Report

Scheduler

During the implementation of the GateTask we have encountered some error with the servo: sometime it does not move when the write() is called. So we chose to use Timer1 for it and because of this for the scheduler we used MSTimer2.

Our implementation

Tasks

The task we have implemented are the following ones:

- AllarmTask;
- WasteLevelTask;
- LedTask;
- ProxTask;
- GateTask;
- SerialCommTask.

ContainerProp

For the communication between the Task we used an object that contains all the information of the system, in particular it knows if there is an alarm or the container is full.

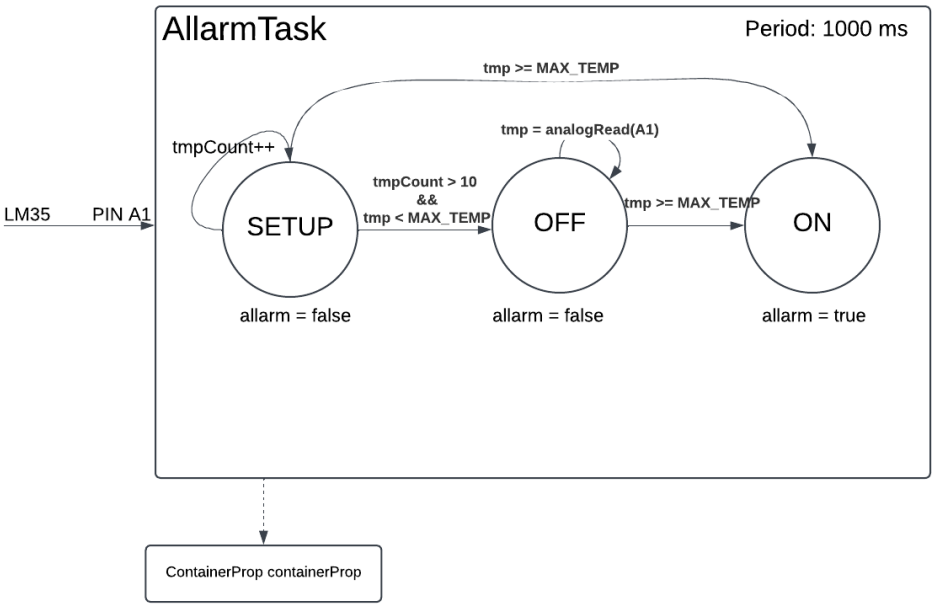
LCDManager

It is a wrapper for the LCD that is passed in all the tasks that can change the LCD. This manager make sure that there isn’t flickering by storing inside the current message displayed so that the LCD print only when the message is different than the current one.

AllarmTask

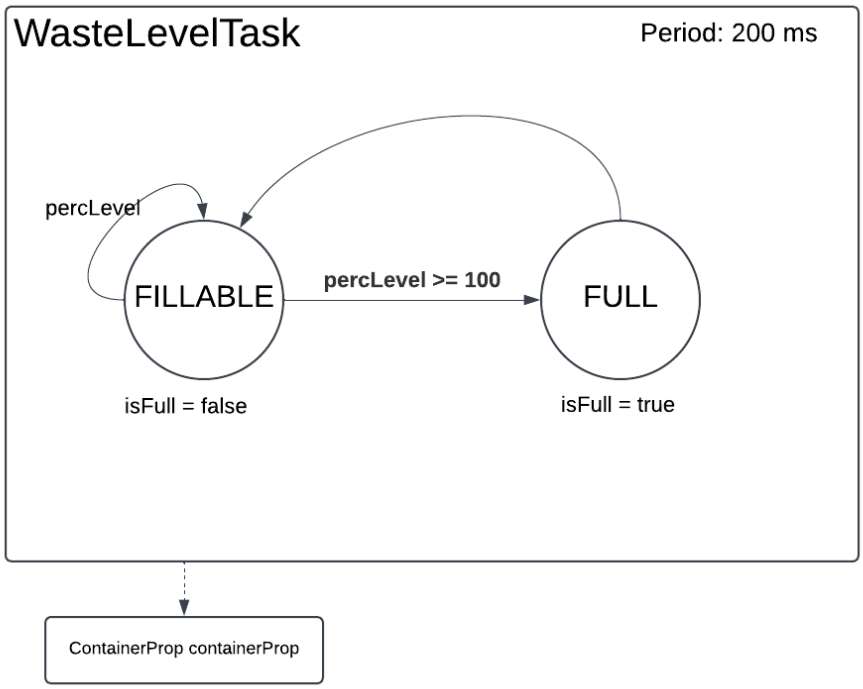
It has the duty of measure the temperature of the system and gives the alarm if she exceeds the threshold.

We chose to make an average of 10 measurements because sometimes the LM35 give some unreal values. So at the start of Arduino the first 10 tick() of this task are for setup.

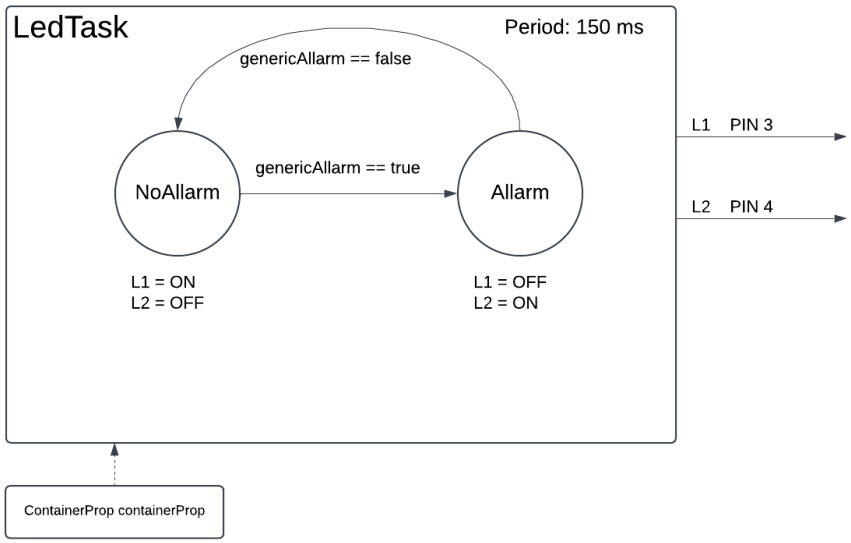


WasteLevelTask

It simply measures the level of the waste and in case notifies, trough the containerProp, the GateTask to close.



LedTask



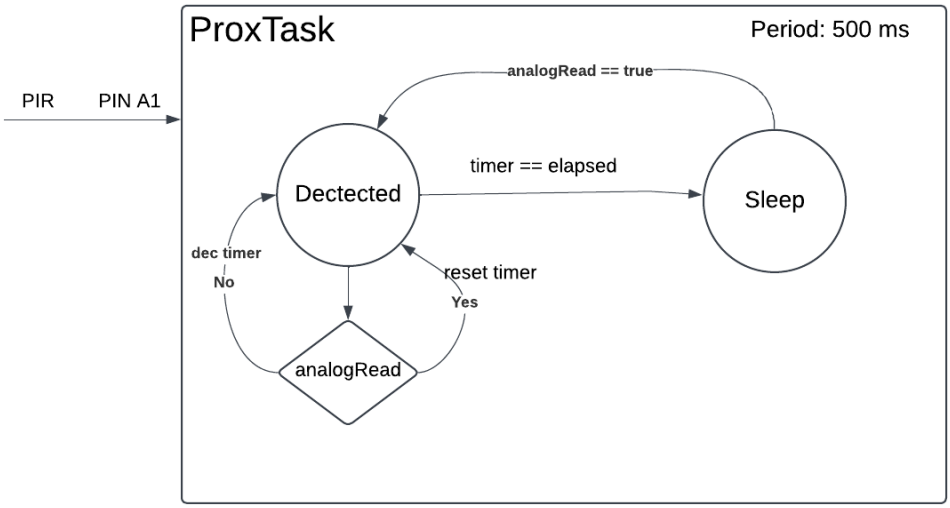
ProxTask

Every tick the sensor reads if there is someone in front of him:

- if it detects someone it resets the timer (a TickCounter class);

- if he doesn’t detect someone it decrements the timer.

Once the timer expires Arduino goes to sleep.



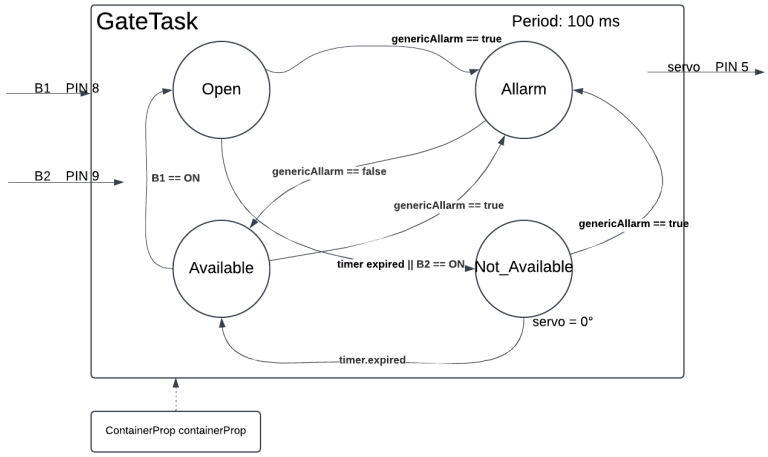
GateTask

This task uses the containerProp for check if there are alarm. Inside it there are 2 class implemented:

- Gate;
- TickCounter.

The first one is used for moving the servo with methods like openGate() etc.. With this implementation we can separate the object Gate with the task in order to operate in a safer and more efficient way.

TickCounter is simply a class that simulate a timer: we set a time and every tick the timer decrements, so we can calculate the seconds using the scheduler frequency.



SerialCommTask

This task simply send every tick the value at the GUI, after that it waits a message from the GUI.

If there is a problem and someone presses the button the GUI sends back a message like “Restore” or “Empty”, but if the user doesn’t press the button the GUI sends a specified message (in our case “X”). We did this because once there is an alarm the Arduino doesn’t get stuck in the receiving mode and is able to send the values to the GUI.