Smart Bridge

Iorio Matteo, Vincenzi Fabio, Rapolla Luca

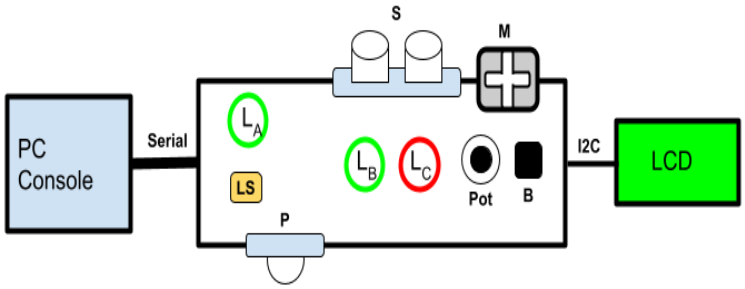
November 2022

1 Introduction

1.1 Description

The prototype is meant to simulate a system mounted on a bridge (over a river), providing smart functionalities. A first functionality is about monitoring the water level of the river and, in the case of dangerous situations (water level too high), opening some valves to let the water flow on some lands. A second functionality is about smart lighting, automatically turning on/off a light on the bridge depending on the presence of people traversing the bridge.

The breadboard of the prototype includes two green *leds* LA and LB, a red led LC, a *tactile button* B, a *potentiometer* Pot, a *servo-motor* M, a sonar S, a *pir* P, a *light sensor* LS, an *LCD*. The embedded system is connected to the PC (Console) through the serial line.



1.2 Systems

1.2.1 Smart Light System - SLS

The *pir* P, *light sensor* LS, *led* LA are part of the subsystem used to realise the smart lighting behaviour: if someone is detected on the bridge by P, then the light LA should be either turned on or not depending on the light level as measured by LS. If the level is less than some threshold **THL**, then the light LA is turned on, otherwise the light is not turned on. The light is turned off either after some time T1 in which no one was detected on the bridge by P, or the level of luminosity becomes higher than the threshold **THL**.

1.2.2 WaterFlow Control System - WCS

The *sonar* S, motor M, *potentiometer* Pot, the *leds* LB and LC , the *button* B and the *LCD* are part of the subsystem used to monitor the water level and eventually take actions in case. In particular: the sonar S is used to continuously measure the river level (by measuring the distance from the water surface). The motor M is meant to control the opening/closing of a valve to allow the river water to flow – 0° degrees corresponds to valve closed and 180° corresponds to fully open valve.

When the river water level is below a water level **WL1** the system is in a normal situation: the green led LB is on and LC is off – it means that the bridge can be used. In this situation, the sampling of the water level measure should be done every period PEnormal When the water level is higher than WL1 and below a level **WL2**, the system is in a pre-alarm situation: The red led LC starts blinking with a period of 2 seconds. Sampling must be done with a period PEprealarm (¡ PEnormal) The LCD is turned on, informing about the pre-alarm and displaying the current water level

When the water level is higher than **WL2** up to **WLMAX**, the system is in an alarm situation The **smart lighting subsystem** is turned off (the led LA must be off) The green led LB is turned off and the red *led* LC is on (without blinking) Sampling must be done with a period PEalarm (¡ PEprealarm) The *valve* must be opened of some ALPHA degrees (0 ¡ ALPHA ¡ 180), whose value linearly depends on the the current water level, **WL2** and **WLMAX** (so 0 degrees corresponds to **WL2** and 180 degrees correspond to **WLMAX**). The opening of the valve changes dynamically depending on the current water level The LCD is still on, informing about the alarm situation and displaying both the current water level and the opening degrees of the valve In the alarm situation, a human user (e.g. a bridge maintainer) can take the control of the valve by pressing the *button* B, then using the *potentiometer* Pot to control the opening (degrees, from 0 to 180) and then pressing again the *button* B to stop controlling manually.

1.2.3 Pc

The program running on the PC Console should: a graph reporting the temporal trend of the water level in the alarm state, the possibility to take control of the valve (like it happens with the button B and Pot)

2 Components

2.1 Hardware

For our project we used different components including :

- Servo Motor
- 3 Leds
- Potentiometer
- Button
- Monitor I2C LCD
- PhotoResistor
- Sonar
- Pir

For each component we did a Class stored in the package *Hardware*. Every single piece of hardware inherit from *SimpleComponent* and some of the Hardware inherit from *Component*.

2.2 Functionalities

After the creation of all hardware components, we created three classes for each System :

- Smart Light System : composed by 1 Led, 1 PhotoResistor and 1 Pir
- Motor Control : composed by 1 Servo Motor, 1 Button and 1 Potentiometer
- WaterFlow Control System : composed by 1 Sonar, 1 Monitor I2C LCD and 2 Leds

Each of this classes is stored inside the package *Functionalities*. We have also created a Class named Bridge which contains all three systems.

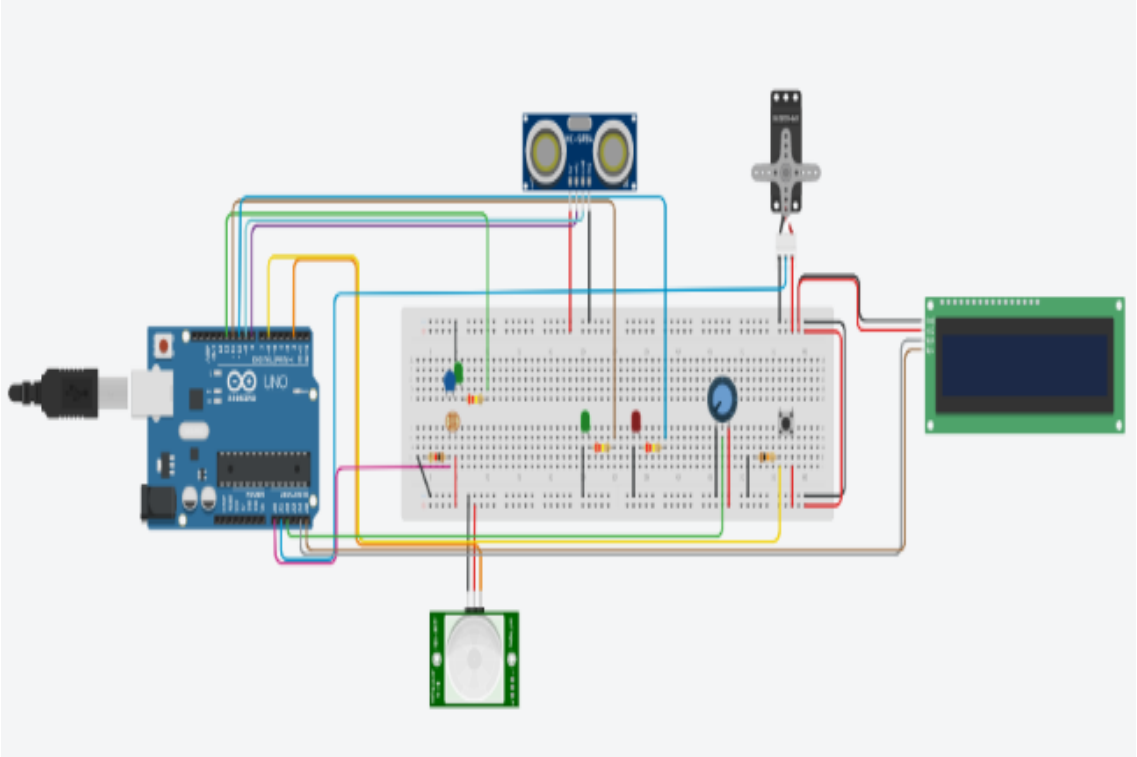
2.3 Logic

Every Functionality has its own state-machine. Every state machine is stored inside the package *Logic*. We find two state-machines:

- logic-smart-light-system : contains the Smart Light System's state-machine, and the method *TickSLS* contains the state-machine
- logic-water-control-system : contains the union of the Motor Control System and the WaterFlow Control Sytem state-machine, contained inside the method TickWCS.

The logic of the Water Flow System can interact with the Smart Light System's logic in order to warn about the Alarm State.

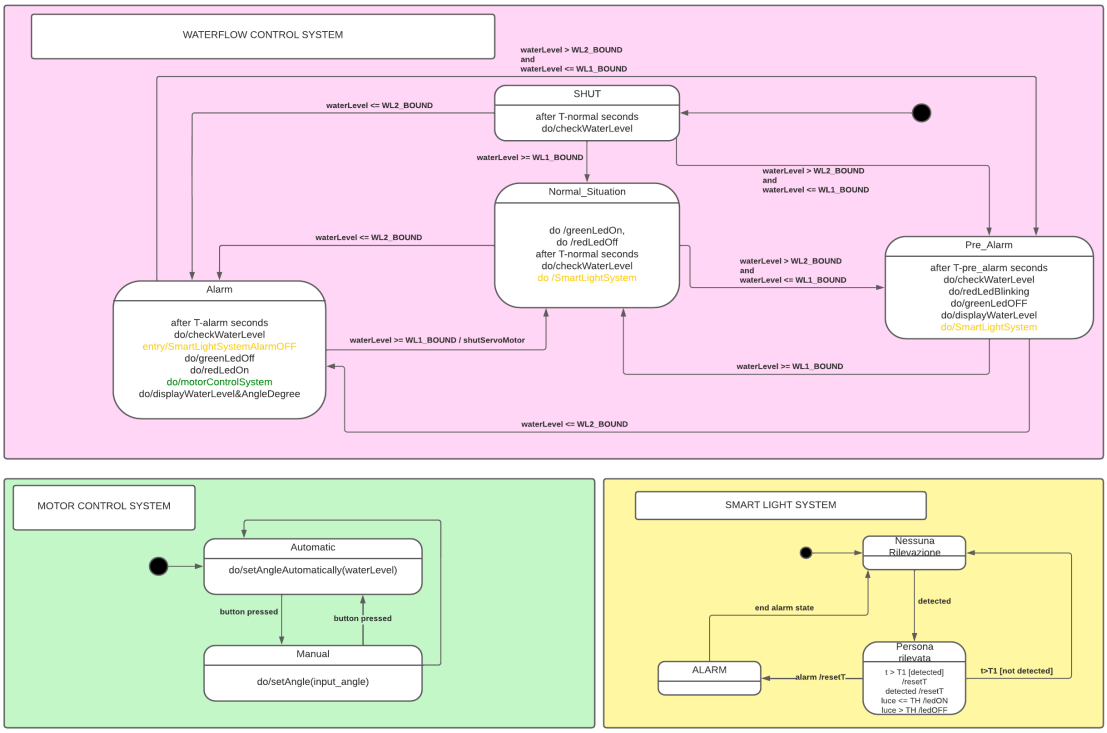
3 Board



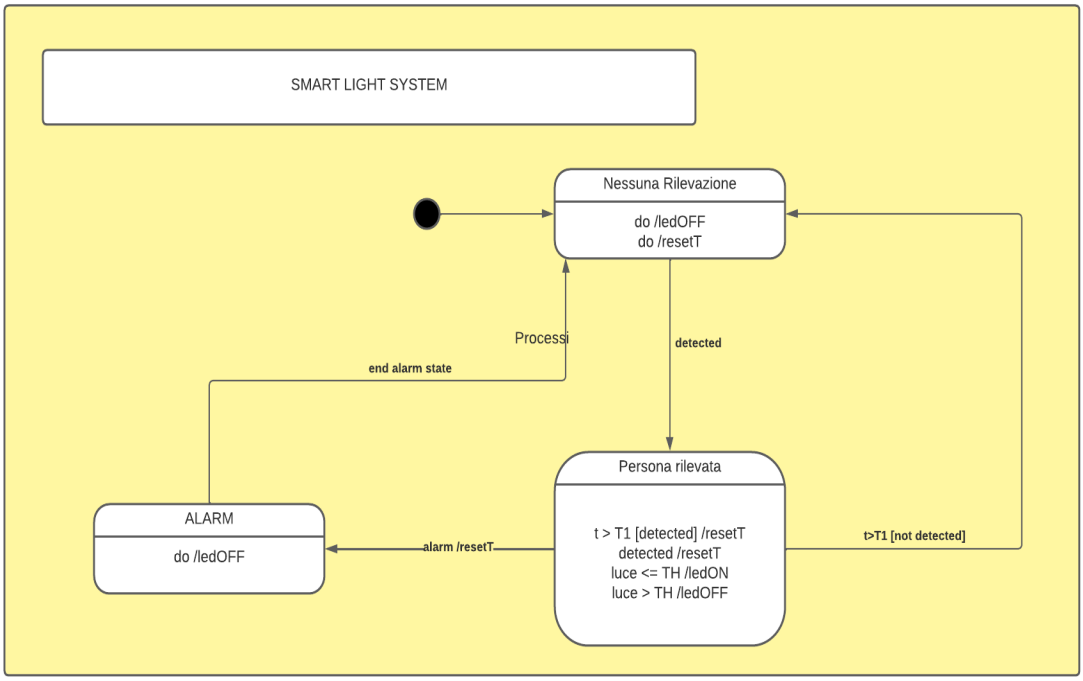
We decided to split the board in order to have a section for the Smart Light System and another section for the WaterFlow Control System.

4 State Diagrams

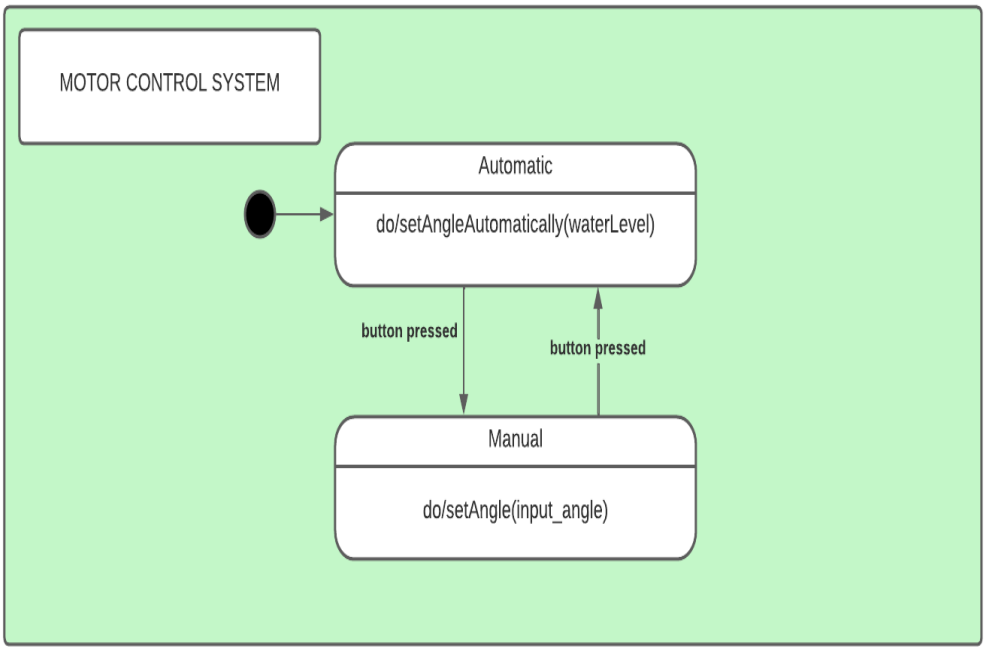
The main system was modelled in the fllowing way:



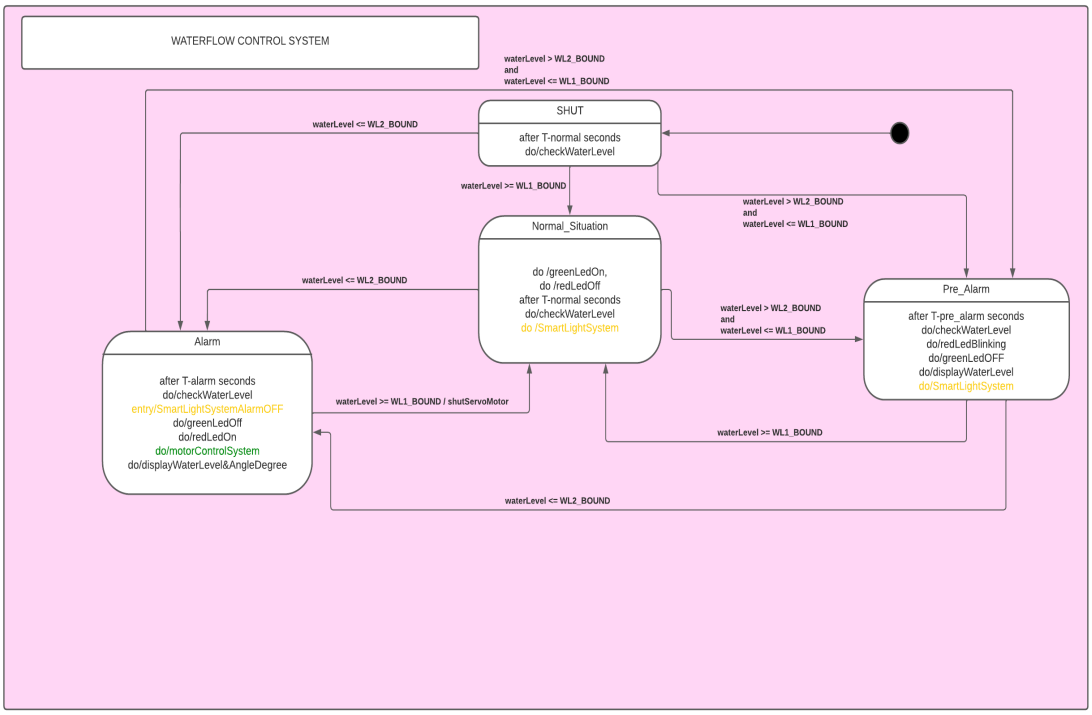
4.1 Smart Light System



4.2 Motor Control System

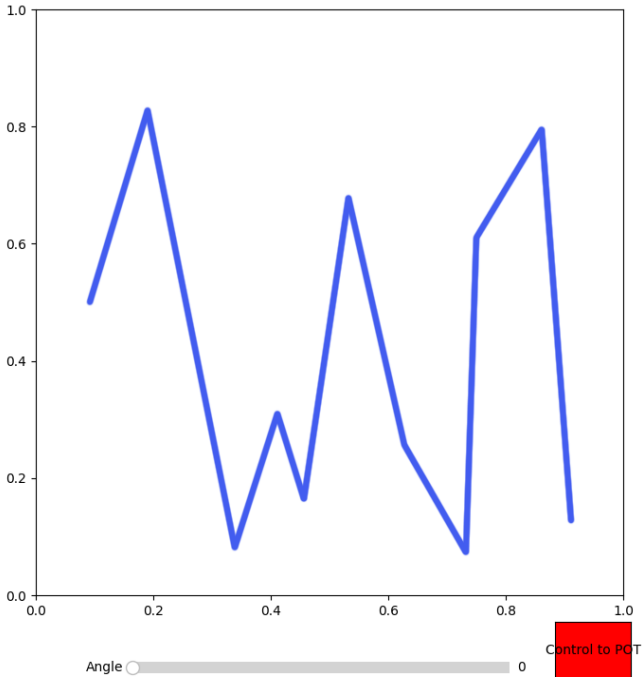


4.3 WaterFlow Control System



5 PC Application

5.1 Python Application



In order to take control with the pc of the Motor Control System, we have to press the button in order to switch from automatic to the manual, after that by using the button "Control to POT" we set as priority the PC. At this point we can set the Servo's angle by using the Slider. If we want to use the Potentiometer just click again the button "Control to POT" and If we want to switch to automatic we can simply press the button on the board.