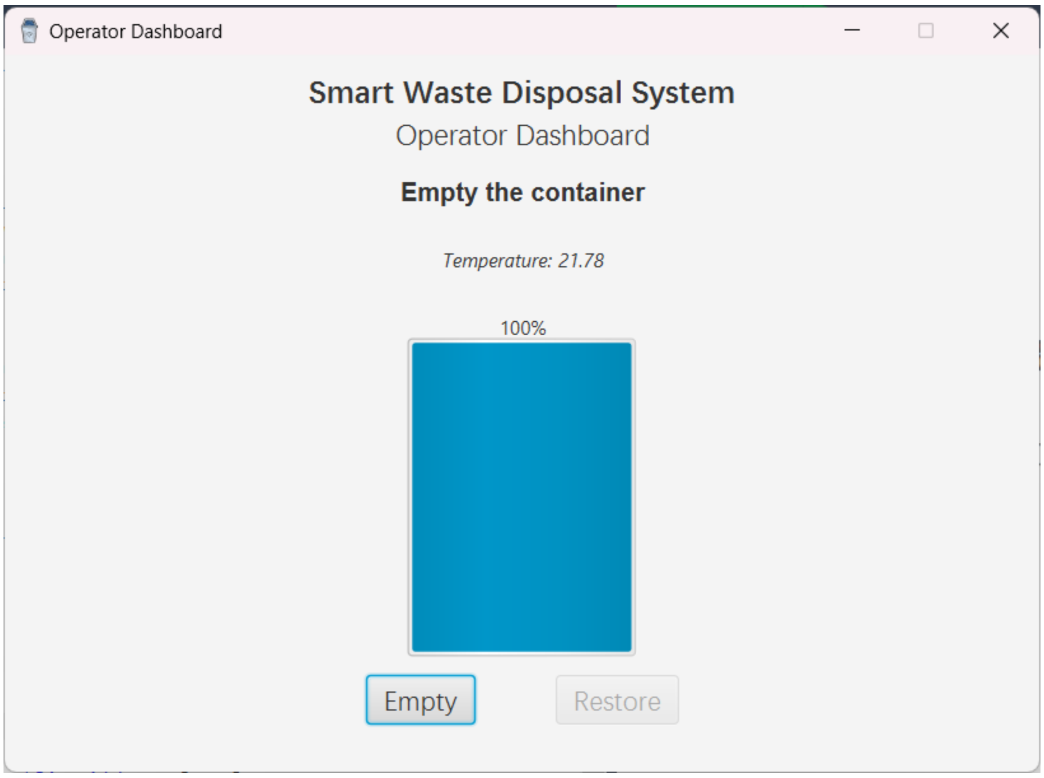
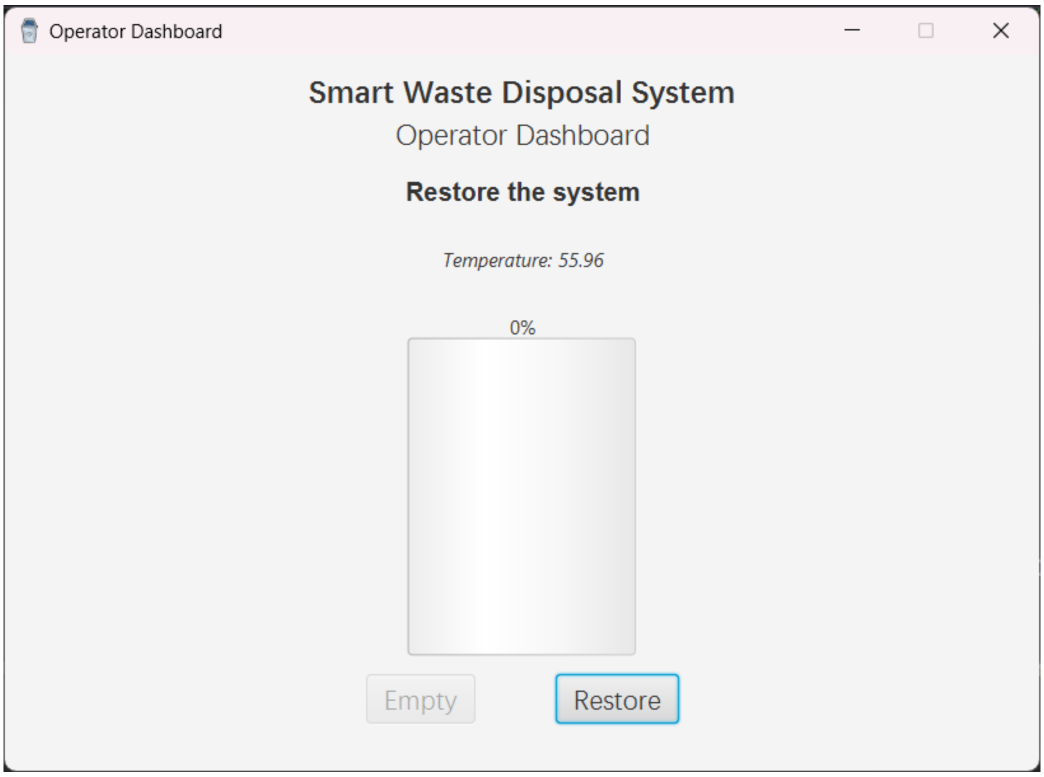


Figure 4.2: Scene of emptying process

If a problem is detected, such as the temperature exceeding the threshold, the **Restore** button on the GUI will be activated for the operator to address the issue.


Operator Dashboard

Smart Waste Disposal System

Operator Dashboard

Restore the system

Temperature: 55.96

0%

Figure 4.3: Scene of restoring