ALMA MATER STUDIORUM -
UNIVERSITY OF BOLOGNA

EMBEDDED SYSTEMS AND INTERNET OF THINGS

ASSIGNMENT #2

Smart Bridge

Author
Fabio VEROLI

November 30, 2022

Contents

1 Introduction

The project represents the development of the second assignment for the Embedded Systems and IoT course at UNIBO. In the assignment, we want to realise an embedded system called Smart Bridge.
The prototype is meant to simulate a system mounted on a bridge (over a river), providing smart functionalities.

1. 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.

2. 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.

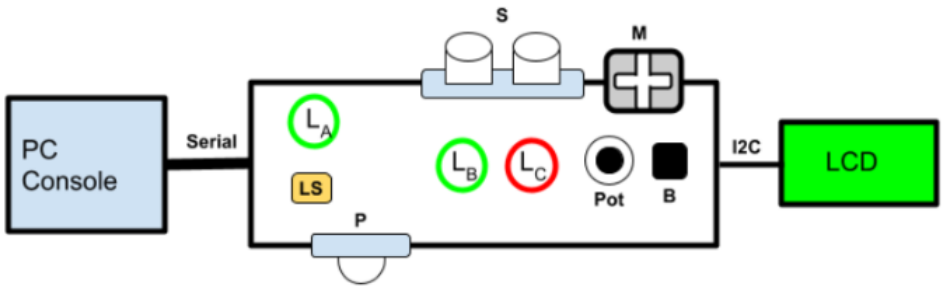

Figure 1: Scheme of components

2 Description

2.1 Smart Lighting Subsystem

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.

2.2 Water Level Subsystem

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.
- The sampling of the water level measure should be done every period PE_{normal}.

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 $PE_{pre\text{-}alarm}$ (< 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 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 PE_{alarm} (< $PE_{pre\text{-}alarm}$). T
- 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.

2.3 PC Console

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)

3 Development

3.1 Circuit Diagram

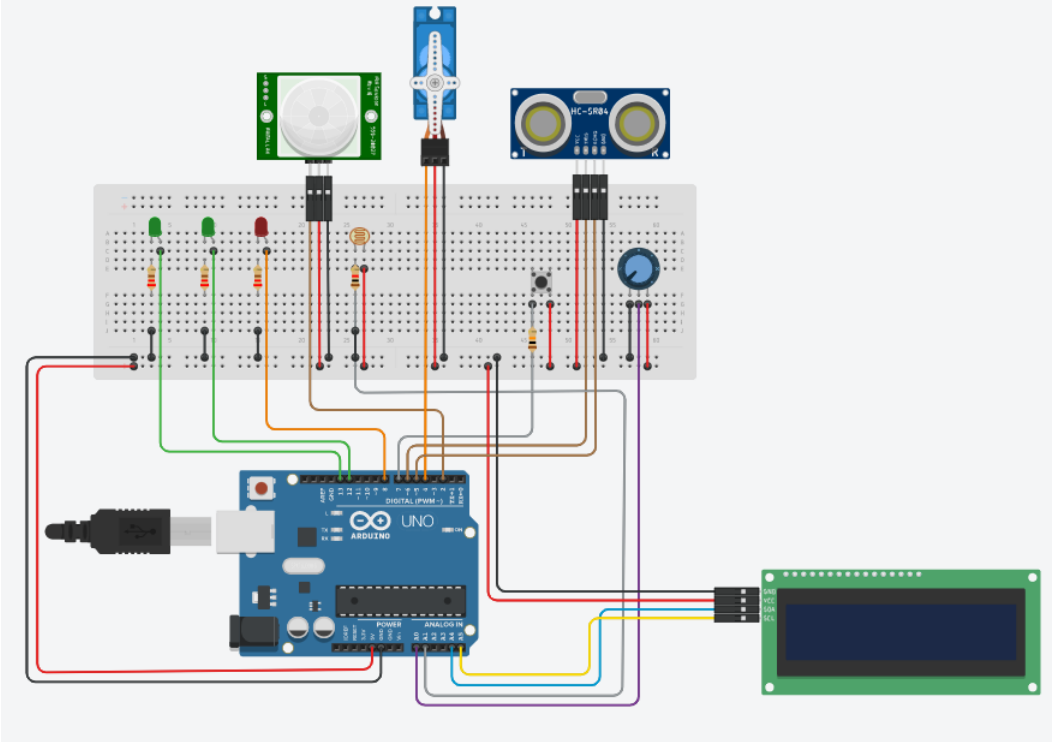

Figure 2: The actual circuital scheme used in the project

Note: Both the sensitivity potentiometer and the time-delay potentiometer, placed on the back of the PIR, were fully turned anticlockwise. In this way, we obtain the minimum detection range and we set to the minimum the time the output will remain HIGH after motion is detected.

Note: Software side we only consider value from the HC-SR04 that are lower than 4.0 meters and higher that 0.02 meters, according to the min and max range given in the sensor data-sheet.

3.2 FSMs Diagram

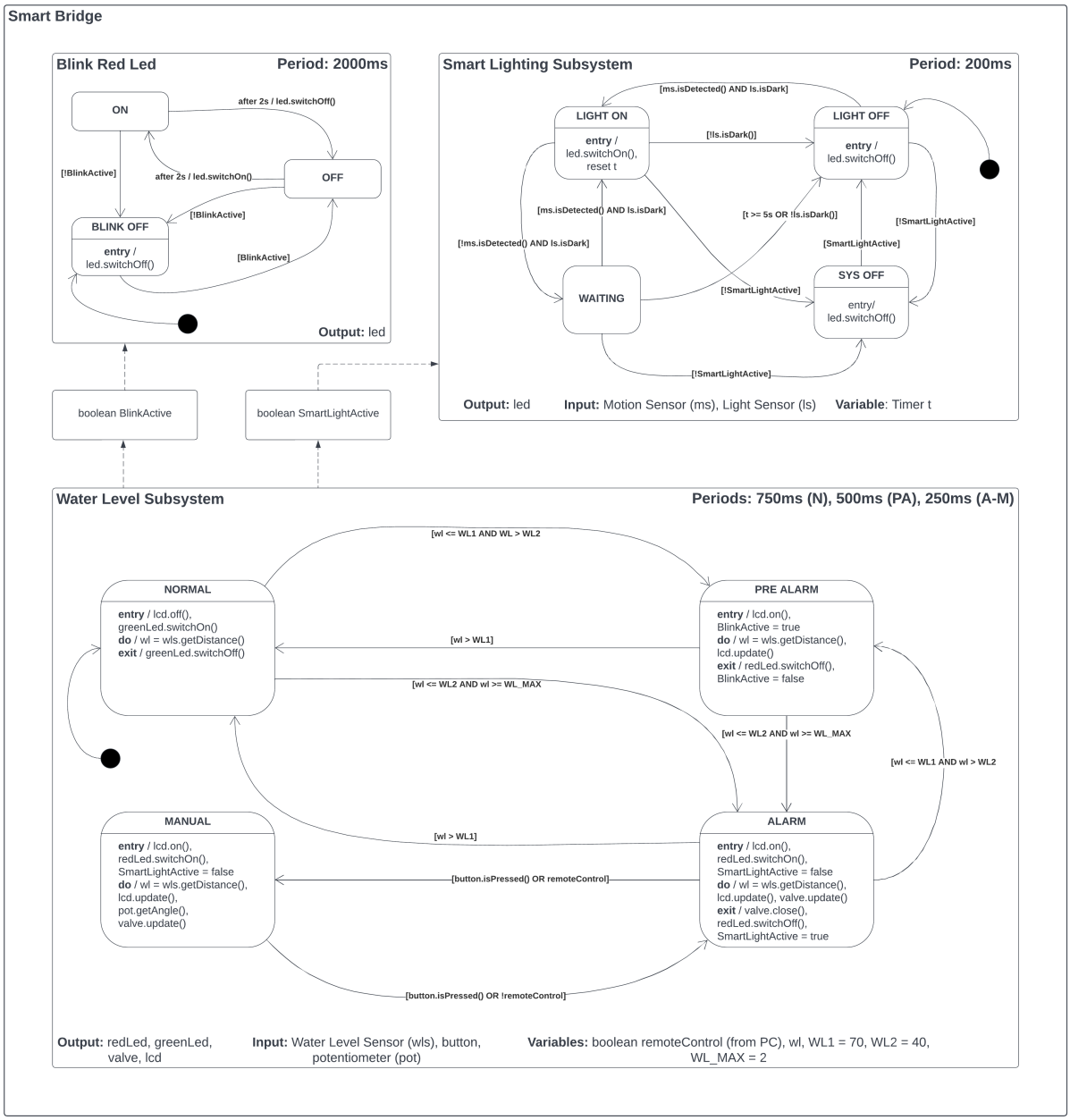

Figure 3: Diagram of the Final State Machines used in the project

The system is composed by three different synchronous FSMs, communicating with each other:

- **Smart Lighting Subsystem:** in this subsystem the sampling of the motion sensor and the light sensor is done every time the period occurs. Depending on the result of the sampling the state of the system is changed in **LIGHT ON**, **LIGHT OFF** or **WAITING**, and the led is turned on/off. Then we have a particular state called **SYS OFF** that can only be set externally by the Water Level Subsystem. We decide to keep this state inside the FSM and not to allow the other subsystem to directly turn off the system, so we can perform some action before the FSM is actually turned off.

- **Blink Red Led:** this FSM is used to made the red led blink, the FSM is active only when the Water Level Subsystem is in pre-alarm state. We introduce a particular state called **BLINK OFF** to turn off the FSM, as in the previous case. If the FSM is active every time the period occurs the led is switched on/off.

- **Water Level Subsystem:** this FSM represent the main subsystem from whose state depend the states of other FSMs. Every time the period occurs we sample the water level, and according to the obtained value we change the state in **NORMAL**, **PRE-ALARM** or **ALARM**. Then according to the current state we update the output components. The period of the FSM depend on the current state. **MANUAL** is a particular state, that can be entered only from the alarm state if the user press the button. The **MANUAL** state has the same characteristics as the ALARM one with the only difference that the valve opening angle is read from the potentiometer, instead of depending linearly from the water level.

We decide to implement the system using only three FSMs because we want to update the output devices with the same period that we use to read data from the input devices. So, we read the water level and then update all the connected devices and FSMs. We sample the light sensor and motion sensor and then update the led state.
For further details about the FSMs diagram see diagram on GitHub repo.

3.3 Arduino

The embedded software was implemented on Arduino using C++/Waring in the PlatformIO environment.
Every input/output device (apart from the button) was mapped on a C++ class that defines its behaviour. Then we created a task class to represent the FSMs behaviour, tasks are dynamic; they can be activated/deactivated and their period can change according to current state.

Then we have a scheduler that keep track of all tasks, at each period it advances the execution of each task by calling its method tick. The period of the scheduler is the greatest common divisor of the periods of the tasks.

The button is the only input device of the system that is not sampled periodically, we attach an interrupt handler to the button pin. This handler will change the Water Level Subsystem state if the system is in alarm state.

We use the serial port to communicate with the program running on the PC Console, we use simple strings communication. Strings composed by one word report the smart lighting subsystem status, while strings composed by two world report both water level and subsystem status.

For further details and to view the embedded software code see the project GitHub repo on the `src/arduino` folder.

3.4 PC

The program running on the PC Console was developed using Java (dependencies were resolved using Maven). For the serial communication we use JSSC library. The GUI was made using WindowBuilder extension for Eclipse, while the graph was created using JFreeChart library.

Figure 4: View of the application

The program consists of a simple GUI were water level, Water Level Subsystem status and Smart Lighting Subsystem status are reported. The data about water level are then insert in the graph according to the time of sampling.
The manual button can be used to change the embedded system state to MANUAL if the previous state was ALARM, then we can use the slider to change the valve angle. The system is designed so that if the manual control is taken from the breadboard only the potentiometer determines the valve angle and only if the tactile button is pressed again the system

exit the MANUAL state. In the same way, if the control is taken from the PC Console only the slider determines the valve angle and only if the console button is pressed again the system exit the MANUAL state.
The massages used to change the state on the embedded system are send using the serial port.

4 Documentation

A brief video demonstrating how the system work can be found both on my OneDrive archive or project GitHub repo

The complete code of the project can be found on the project GitHub repo.