Embedded System and IoT - a.y. 2022-2023

v1.0 - 20221001

Assignment #2 - *Smart Bridge*

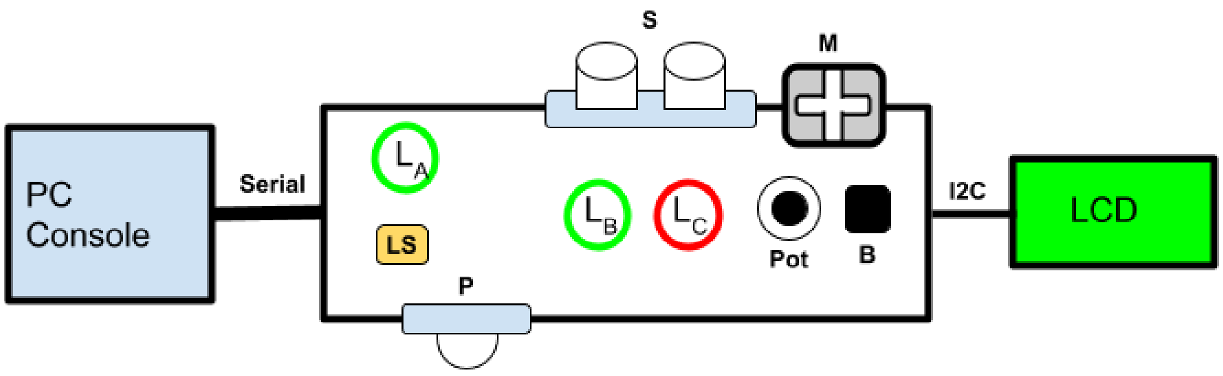
We want to realise an embedded system called *Smart Bridge*

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 L_A and L_B, a red led L_C, 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.



The pir P, light sensor LS, led L_A are part of the subsystem used to realise the smart lighting behaviour:

- if someone is detected on the bridge by P, then the light L_A should be either turned on or not depending on the light level as measured by LS.
- If the level is less than some threshold TH_L, then the light L_A is turned on, otherwise the light is not turned on.
- The light is turned off either after some time T_1 in which no one was detected on the bridge by P, or the level of luminosity becomes higher than the threshold TH_L.

The sonar S, motor M, potentiometer Pot, the leds L_B and L_C , 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 WL_1 the system is in a *normal* situation:

- the green led L_B is on and L_C 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 PE_{normal}

When the water level is higher than WL_1 and below a level WL_2, the system is in a *pre-alarm* situation:

- The red led L_C starts blinking with a period of 2 seconds.
- Sampling must be done with a period $PE_{prealarm}$ (< PE_{normal})
- The LCD is turned on, informing about the pre-alarm and displaying the current water level

When the water level is higher than WL_2 up to WL_{MAX}, the system is in an *alarm* situation

- The smart lighting subsystem is turned off (the led L_A must be off)
- The green led L_B is turned off and the red led L_C is on (without blinking)
- Sampling must be done with a period PE_{alarm} (< $PE_{prealarm}$)
- The valve must be opened of some ALPHA degrees (0 < ALPHA < 180), whose value linearly depends on the the current water level, WL_2 and WL_{MAX} (so 0 degrees corresponds to WL_2 and 180 degrees correspond to WL_{MAX}). 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.

The program running on the PC Console should:

- [*basic* version] just report the current state of the bridge, either using a GUI or a text-based output
  - if the smart light are either on or off
  - normal/pre-alarm/alarm situation
- [*complete* version] like simplest, plus:
  - a graph reporting the temporal trend of the water level
- [*full-fledged* version] like complete, plus:

    - in the alarm state, the possibility to take control of the valve (like it happens with the button B and Pot)

Develop the embedded software on Arduino + PC connected through the serial line, implementing the Arduino part in C++/Wiring e the PC part in Java or in another favourite language. The Arduino program must be designed and implemented using task-based architectures and synchronous Finite State Machines.

For any aspect not specified, you are free to choose the approach you consider more useful or valuable.

The Deliverable

The deliverable consists in a zipped folder **assignment-02.zip** including two subfolders:

- **src** folder, including
    - two further folders: arduino and java, the former must contain a smart_cm folder, containing the Arduino project, and the latter the sources of the Java program
- **doc** folder, including:
    - a brief report describing the solution, in particular the diagram of the FSMs and the representation of the schema/breadboard – using tools such as TinkerCad or Fritzing
    - a short video (or the link to a video on the cloud) demonstrating the system